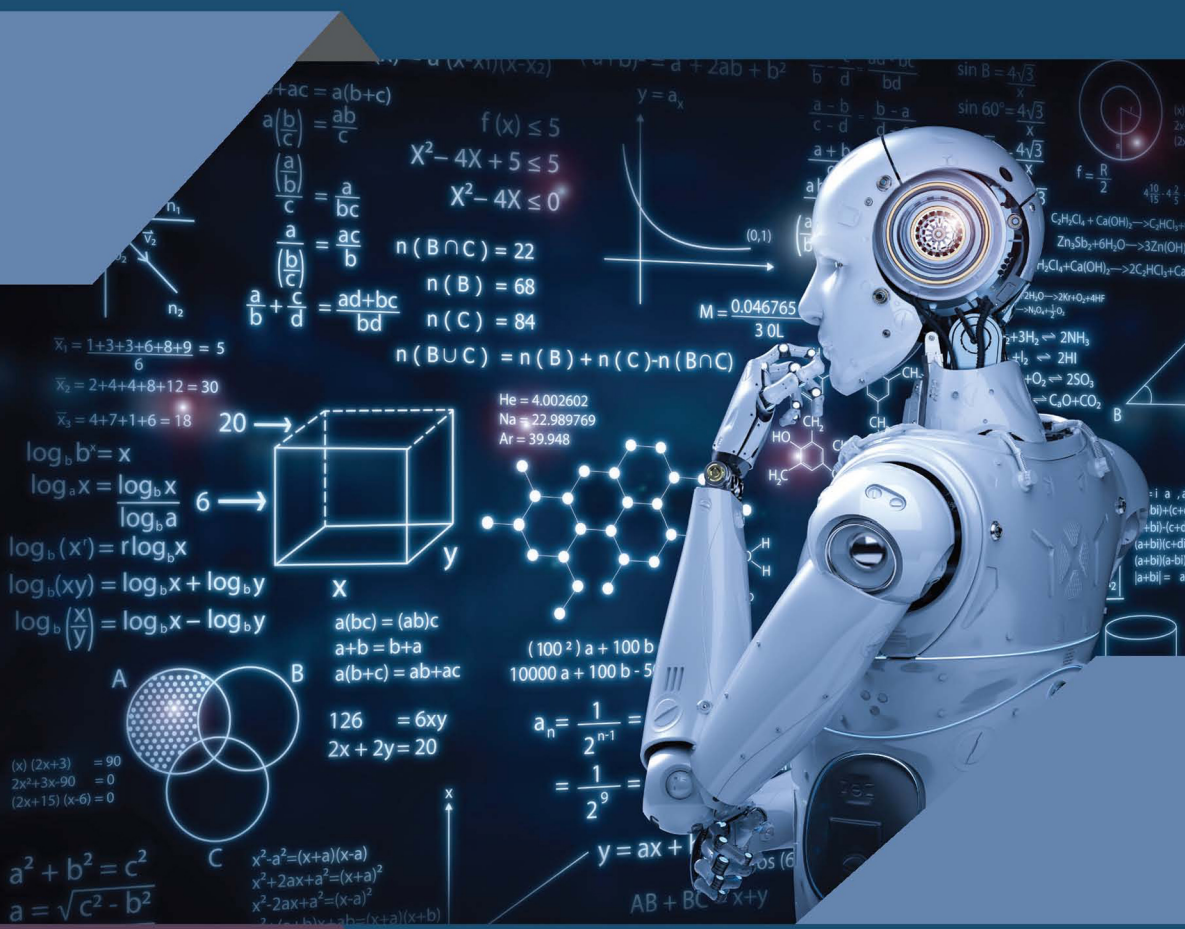
Intelligent Informatics



Edited by: **Jiquan Chen**

BIBLIOTEX
Digital Library

Intelligent Informatics

Intelligent Informatics

Editor:

Jiquan Chen

BIBLIOTEX
Digital Library
www.bibliotex.com

Intelligent Informatics

Editor: Jiquan Chen

www.bibliotex.com

email: info@bibliotex.com

e-book Edition 2022

ISBN: 978-1-98467-790-7 (e-book)

In Collaboration with Intelliz Press. Originally Published in printed book format by Intelliz Press with ISBN 978-1-68251-850-2

TABLE OF CONTENTS

PREFACE

Data is all around us. The Internet of Things (IoT) and sensors have the ability to harness large volumes of data, while artificial intelligence (AI) can learn patterns in the data to automate tasks for a variety of business benefits. Artificial Intelligence enhances the speed, precision and effectiveness of human efforts. In financial institutions, AI techniques can be used to identify which transactions are likely to be fraudulent, adopt fast and accurate credit scoring, as well as automate manually intense data management tasks. Artificial intelligence is going to change every industry, but we have to understand its limits. The principle limitation of AI is that it learns from the data. There is no other way in which knowledge can be incorporated. That means any inaccuracies in the data will be reflected in the results. And any additional layers of prediction or analysis have to be added separately. Artificial intelligence (AI) has become a technological reality for businesses and organizations across industries. Even if its benefits may not be always easy to quantify, AI has proven itself capable of improving process efficiency, reducing human errors and labor, and extracting insights from big data.

Intelligence and security informatics (ISI) is an emerging field of study aimed at developing advanced information technologies, systems, algorithms, and databases for national - and homeland-security-related applications, through an integrated technological, organizational, and

policy-based approach. This book summarizes the broad application and policy context for this emerging field. On a smaller scale, AI has made it into many people's lives in one form or another, even if its presence is not always recognized. To better understand the state of AI, let's take a closer look at the role AI plays in business and in many people's lives even as we speak. It will cover fundamental principles and algorithms of intelligent informatics. It also deals with understand the data collection, data integration, data cleansing, data representation, model construction, model selection, model averaging, model evaluation, and model interpretation components in intelligent informatics.

CHAPTER 1

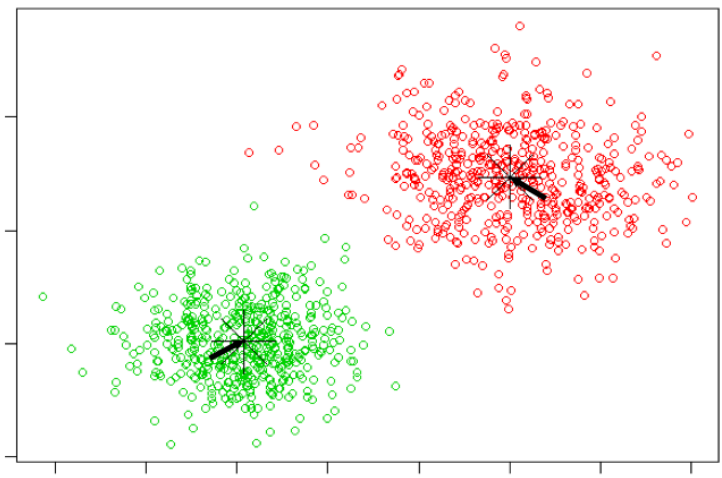
DATA MINING, CLUSTERING AND INTELLIGENT INFORMATION SYSTEMS

INTRODUCTION

Data mining is a process used by companies to turn raw data into useful information. By using software to look for patterns in large batches of data, businesses can learn more about their customers to develop more effective marketing strategies, increase sales and decrease costs. Data mining depends on effective data collection, warehousing, and computer processing.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Let's understand this with an example. Suppose, you are the head of a rental store and wish to understand preferences of your costumers to scale up your business. Is it possible for you to look at details of each costumer and devise a unique business strategy for each one of them? Definitely not. But, what you can do is to cluster all of your costumers into say 10 groups based on their purchasing habits and use a separate strategy for costumers in each of these 10 groups. And this is what we call clustering.



Intelligent Information Systems (IIS) can be defined as the next generation of Information Systems (IS) developed as a result of integration of AI and database (DB) technologies. IIS embody knowledge that allows them to exhibit intelligent behavior, allows them to cooperate with users and other systems in problem solving, discovery, retrieval, and manipulation of data and knowledge. For any IIS to serve its purpose, the information must be available when it is needed. This means that the computing systems used to store data and process the information, and the security controls used to protect it must be functioning correctly.

1.1 DATA MINING

Data mining is the process of finding anomalies, patterns and correlations within large data sets to predict outcomes. Using a broad range of techniques, you can use this information to increase revenues, cut costs, improve customer relationships, reduce risks and more. Data mining involves exploring and analyzing large blocks of information to glean meaningful patterns and trends. It can be used in a variety of ways, such as database marketing, credit risk management, fraud detection, spam Email filtering, or even to discern the sentiment or opinion of users.

The data mining process breaks down into five steps. First, organizations collect data and load it into their data warehouses. Next, they store and manage the data, either on in-house servers or the cloud. Business analysts, management teams and information technology professionals access the data and determine how they want to organize it. Then, application software sorts the data based on the user's results, and finally, the end-user presents the data in an easy-to-share format, such as a graph or table.

1.1.1 Meaning

The process of extracting information to identify patterns, trends, and useful data that would allow the business to take the data-driven decision from huge sets of data is called Data Mining.

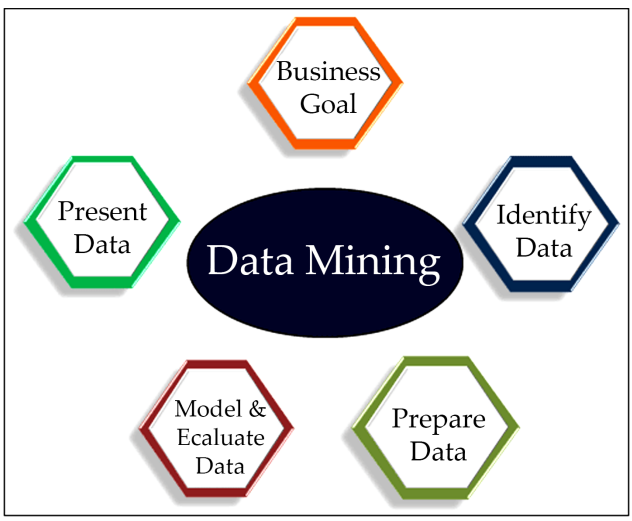
In other words, we can say that Data Mining is the process of investigating hidden patterns of information to various perspectives for categorization into useful data, which is collected and assembled in particular areas such as data warehouses, efficient analysis, data mining algorithm, helping decision making and other data requirement to eventually cost-cutting and generating revenue.

Data mining is the act of automatically searching for large stores of information to find trends and patterns that go beyond simple

analysis procedures. Data mining utilizes complex mathematical algorithms for data segments and evaluates the probability of future events. Data Mining is also called Knowledge Discovery of Data (KDD).

Data Mining is a process used by organizations to extract specific data from huge databases to solve business problems. It primarily turns raw data into useful information.

Data Mining is similar to Data Science carried out by a person, in a specific situation, on a particular data set, with an objective. This process includes various types of services such as text mining, web mining, audio and video mining, pictorial data mining, and social media mining. It is done through software that is simple or highly specific. By outsourcing data mining, all the work can be done faster with low operation costs. Specialized firms can also use new technologies to collect data that is impossible to locate manually. There are tonnes of information available on various platforms, but very little knowledge is accessible. The biggest challenge is to analyze the data to extract important information that can be used to solve a problem or for company development. There are many powerful instruments and techniques available to mine data and find better insight from it.

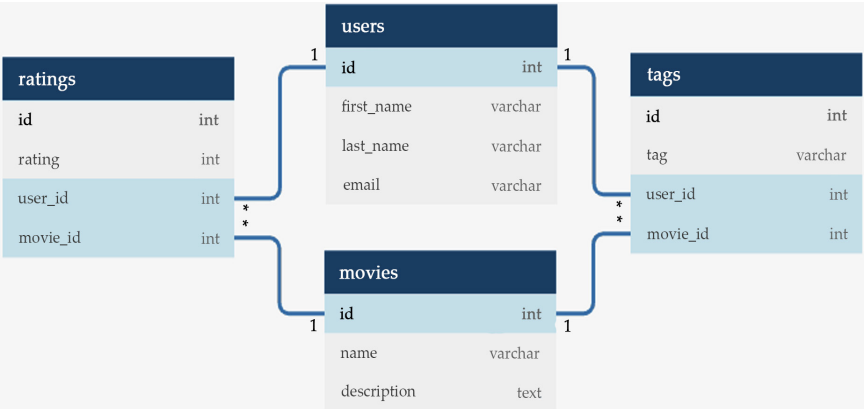


1.1.2 Types of Data Mining

Data mining can be performed on the following types of data:

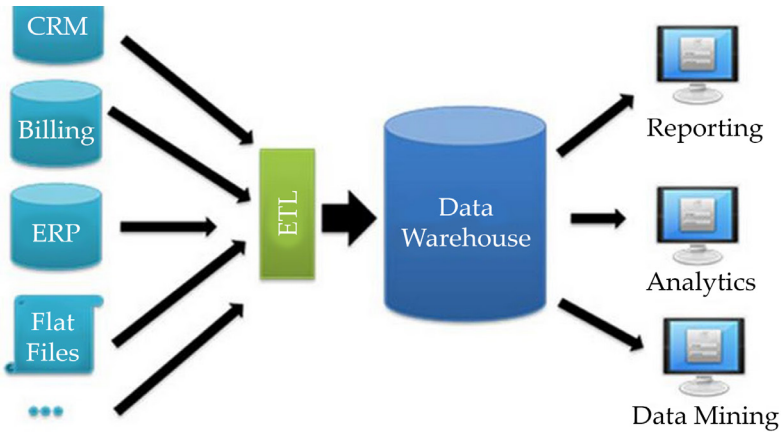
Relational Database

A relational database is a collection of multiple data sets formally organized by tables, records, and columns from which data can be accessed in various ways without having to recognize the database tables. Tables convey and share information, which facilitates data searchability, reporting, and organization.



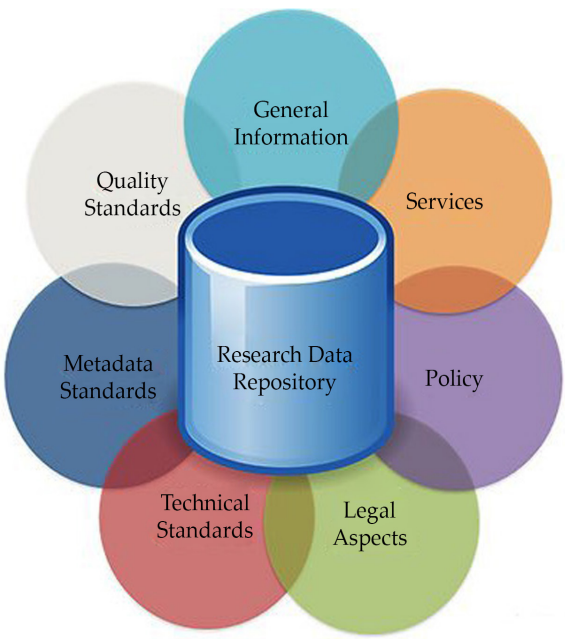
Data warehouses

A Data Warehouse is the technology that collects the data from various sources within the organization to provide meaningful business insights. The huge amount of data comes from multiple places such as Marketing and Finance. The extracted data is utilized for analytical purposes and helps in decision- making for a business organization. The data warehouse is designed for the analysis of data rather than transaction processing.



Data Repositories

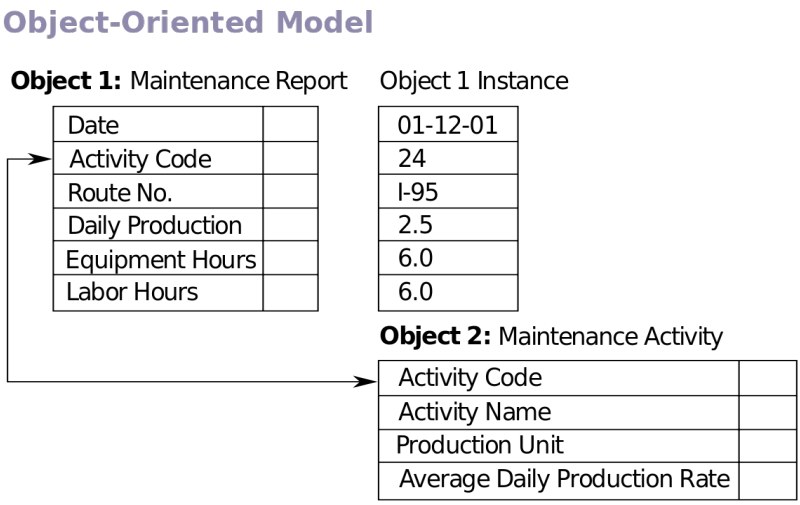
The Data Repository generally refers to a destination for data storage. However, many IT professionals utilize the term more clearly to refer to a specific kind of setup within an IT structure. For example, a group of databases, where an organization has kept various kinds of information.



Object-Relational Database

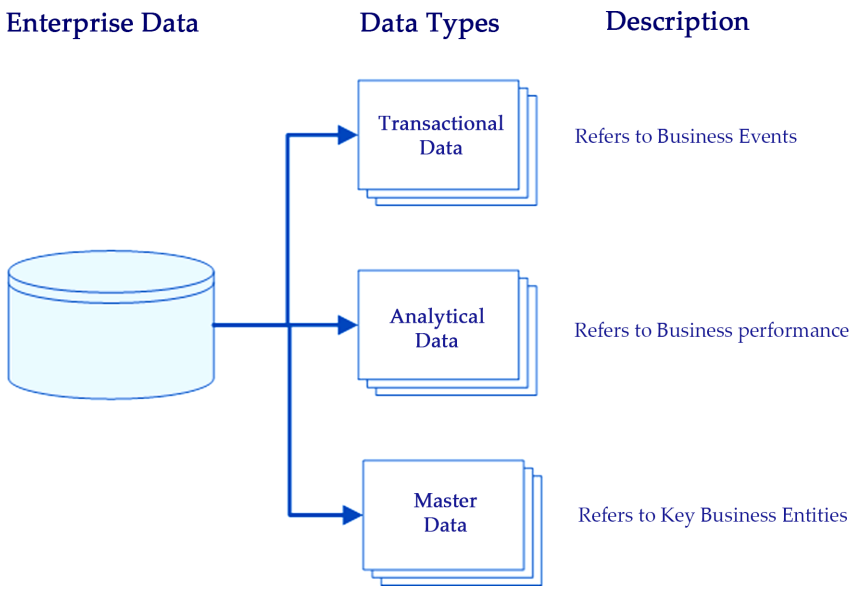
A combination of an object-oriented database model and relational database model is called an object-relational model. It supports Classes, Objects, Inheritance, etc.

One of the primary objectives of the Object-relational data model is to close the gap between the Relational database and the object-oriented model practices frequently utilized in many programming languages, for example, C++, Java, C#, and so on.



Transactional Database

A transactional database refers to a database management system (DBMS) that has the potential to undo a database transaction if it is not performed appropriately. Even though this was a unique capability a very long while back, today, most of the relational database systems support transactional database activities.



1.1.3 The Data Mining Process

Figure 1 illustrates the phases, and the iterative nature, of a data mining project. The process flow shows that a data mining project does not stop when a particular solution is deployed. The results of data mining trigger new business questions, which in turn can be used to develop more focused models.

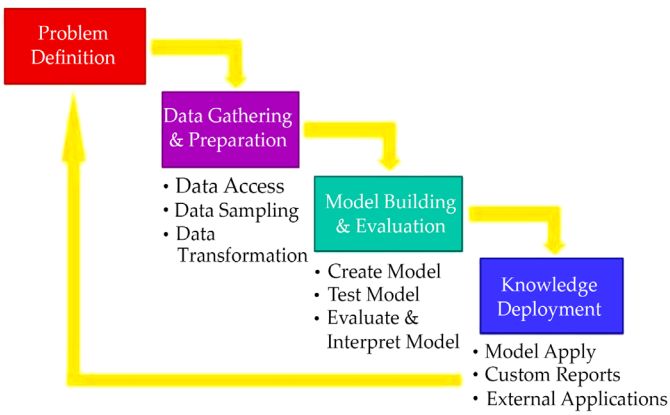


Figure 1: The Data Mining Process

Problem Definition

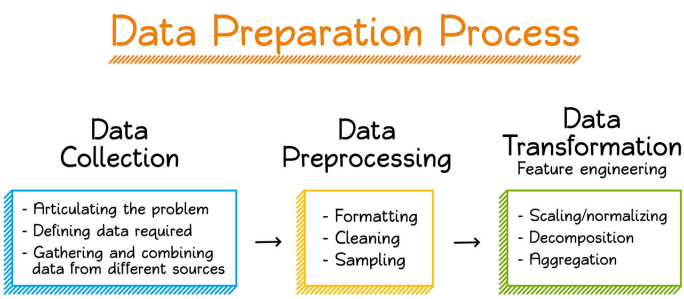
This initial phase of a data mining project focuses on understanding the project objectives and requirements. Once you have specified the project from a business perspective, you can formulate it as a data mining problem and develop a preliminary implementation plan.

For example, your business problem might be: "How can I sell more of my product to customers?" You might translate this into a data mining problem such as: "Which customers are most likely to purchase the product?" A model that predicts who is most likely to purchase the product must be built on data that describes the customers who have purchased the product in the past. Before building the model, you must assemble the data that is likely to contain relationships between customers who have purchased the product and customers who have not purchased the product. Customer attributes might include age, number of children, years of residence, owners/renters, and so on.

Data Gathering and Preparation

The data understanding phase involves data collection and exploration. As you take a closer look at the data, you can determine how well it addresses the business problem. You might decide to remove some of the data or add additional data. This is also the time to identify data quality problems and to scan for patterns in the data.

The data preparation phase covers all the tasks involved in creating the case table you will use to build the model. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, case, and attribute selection as well as data cleansing and transformation. For example, you might transform a DATE_OF_BIRTH column to AGE; you might insert the average income in cases where the INCOME column is null.



Additionally you might add new computed attributes in an effort to tease information closer to the surface of the data. For example, rather than using the purchase amount, you might create a new attribute: "Number of Times Amount Purchase Exceeds $500 in a 12 month time period." Customers who frequently make large purchases may also be related to customers who respond or don't respond to an offer.

Thoughtful data preparation can significantly improve the information that can be discovered through data mining.

Model Building and Evaluation

In this phase, you select and apply various modeling techniques and calibrate the parameters to optimal values. If the algorithm requires data transformations, you will need to step back to the previous phase to implement them.

In preliminary model building, it often makes sense to work with a reduced set of data (fewer rows in the case table), since the final case table might contain thousands or millions of cases.

At this stage of the project, it is time to evaluate how well the model satisfies the originally-stated business goal (phase 1). If the model is supposed to predict customers who are likely to purchase a product, does it sufficiently differentiate between the two classes? Is there sufficient lift? Are the trade-offs shown in the confusion matrix acceptable? Would the model be improved by adding text

data? Should transactional data such as purchases (market-basket data) be included? Should costs associated with false positives or false negatives be incorporated into the model?

Knowledge Deployment

Knowledge deployment is the use of data mining within a target environment. In the deployment phase, insight and actionable information can be derived from data.

Deployment can involve scoring (the application of models to new data), the extraction of model details (for example the rules of a decision tree), or the integration of data mining models within applications, data warehouse infrastructure, or query and reporting tools.

Because Oracle Data Mining builds and applies data mining models inside Oracle Database, the results are immediately available. BI reporting tools and dashboards can easily display the results of data mining. Additionally, Oracle Data Mining supports scoring in real time: Data can be mined and the results returned within a single database transaction. For example, a sales representative could run a model that predicts the likelihood of fraud within the context of an online sales transaction.

1.1.4 Advantages of Data Mining

Data is pouring into businesses in a multitude of formats at unprecedented speeds and volumes. Being a data-driven business is no longer an option; the business' success depends on how quickly you can discover insights from big data and incorporate them into business decisions and processes, driving better actions across your enterprise. However, with so much data to manage, this can seem like an insurmountable task.

Data mining empowers businesses to optimize the future by understanding the past and present, and making accurate predictions about what is likely to happen next.

For example, data mining can tell you which prospects are likely to become profitable customers based on past customer profiles, and which are most likely to respond to a specific offer. With this knowledge, you can increase your return on investment (ROI) by making your offer to only those prospects likely to respond and become valuable customers.

You can use data mining to solve almost any business problem that involves data, including:

- Increasing revenue.
- Understanding customer segments and preferences.
- Acquiring new customers.
- Improving cross-selling and up-selling.
- Retaining customers and increasing loyalty.
- Increasing ROI from marketing campaigns.
- Detecting fraud.
- Identifying credit risks.
- Monitoring operational performance.

Through the application of data mining techniques, decisions can be based on real business intelligence — rather than instinct or gut reactions — and deliver consistent results that keep businesses ahead of the competition.

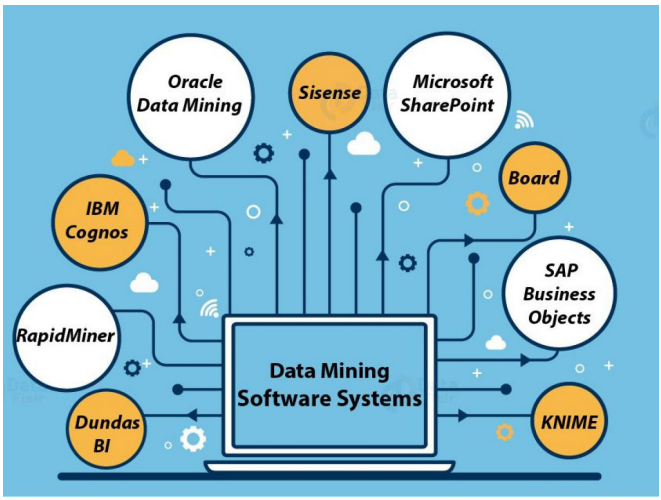
As large-scale data processing technologies such as machine learning and artificial intelligence become more readily accessible, companies are now able to dig through terabytes of data in minutes or hours, rather than days or weeks, helping them innovate and grow faster.

1.1.5 Disadvantages of Data Mining

- There is a probability that the organizations may sell useful data of customers to other organizations for money. As per the report, American Express has sold credit card purchases of their customers to other organizations.
- Many data mining analytics software is difficult to operate and needs advance training to work on.
- Different data mining instruments operate in distinct ways due to the different algorithms used in their design. Therefore, the selection of the right data mining tools is a very challenging task.
- The data mining techniques are not precise, so that it may lead to severe consequences in certain conditions.

1.1.6 Data Mining Software and Tools

There is no doubt that data mining has the power to transform enterprises; however, implementing a solution that meets the needs of all stakeholders can frequently stall platform selection. The wide range of options available to analysts, including open source languages such as R and Python and with familiar tools like Excel, combined with the diversity and complexity of tools and algorithms, can further complicate the process.



Businesses that gain the most value from data mining typically select a platform that:

- Incorporates best practices for their industry or type of project. Healthcare organizations, for example, have different needs than e-commerce companies.
- Manages the entire data mining lifecycle, from data exploration to production.
- Aligns with the enterprise applications, including BI systems, CRM, ERP, financial, and other enterprise software it must interoperate with for maximum return on investment.
- Integrates with leading open source languages, providing developers and data scientists with the flexibility and collaboration tools to create innovative applications.

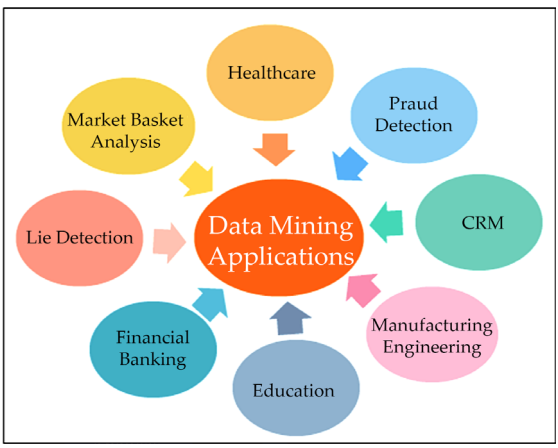
- Meets the needs of IT, data scientists, and analysts, while also serving the reporting and visualization needs of business users<

The Talend Big Data Platform provides a complete suite of data management and data integration capabilities to help data mining teams respond more quickly to the needs of their business.

Based on an open, scalable architecture and with tools for relational databases, flat files, cloud apps, and platforms, this solution complements your data mining platform by putting more data to work in less time — which translates into faster time to insight and competitive advantage.

1.1.7 Data Mining Applications

Data Mining is primarily used by organizations with intense consumer demands- Retail, Communication, Financial, marketing company, determine price, consumer preferences, product positioning, and impact on sales, customer satisfaction, and corporate profits. Data mining enables a retailer to use point-of-sale records of customer purchases to develop products and promotions that help the organization to attract the customer.



These are the following areas where data mining is widely used:

Data Mining in Healthcare

Data mining in healthcare has excellent potential to improve the health system. It uses data and analytics for better insights and to identify best practices that will enhance health care services and reduce costs. Analysts use data mining approaches such as Machine learning, Multi-dimensional database, Data visualization, Soft computing, and statistics. Data Mining can be used to forecast patients in each category. The procedures ensure that the patients get intensive care at the right place and at the right time. Data mining also enables healthcare insurers to recognize fraud and abuse.

Data Mining in Market Basket Analysis

Market basket analysis is a modeling method based on a hypothesis. If you buy a specific group of products, then you are more likely to buy another group of products. This technique may enable the retailer to understand the purchase behavior of a buyer. This data may assist the retailer in understanding the requirements of the buyer and altering the store's layout accordingly. Using a different analytical comparison of results between various stores, between customers in different demographic groups can be done.

Data mining in Education

Education data mining is a newly emerging field, concerned with developing techniques that explore knowledge from the data generated from educational Environments. EDM objectives are recognized as affirming student's future learning behavior, studying the impact of educational support, and promoting learning science. An organization can use data mining to make precise decisions and also to predict the results of the student. With the results, the institution can concentrate on what to teach and how to teach.

Data Mining in Manufacturing Engineering

Knowledge is the best asset possessed by a manufacturing company. Data mining tools can be beneficial to find patterns in a complex manufacturing process. Data mining can be used in system-level designing to obtain the relationships between product architecture, product portfolio, and data needs of the customers. It can also be used to forecast the product development period, cost, and expectations among the other tasks.

Data Mining in CRM (Customer Relationship Management)

Customer Relationship Management (CRM) is all about obtaining and holding Customers, also enhancing customer loyalty and implementing customer-oriented strategies. To get a decent relationship with the customer, a business organization needs to collect data and analyze the data. With data mining technologies, the collected data can be used for analytics.

Data Mining in Fraud detection

Billions of dollars are lost to the action of frauds. Traditional methods of fraud detection are a little bit time consuming and sophisticated. Data mining provides meaningful patterns and turning data into information. An ideal fraud detection system should protect the data of all the users. Supervised methods consist of a collection of sample records, and these records are classified as fraudulent or non-fraudulent. A model is constructed using this data, and the technique is made to identify whether the document is fraudulent or not.

Data Mining in Lie Detection

Apprehending a criminal is not a big deal, but bringing out the truth from him is a very challenging task. Law enforcement may use data mining techniques to investigate offenses, monitor suspected terrorist communications, etc. This technique includes

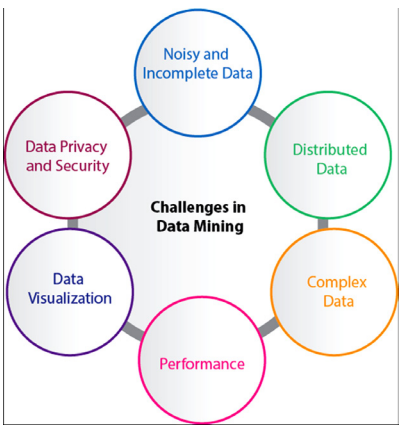
text mining also, and it seeks meaningful patterns in data, which is usually unstructured text. The information collected from the previous investigations is compared, and a model for lie detection is constructed.

Data Mining Financial Banking

The Digitalization of the banking system is supposed to generate an enormous amount of data with every new transaction. The data mining technique can help bankers by solving business-related problems in banking and finance by identifying trends, casualties, and correlations in business information and market costs that are not instantly evident to managers or executives because the data volume is too large or are produced too rapidly on the screen by experts. The manager may find these data for better targeting, acquiring, retaining, segmenting, and maintain a profitable customer.

1.1.8 Challenges of Implementation in Data mining

Although data mining is very powerful, it faces many challenges during its execution. Various challenges could be related to performance, data, methods, and techniques, etc. The process of data mining becomes effective when the challenges or problems are correctly recognized and adequately resolved.



Incomplete and noisy data

The process of extracting useful data from large volumes of data is data mining. The data in the real-world is heterogeneous, incomplete, and noisy. Data in huge quantities will usually be inaccurate or unreliable. These problems may occur due to data measuring instrument or because of human errors. Suppose a retail chain collects phone numbers of customers who spend more than $ 500, and the accounting employees put the information into their system. The person may make a digit mistake when entering the phone number, which results in incorrect data. Even some customers may not be willing to disclose their phone numbers, which results in incomplete data. The data could get changed due to human or system error. All these consequences (noisy and incomplete data)makes data mining challenging.

Data Distribution

Real-worlds data is usually stored on various platforms in a distributed computing environment. It might be in a database, individual systems, or even on the internet. Practically, It is a quite tough task to make all the data to a centralized data repository mainly due to organizational and technical concerns. For example, various regional offices may have their servers to store their data. It is not feasible to store, all the data from all the offices on a central server. Therefore, data mining requires the development of tools and algorithms that allow the mining of distributed data.

Complex Data

Real-world data is heterogeneous, and it could be multimedia data, including audio and video, images, complex data, spatial data, time series, and so on. Managing these various types of data and extracting useful information is a tough task. Most of the time, new technologies, new tools, and methodologies would have to be refined to obtain specific information.

Performance

The data mining system's performance relies primarily on the efficiency of algorithms and techniques used. If the designed algorithm and techniques are not up to the mark, then the efficiency of the data mining process will be affected adversely.

Data Privacy and Security

Data mining usually leads to serious issues in terms of data security, governance, and privacy. For example, if a retailer analyzes the details of the purchased items, then it reveals data about buying habits and preferences of the customers without their permission.

Data Visualization

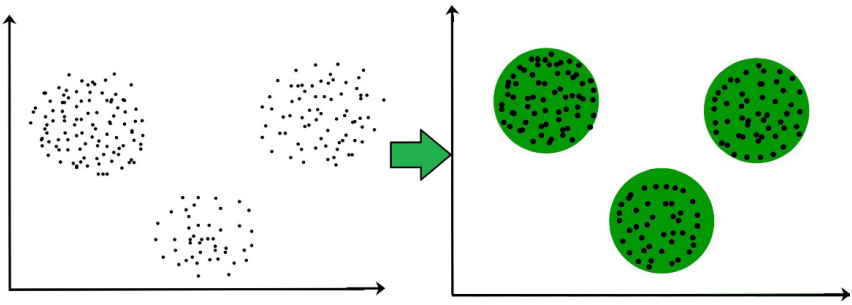
In data mining, data visualization is a very important process because it is the primary method that shows the output to the user in a presentable way. The extracted data should convey the exact meaning of what it intends to express. But many times, representing the information to the end-user in a precise and easy way is difficult. The input data and the output information being complicated, very efficient, and successful data visualization processes need to be implemented to make it successful. There are many more challenges in data mining in addition to the problems above-mentioned. More problems are disclosed as the actual data mining process begins, and the success of data mining relies on getting rid of all these difficulties.

1.2 CLUSTERING

It is basically a type of *unsupervised learning method* . An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For ex– The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.



It is not necessary for clusters to be a spherical. Such as :

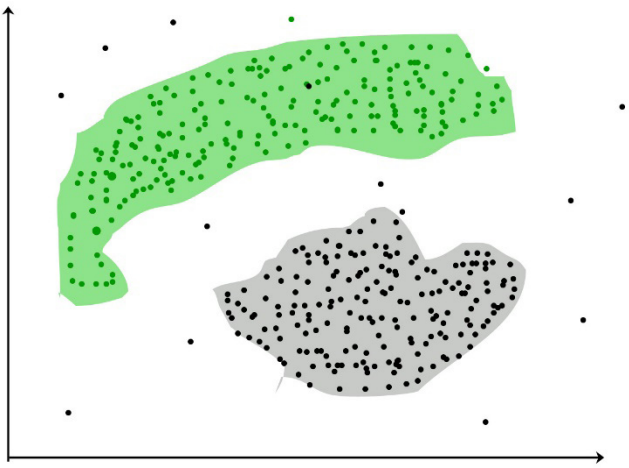


Figure 2: DBSCAN. Density-based Spatial Clustering of Applications with Noise.

These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster centre.

Various distance methods and techniques are used for calculation of the outliers.

1.2.1 Why Clustering ?

Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding "natural clusters" and describe their unknown properties ("natural" data types), in finding useful and suitable groupings ("useful" data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters.

1.2.2 Types of Clustering

Broadly speaking, clustering can be divided into two subgroups :

- **Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.
- **Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store

1.2.3 Clustering Methods

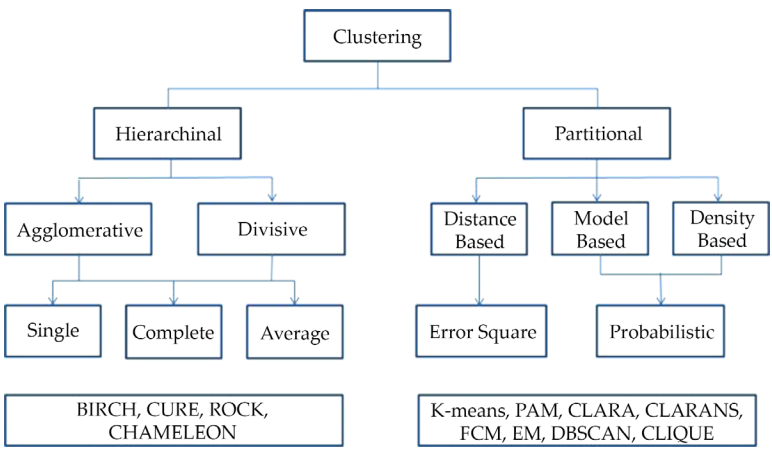
- **Density-Based Methods :** These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space.

These methods have good accuracy and ability to merge two clusters.Example *DBSCAN (Density-Based Spatial Clustering of Applications with Noise) , OPTICS (Ordering Points to Identify Clustering Structure)* etc.

- **Hierarchical Based Methods :** The clusters formed in this method forms a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category
  - **Agglomerative** (*bottom up approach*)
  - **Divisive** (*top down approach*)

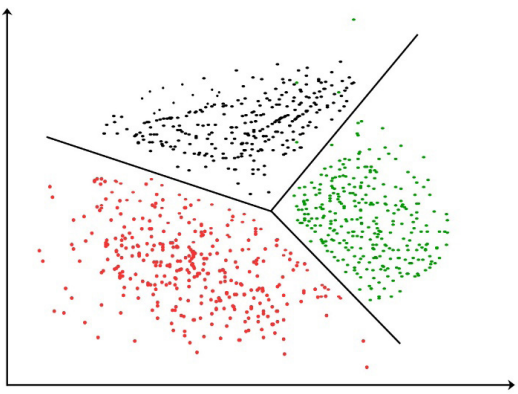
examples *CURE (Clustering Using Representatives), BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies)* etc.

- **Partitioning Methods :** These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example *K-means, CLARANS (Clustering Large Applications based upon Randomized Search)* etc.
- **Grid-based Methods :** In this method the data space is formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example *STING (Statistical Information Grid), wave cluster, CLIQUE (CLustering In Quest)* etc.



1.2.4 Clustering Algorithms

K-means clustering algorithm – It is the simplest unsupervised learning algorithm that solves clustering problem.K-means algorithm partition n observations into k clusters where each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster .



Types of clustering algorithms

Since the task of clustering is subjective, the means that can be used for achieving this goal are plenty. Every methodology follows a different set of rules for defining the '*similarity*' among data points. In fact, there are more than 100 clustering algorithms known. But few of the algorithms are used popularly, let's look at them in detail:

- **Connectivity models:** As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These

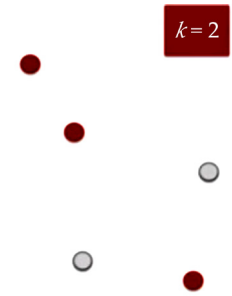
models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.

- **Centroid models:** These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category. In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset. These models run iteratively to find the local optima.
- **Distribution models:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.
- **Density Models:** These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.

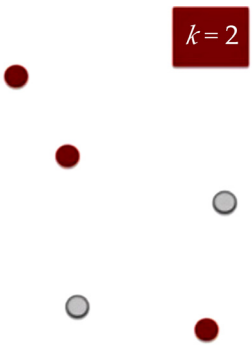
K Means Clustering

K means is an iterative clustering algorithm that aims to find local maxima in each iteration. This algorithm works in these 5 steps :

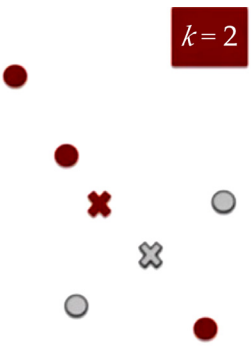
1. Specify the desired number of clusters K : Let us choose k=2 for these 5 data points in 2-D space.



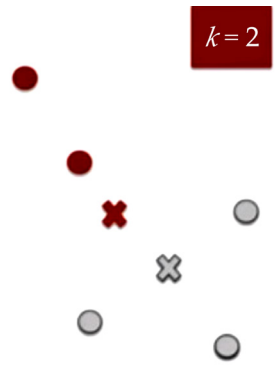
2. Randomly assign each data point to a cluster : Let's assign three points in cluster 1 shown using red color and two points in cluster 2 shown using grey color.



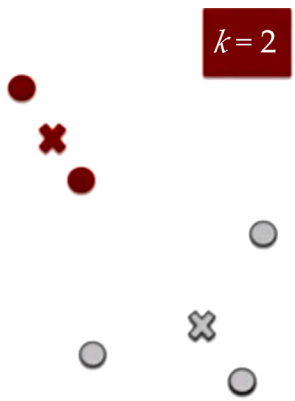
3. Compute cluster centroids : The centroid of data points in the red cluster is shown using red cross and those in grey cluster using grey cross.



4. Re-assign each point to the closest cluster centroid : Note that only the data point at the bottom is assigned to the red cluster even though its closer to the centroid of grey cluster. Thus, we assign that data point into grey cluster



5. Re-compute cluster centroids : Now, re-computing the centroids for both the clusters.



6. Repeat steps 4 and 5 until no improvements are possible : Similarly, we'll repeat the 4^{th} and 5^{th} steps until we'll reach global optima. When there will be no further switching of data points between two clusters for two successive repeats. It will mark the termination of the algorithm if not explicitly mentioned.

1.2.5 Applications of Clustering in different fields

- **Marketing:** It can be used to characterize & discover customer segments for marketing purposes.
- **Biology:** It can be used for classification among different species of plants and animals.
- **Libraries:** It is used in clustering different books on the basis of topics and information.
- **Insurance:** It is used to acknowledge the customers, their policies and identifying the frauds.
- **City Planning:** It is used to make groups of houses and to study their values based on their geographical locations and other factors present.
- **Earthquake studies:** By learning the earthquake-affected areas we can determine the dangerous zones.

1.3 INTELLIGENT INFORMATION SYSTEMS

The recent years of manufacturing research and development have been governed by the trends of Artificial Intelligence, database management, and information systems (IS).

Intelligent Information System (IIS) represents the evolution of knowledge. It has been led by modern technology integrations, data processing, and distribution in multiple computational environments.

Machines are programmed perfectly for a fixed type of environment to provide an expected result by processing a particular data in a given manner.

The information retrieved depends upon the perfect input and situations that the machine was programmed to deal with, and resultantly shows little to no intelligence. Intelligent information systems are aspired to work like an evolved human brain. With changing circumstances, a tool needs to be optimal at decision making.

It must be able to adapt and change themselves according to the data and information fed with the least cost and effort.

With Intelligent Information Systems, a group of users, devices, and data could coordinate their actions together for decision making without needing a third party.

For it to work flawlessly, the system ought to be fed with refined substantiated and ratified data & information.

With higher cognitive science, natural language processing, machine learning, semantic technologies, and knowledge-based systems (KBS) technologies, IIS has been advancing a lot in the last few years.

IIS also employs other data related techniques like data mining, data sharing and warehousing, analytics, image analysis, information logistics, and knowledge discovery.

IIS is a smart information processing that uses neural networks and IS with advanced data analysis. It enables the user without having any expertise in problem-solving analysis to take evidence-based decisions in complex situations.

With all the information and artificial neural network tools, IIS embodies intelligent strategic, tactical, and operational decision-making skills.

With cutting edge technologies in the bay, IIS implementation is seeing a boost in popularity for problem assessment.

The system is now in various domains, including marketing, financial decision makings, digital multimedia forensics, personal data analysis, healthcare systems, media archiving and production, and predictive modeling.

IIS enables organizations to attain complex tasks with relative ease and in lesser overall cost and creates adaptive and intelligent workflows.

Apart from decision support, IIS concerns with executive information, management information, and transaction processing systems.

It adapts itself perfectly with the changing programming environment while assisting in remote monitoring and management with its problem solving and decision making tools and techniques.

With a high level of data granularity, the IIS expert system exhibits intelligent behavior.

Discovery, access, retrieval, and manipulation of a large variety of data and knowledge processed and stored in previous processes help them with optimal decision making in future tasks.

Complex statistical methods into machine learning display the information when required.

An intelligent Information System can gather and analyze data and communicate with other systems, even with a changing environment.

With the help of ANN, deep learning, and other related techniques, the system gathers and analyses data while learning from the past results, data, security, and connectivity issues. With its effective mechanism, the system determines the actions to be taken if any conditions occur.

An intelligent information system can further accurately predict the process timing and resources needed. Thus, it operates accordingly and prepares and stores the result before the retrieval request arrives, ready for quick access.

With all of this, it's possible to converge, diverge, extract, and process tons of data and generate probabilities and higher prediction accuracy.

1.3.1 Examples of Intelligent Systems

If intelligence is the ability to adapt and change with changing circumstances all life-forms show intelligence. Life-forms adapt and change and learn to fit in with their environment for the life-form to survive with the least cost and effort. Individuals do not survive, but the species survive. We call the process evolution. Individuals change in small steps, and those best fitted to the environment survive longest and are more likely to reproduce.

Humans have evolved to communicate knowledge between individuals. The species is able to pass survival skills between members of the species through the information system we call language. We think the most advanced information system that exhibits intelligence is the human brain. If this is the case, then we would be wise to build our information systems to operate in a similar way to the brain.

People have long thought of computers as being electronic brains, and they have the capability of becoming more brain-like, but only if we structure the information systems to work more like a human brain. The good news is that we can make intelligent information systems from all existing information systems that use computers as a tool by making the information systems brain-like.

1.3.2 Heuristics and Information Systems

Our brains and life-forms information systems use heuristics to make decisions and to take action. Heuristics are shortcuts that reduce the cost of making a decision. It is too expensive to work out the ramifications of any decision and so we invent shortcuts or heuristics to reduce the cost.

Examples in the brain include using a rule of thumb, an educated guess, an intuitive judgment, guesstimate, stereotyping, profiling, or common sense. In life-forms, the rule of thumb is survival of the fittest. It is a way of communicating information between generations. Changes that reduce the chance of an individual surviving means those changes disappear from the life-form.

Today almost all information systems use life-form heuristics. Information systems work by repeating what worked last time. If it doesn't work, we change the information system so that it will work as expected and hope that it doesn't have secondary effects or emergent properties that give different outcomes in other situations. Anyone who has ever tried to change a large information system knows they are very, very difficult to change.

We need to find better ways for information systems to evolve. We need to go beyond machine heuristics.

1.3.3 Metaheuristics and Information Systems

A metaheuristic is a heuristic designed to find, generate, or select a heuristic to provide a sufficiently good solution to an optimisation problem, especially when we have incomplete or imperfect

information or limited computation capacity. We use search metaheuristics to model information systems, and the techniques are widely known and used in forecasting. Engineering and Science use them to get answers to hard problems that have no known analytic solution. Metaheuristic solutions dynamically change the heuristics used over time.

Weather forecasting uses metaheuristics to predict the weather by modelling the atmosphere as a complex adaptive system. It works by dividing the atmosphere into cells and then treating each cell as an independent entity. Each cell interacts with neighbouring cells. The outcome of the interaction is probabilistic as it is uncertain how the cells will interact. The weather forecasting program runs many many simulations starting from known conditions and plot the results. Forecasters analyse the results and come up with a prediction and probabilities for temperature, rain, and wind.

Our information systems can operate the same way as weather models. In an information system for the economy, we have individual entities who interact with other entities. Each entity acts in a way dependent on what happened previously but how it acts depends on how other entities act. The rules it uses to determine how to act can be metaheuristics. The approach contrasts with machine information systems that use fixed heuristics to determine what action each entity will take with the passing of time.

We can build our information systems to operate with metaheuristics rather than fixed heuristics. When we do, information systems become low-cost and adaptable. Some examples of how to build intelligent information systems follow.

1.3.4 Information Systems to Fund Infrastructure

Typically infrastructure is constructed by a single organisation for use by a large number of people. The funding to build infrastructure comes from large third parties such as investors and financial organisations. We do this because of difficulties organising all the users to fund the infrastructure over many years. A large stable

funding organisation accommodates changes to the environment by charging a fee for the use of money. To provide certainty the funding organisation uses machine information systems as it needs to remain stable and predictable.

The organisation that owns the infrastructure is often a local government organisation, and they pay investors for the use of their money to pay the builders. User fees on the infrastructure cover the cost of construction and the cost of operation. Again the information systems of the government are stable and machine like. For a single infrastructure investment we can remove the need for the machine information systems of a financing body and the government. We can allow the people who use the infrastructure and who provide the funds to work together using an intelligent information system.

Metaheuristic intelligent information systems provide a way for a large number of people to coordinate their actions without the need for a third party funding organisation and outside the complexities of government. It means the people who use the infrastructure can fund the construction and the operation of the infrastructure and save the cost of renting money. For long-lasting infrastructure, it reduces the cost of providing infrastructure by at least a half and often more. The operation of metaheuristic information systems is low cost because they dynamically adapt and modify themselves with the passing of time and the changes to the government environment. It is important that the governance structures of metaheuristic information systems are themselves metaheuristic information systems. Without this, the information systems soon degenerate to heuristic systems as some entities within the systems impose heuristics favourable to them at the expense of other entities.

1.3.5 A Metaheuristic for Identity

Identity is the ability of a person to distinguish another person from all other persons. A metaheuristic for people is the mutual identification of two people. If a person identifies a person and

vice versa then if they meet again they mutually identify each other. A common heuristics is mutual facial recognition of past encounters. Other heuristics are common memories or common acquaintances. A metaheuristic identity system builds an identity on any combination of heuristics of mutual identification. It enables the success or failure of any heuristic to pass on identity information. Electronic communications adds many more heuristics for mutual identification

Mutual identification can apply to things or animals or people. So a person who uses a car identifies themselves to the car through the use of the heuristic of starting the car with a key and with an independent mutual identification of a location heuristic.

It is this combination of heuristics and choosing the most appropriate heuristic that makes metaheuristic identification secure, private and low-cost. Privacy and security come because only the parties mutually identifying each other have the information about their connection. Strength of identification comes from the number of times there has been mutual identifications and from the network of many pairs.

An identity system built this way is robust as there is no single point of failure. There is no central collection of identity. Identity happens afresh each time a connection or re-connection is made. The system adapts incrementally as the identity characteristics of people, such as appearance, change over time.

All machine like information systems have some part centralised. All client-server systems are machine like. We can turn any machine information system into an intelligent system by making all the clients independent and equal to the server. Each entity in the network is both a client and a server. By distributing control and with it the data associated with actions we can turn all our information systems into intelligent adaptive life like systems.

REFERENCES

1. A. K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters .
2. J. Han, M. Kamber, Data Mining Concepts and Techniques, second edition Edition, Morgan Kaufmann Publishers, 2000.
3. K. Hammouda, M. Kamel, Incremental document clustering using cluster similarity histograms, in: Proceedings of IEEE/WIC International Conference on Web Intelligence (WI 2003), Vol. 1, IEEE, 2003, pp. 597–601.
4. K. Q. Weinberger, J. Blitzer, L. K. Saul, Distance metric learning for large margin nearest neighbor classification, Advances in Neural Information Processing Systems (NIPS) .
5. M. Ackerman, S. Ben-David, Measures of clustering quality: A working set of axioms for clustering, Advances in Neural Information Processing Systems(NIPS) .
6. M. Fillippone, F. Camastra, F. Masulli, S. Rovetta, A survey of kernel and spectral methods for clustering, Pattern Recognition 41 (2008) 176–190.
7. M. Kleinberg, J, An impossibility theorem for clustering, Advances in Neural Information Processing Systems(NIPS) (2002) 446–453.
8. N. Critianini, J. Shewe-Taylor, Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods, Cambridge University Press, Cambridge, U.K., 2000.
9. P. Lingras, C. West, Interval set clustering of web users with rough k-means, Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies 23 (1) (2004) 5–16.
10. P. Viswanath, V. S. Babu, Rough DBSCAN: A fast hybrid density based clustering method for large data sets, Pattern Recognition Letters 30 (2009) (2009) 1477–1488.
11. Paval Berkhin, Survey Of Clustering Data Mining Techniques, Technical Report, Accure Software, 2002.

12. R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Transactions on Neural Networks 16 (3) (2005) 645–678.
13. S. Asharaf, M. N. Murty, An adaptive rough fuzzy single pass algorithm for clustering large data sets, Pattern Recognition 36 (12) (2003) 3015–3018.
14. V. S. Babu, P. Viswanath, Rough-fuzzy weighted k-nearest leader classifier for large data sets, Pattern Recognition 42 (2009) 1719–1731.

CHAPTER 2

MULTI AGENT SYSTEM

INTRODUCTION

A multi-agent system (MAS) is a computerized system composed of multiple interacting intelligent agents Multi-agent systems can solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Intelligence may include methodic, functional, procedural approaches, algorithmic search or reinforcement learning.

Despite considerable overlap, a multi-agent system is not always the same as an agent-based model (ABM). The goal of an ABM is to search for explanatory insight into the collective behavior of agents (which don't necessarily need to be "intelligent") obeying simple rules, typically in natural systems, rather than in solving specific practical or engineering problems. The terminology of ABM tends

to be used more often in the science, and MAS in engineering and technology. Applications where multi-agent systems may deliver an appropriate approach include online trading, disaster response, target surveillance and social structure modelling.

2.1. INTRODUCTION TO MULTI-AGENT SYSTEMS

Multi-agent systems are a particular type of distributed intelligent systems in which autonomous agents inhabit a world with no global control or globally consistent knowledge. Figure 1 presents the so called multi-agent system equation, which states that in a multi-agent system a task is solved by agents that communicate among them.

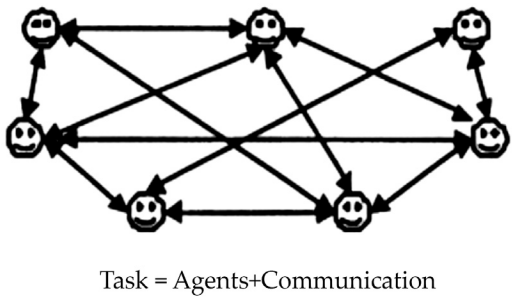


Figure 1. The multi-agent system equation.

We could view a multi-agent system as a society of individuals (agents) that interact by exchanging knowledge and by negotiating with each other in order to achieve either their own interest or some global goal. One of the characteristics of some MASs is the openness, which means that new agents can be created or can enter a MAS (i.e. mobile agents can arrive), and some unknown entities (e.g. legacy and elsewhere implemented entities) may enter a MAS. This characteristic has some technological implications: the need for standards (such as FIPA) and the existence of a proper infrastructure that support interoperations.

In a MAS, agents are embedded in a certain environment which could be dynamic, unpredictable and open. This environment

is the world of resources the agents perceive. The interactions between agents are the core of a multi-agent system functioning. Starting from, Nick Jennings has introduced the definition of a new computer level, the Social Level (SL) in order to solve the problems related to flexible social interactions. With a SL incorporated the Knowledge Level (KL), the prediction of the behavior of the social agents and of the whole MAS could be easily made. Following the Newell's notation, a preliminary version of the SL is given by: the system (an agent organization), the components (primitive elements from which the agent organization is built), composition laws (e.g. roles of agents in the organization), behavior laws, and the medium (the elements the system processes in order to obtain the desired behavior). The social level allows to create organizational models of multi-agent systems.

In a multi-agent system, agents are connected through different schemes, usually following mesh and hierachical structures.

The main characteristics of a multi-agent system are: autonomy (agents may be active and are responsible for their own activities), complexity (induced by the mechanisms of decision-making, learning, reasoning, etc), adaptability (adjust the agents activities to the dynamic environmental changes), concurrency (in case of tasks parallel processing), communication (inter-agent, intra-agent), distribution (MASs often operate on different hosts and are distributed over a network), mobility (agents need to migrate between platforms and environments), security and privacy (possible intrusion of the agents' data, state, or activities), and openness (MASs can dynamically decide upon their participants).

A multi-agent system has functional and non-functional properties. The functional properties are coordination, rationality, knowledge modelling. The non-functional properties are performance (response time, number of concurrent agents/task, computational time, communication overhead, etc), scalability (increased loading on an agent which is caused by its need to interact with more agents because the size of the society has increased), stability (a property of an equilibrium). Scalability is a property that becomes important when developing practical MASs. Most

agent systems that have been built so far involve a relatively small number of agents. When multi-agent systems are employed in larger applications this property need a very careful analysis. The scalability of a MAS is the average measure of the degree of performance degradation of individual agents in the society as their environmental loading increases, due to an expansion in the size of the society.

In a multi-agent system, agents have only local views, goals and knowledge that may conflict with others, and they can interact and work with other agents for the desired overall system's behavior. In order to achieve the common goals, a multi-agent system need to be coordinated. Coordination has to solve several problems such as the distributed expertise, resources or information, dependencies between agents' actions, efficiency. Two main dependencies can be encountered in a MAS, inter-agent dependencies and intra-agent dependencies. Several approaches that tackle the coordination problem were developed in Distributed Artificial Intelligence and Social Sciences, starting with different interaction protocol, partial global planning, and ending with the social laws. Depending on the domain of application specific coordination techniques are more appropriate. Also, the type of coordination protocol that is employed will influence the performance of the MAS.

A multi-agent infrastructure has to enable and rule interactions. It is the "middleware" layer that supports communication and coordination activities. The communication infrastructures (e.g. FIPA defined communication infrastructures) are dedicated to the control of the global interaction in a MAS. It includes routing messages and facilitators. The coordination infrastructures (e.g. MARS and Tucson coordination infrastructures) are dedicated to laws that establish which agents can execute which protocols and where. It includes synchronization and constraints on interactions.

The main benefits of multi-agent systems approaches are the following: address problems that are too large for a centralized single agent (e.g. because of resource limitations or for robustness concerns), allow the interconnection and interoperation of multiple existing legacy systems (e.g. expert systems, decision

support systems, legacy network protocols), improve scalability, provide solutions to inherently distributed problems (e.g. telecommunication control, workflow management), and provide solutions where the expertise is distributed. Some of the problems that could appear in a MAS are related to the emergent behavior, system robustness, and system reliability.

A major characteristic in agent research and applications is the high heterogeneity of the field. The heterogeneity means agent model heterogeneity (different models and formalisms for agents), language heterogeneity (different communication and interaction schemes used by agents), and application heterogeneity (various goals of a MAS for many application domains). The heterogeneity has to be manageable with appropriate models and software toolkits. Some models for agent architectures, communication, coordination, negotiation, learning in a multi-agent system, and, finally, we made a short presentation of the most known and used MAS development methodologies, and MAS development software.

2.1.1. Agent architectures

Two main complementary approaches are currently used to characterize intelligent (i.e. rational) agents and multi-agent systems: operational (agents and MASs are systems with particular features, i.e. a particular structure and a particular behavior), and based on system levels (agents and MASs are new system levels). The first approach define rational agents in terms of beliefs (information about the current world state), desires (preferences over future world states) and intentions (set of goals the agent is committed to achieve) - BDI, thus being independent from the internal agent architecture. The advantage is that uses the well founded logics (e.g. modal logics). One of the problems is related to the ground of rationality on axioms of a logic.

The second approach hides details in hardware design. System levels are levels of abstraction. The agent is modelled as being composed of a body (i.e. means for the agent to interact with

its environment), a set of actions the agent can perform on its environment, and a set of goals.

Figure 2 presents the general architecture of an agent.

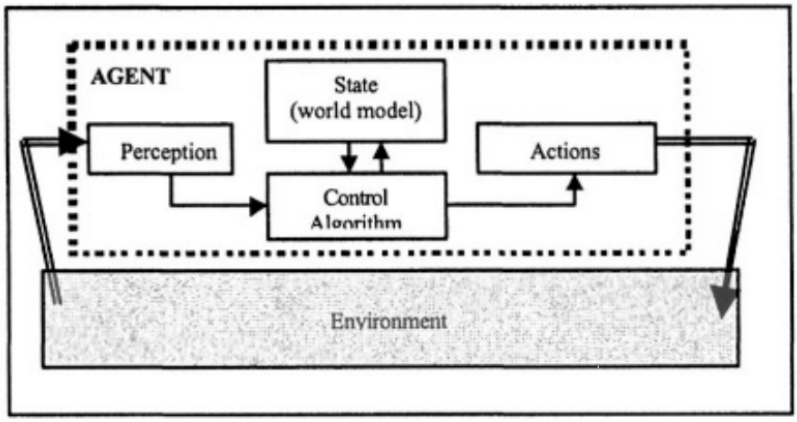


Figure 2. The general architecture of an agent.

The main agent architectures reported in the literature are the deliberative architecture, the reactive architecture, and the hybrid architecture. Most agent architectures are dedicated to the fulfillment of precise tasks or to problem solving, typically requiring reasoning and planning. Other approaches simulate emotions, which direct the agent in a more reactive way.

An agent that uses the deliberative architecture contains a symbolic world model, develops plans and makes decisions in the way proposed by symbolic artificial intelligence. Two important problems need to be solved in this case: the transduction problem and the representation/reasoning problem. The solution of the first problem led to work on vision, speech understanding, learning, etc. The solution for the second problem led to work on knowledge representation, automated reasoning/planning etc. The answer to the question "how should an agent decide what to do?" is that it should deduce its best action in light of its current goals and world model, so, it should plan. The world model can be constructed through learning.

An agent that uses the reactive architecture does not include any kind of central symbolic world model, and does not use complex symbolic reasoning. Rodney Brooks had introduced two ideas: situatedness and embodiment. In his view "intelligent" behavior arises as a result of an agent's interaction with its environment, and the agent is specified in terms of perceptions and actions. It is not a central dogma of the embodied approach that no internal representations exist. The being of the agent is dependent on the context in which it is encountered, and it is derived from purposeful interaction with the world. A possible answer to the question "how should an agent decide what to do?" is to do planning in advance and compile the result into a set of rapid reactions, or situation-action rules, which are then used for real-time decision making, or to learn a good set of reactions by trial and error.

An agent that uses a hybrid architecture has a layered architecture with both components: deliberative and reactive, usually, the reactive one having some kind of precedence over the deliberative one. Two important problems need to be solved: the management of the interactions between different layers and the development of the internal structure of an internally unknown system characterized by its I/O behavior. A possible answer to the question "how should an agent decide what to do?" is by integrating planning system, reactive system and learning system into a hybrid architecture, even included into a single algorithm, where each appears as a different facet or different use of that algorithm.

2.1.2. Communication

Interaction is a fundamental characteristic of a multi-agent system. The agent communication capability is needed in most cases for the coordination of activities. A conversation is a pattern of message exchange that two or more agents agree to follow in communicating with one another. Actually, it is a pre-established coordination protocol. Several methods could be used for the representation of conversations: state transition diagrams, finite-state machines, Petri nets, etc. An Agent Communication Language (ACL) is a collection

of speech-act-like message types, with agreed-upon semantics, which facilitates the knowledge and information exchange between software agents. The standardization efforts made so far generated a standard framework for agent communication (a standard agent communication language – KQML, FIPA ACL – both based on the notion of speech act). Current used ACLs are not accepted by all researchers due to some problems they have: the formal semantics for such languages (define semantics based on mental states, or equate the meaning of a speech act with the set of allowable responses), the relationships between speech acts and various related entities such as conversations, agent mental state. A possible alternative model of agent communication is Albatross (Agent language based on a treatment of social semantics) that has a commitment-based semantics for speech acts.

KQML (Knowledge Query and Manipulation Language) is a high level message-oriented communication language and protocol for information exchange independent of content syntax and applicable ontology. It is independent of the transport mechanism (TCP/IP, SMTP, etc), independent of the content language (KIF, SQL, Prolog, etc), and independent of the ontology assumed by the content. A KQML message has three layers: content, communication, and message. The syntax of KQML is based on the s-expression used in Lisp (per formative arguments). The semantics of KQML is provided in terms of pre-conditions, post-conditions, and completion conditions for each per formative. FIPA ACL is similar to KQML. The communication primitives are called communicative acts (CA). SL is the formal language used to define the semantics of FIPA ACL. It is a multi-modal logic with modal BDI operators, and can represent propositions, objects, and actions. In FIPA ACL, the semantics of each CA are specified as sets of SL formulae that describes the act's feasibility pre-conditions and its rational effect.

A message has three aspects: locution (how is the message phrased), illocution (how is the message meant by the sender or understood by the receiver), and per locution (how does the message influence the receiver's behavior). Figure 3 shows an example of ACL message.

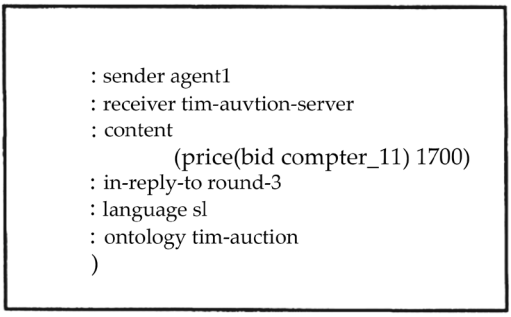


Figure 3. Example of an ACL message.

One important issue in agent communication is the understanding of messages meaning. The message ontology gives the meaning of a message. The ontology provides interpretation of the message, giving a meaning for each word included in the content of the message. More generally, ontology is a description of the concepts and relationships that can exist for an agent. Usually, an ontology is designed for a specific multi-agent system, thus being application domain dependent. Therefore, one of the problems that may occur is the communication among agents that use different ontologies.

Agent communication languages such as KQML and FIPA ACL have provided a tool and framework to tackle the interoperability problems of inter-agent communication.

2.1.3. Coordination

In a multi-agent system agents have their own plans, intentions and knowledge, and are willing to solve their local goals, while for the global goal of the system it is needed a coordination mechanism to solve the conflicts that may arise due to limited resources or to the opposite intentions the agents might have. Coordination is a process in which the agents engage in order to ensure that the multi-agent system acts in a coherent manner. For this purpose, the agents must share information about their activities. One way the agents may achieve coordination is by communication. Another way, without communication, assume that the agents

have models of each other's' behaviors. Coordination avoids unnecessary activity and allows a reduce resource contention, avoids deadlock, and lovelock and maintains safety conditions (minimizing the conflicts). Deadlock refers to a state of affairs in which further action between two or more agents is impossible. Lovelock refers to a scenario where agents continuously act (exchange tasks, for example), but no progress is made.

Comprehensive coordination technique must have four major components: (1) a set of structures that enable the agents' interaction in predictable ways; (2) flexibility in order to allow the agents to operate in dynamic environment and to cope with their inherently partial and imprecise viewpoint of the community; (3) a set of social structures which describe how agents should behave towards one another when they are engaged in the coordination process; (4) sufficient knowledge and reasoning capabilities must be incorporated in order to exploit both the available structure (individual and social) and the flexibility. In a coordination model is described by three elements: the coordinates, i.e. the objects of the coordination (e.g. the software agents), the coordination media, i.e. what enables the interaction between the coordinables (e.g. the agent communication language), and the coordination laws that govern the interaction between the coordination media and the coordinables, and the rules that the coordination media employs (e.g. the finite state machine that describes the interaction protocol).

Coordination may require cooperation between agents, but sometimes, coordination may occur without cooperation. The design of a specific coordination mechanism will take into account appart from the domain of the application, the type of architecture that is adopted for the MAS design. There are mediated interaction coordination models (e.g. blackboard based interaction), and non-mediated interaction ones. When a mediated interaction coordination protocol is applied, the state of the interaction could be inspected in order to check the coordination trace.

In the literature there have been reported different coordination techniques. In it is made a classification of the existing coordination

techniques applied to software agent systems. Four broad categories were identified: organizational structuring, contracting, multi-agent planning, and negotiation. The first category includes coordination techniques like the classical client-server or master-slave techniques. A high level coordination strategy from the second category is given by the Contract Net Protocol (CNP). A multi-agent planning technique, from the third category, involves the construction of a multi-agent plan that details all the agents' future actions and interactions required to achieve their goals, and interleave execution with more planning and re-planning. The fourth category of coordination techniques uses negotiation to solve the conflicts that may arise in a MAS. Usually, negotiation compromises speed for quality because there is an overhead during negotiation before compromise is made.

In it is made a comparison between three types of coordination models: hybrid coordination models based on tuple centers, interaction protocols as a coordination mechanism, and implicit coordination through the semantic of classic ACLs. All these models were proposed by different research communities. The coordination community proposed the first one, while the second one was proposed by the agent community, and the last one by the more formally inclined part of the agent community.

In it is presented a framework that enables agents to dynamically select the mechanism they employ in order to coordinate their inter-related activities. The classification made in reveals two extreme classes, the social laws (long-term rules that prescribe how to behave in a given society), and the interaction protocols (e.g. CNP, that coordinates the short-term activities of agents in order to accomplish a specific task). Between these classes it is situated partial global planning. In another approach is described, the use of coordination evaluation signals to guide an agent's behavior. In it is presented a precise conceptual model of coordination as structured "conversations" involving communicative actions amongst agents. The model was extended with the COOL (CO Ordination Language) language and it was applied in several industrial MAS. Coordination gradually emerges among agents

and their social context. The agents are embedded in a social context in which a set of norms is in force. These norms influence the agents' decision-making and goal generation processes by modifying their context.

In cases where the communication costs are prohibitive, a solution is to coordinate a set of individually motivated agents by choosing an equilibrium point. Such an evolutionary approach is used in where by learning from observations the coordination point (viewed as an equilibrium point) is reached.

Another coordination model is given by stigmergy, which means that agents put signs, called sigma in Greek, in their environment to mutually influence each other's behavior. Such indirect coordination mechanism is suitable for small-grained interactions compared to coordination methods that usually require an explicit interaction between the agents. With stigmergy, agents observe signs in their environment and act upon them without needing any synchronization with other agents. The signs are typically multi-dimensional and reflect relevant aspects of the coordination task.

2.1.4. Negotiation

Negotiation is a discussion in which interested parties exchange information and, eventually, come to an agreement. In a multi-agent system, negotiation has been viewed as a solution for problems such as network coherency, the problem decomposition and allocation, and more generally for the coordination problems. Negotiation can be viewed as a process whereby agents communicate to reach a common decision. Thus, the negotiation process involves the identification of interactions (through communication) and the modification of requirements through proposals and counter-proposals. The main steps of negotiation are (1) exchange of information; (2) each party evaluates information from its own perspective; (3) final agreement is reached by mutual selection.

2.1.5. Learning

In a multi-agent system the agents are embedded in the environment where they live, and need to interact with other agents in order to achieve their goals. Usually, they try to adapt to the environment by learning or by an evolutionary process, thus doing an anticipation of the interaction with the other agents. Learning in a multi-agent environment is complicated by the fact that, as other agents learn, the environment effectively changes. When agents are acting and learning simultaneously, their decisions affect and limit what they subsequently learn.

Adaptability and embodiment are two important issues that need to be addressed when designing flexible multi-agent systems. Adaptability allows the generation of a model of the selection process within the system and thus results in internal representations that can indicate future successful interactions. Agents can be seen as having a "body" that is embedded in their work environment, and is adapted to this environment by learning or by an evolutionary process. In the context of a multi-agent system, the two properties, adaptability and embodiment, are tightly related to each other.

The most used learning algorithms that were experimented in the case of multi-agent systems are reinforcement learning (e.g. Q-learning), Bayesian learning, and model-based learning

2.1.6. Methodologies for multi-agent system development

The development of multi-agent systems applications has generated the development of an agent-specific software engineering, called Agent Oriented Software Engineering (AOSE), which defines abstractions (of agents, environment, interaction protocols, context), specific methodologies and tools, and could be applicable to a very wide range of distributed computing applications. The adoption of object-oriented (OO) methodologies from object-oriented software engineering is an option, but some mismatches could appeared, as each methodology may introduce

new abstractions (e.g. roles, organization, responsibility, belief, desire, and intentions).

Usually, the whole life-cycle of system development (analysis, design, implementation, validation) is covered by a methodology. Let's consider the analysis and design steps. During the analysis step, agents are associated with the entities of the analyzed scenarios. Then, roles, responsibilities and capabilities are associated accordingly. Finally, interaction patterns between agents are identified. At the knowledge level, for each agent we need to identify its beliefs, goals, body (i.e. the way it interacts with the environment), and actions. The environment behavior should be identified. At the social level, the analysis step focus on the analysis of an organization and it is needed to identify the roles in the organization, the organizational relations between roles, the dependency between roles, the interaction channels, the obligations, and the influence mechanisms. At the agent design step, we associate agents with the components used to build the system.

There are two approaches for tackle the Agent-Oriented Methodologies (AOM), the Knowledge Engineering (KE) approach and the Software Engineering (SE) approach. The KE approach provides techniques for agent's knowledge modelling. Example of such tools are DESIRE and MASCommonKADS. The SE approach used the OO approach, in which an agent is viewed as an active object. Examples of such tools are AUML, GAIA, ADEPT, MESSAGE/UML, OPM/MAS. AUML is an extension of the standard SE approach, UML (Unified Modeling Language). The FIPA Agent UML (AUML) standard is under development. MASs are sometimes characterized as extensions of object-oriented systems. This overlay simplified view has often troubled system designers as they try to capture the unique features of MASs using OO tools. Therefore, an agent based unified modeling language (AUML) is being developed. The ZEUS toolkit was developed in parallel with an agent development methodology, which is supported by the ZEUS Agent Generator tool. DESIRE provides formal specifications to automatically generate a prototype. It

has a design approach more than an analysis approach. Agents are viewed as composed components, and MAS interaction as components interaction. OPM/MAS offers an approach that combines OO and process orientation. GAIA is a top-down design methodology that has a solid social foundation and is an extension of the SE approach.

2.1.7. Multi-Agent System Development Software

Agent platforms support effective design and construction of agents and multi-agent systems. An agent platform has to provide the following functions: agent management, agent mobility, agent communication, directory services (yellow pages), and interface to plug-in additional services. Figure 4 presents the architecture of a FIPA Agent Platform.

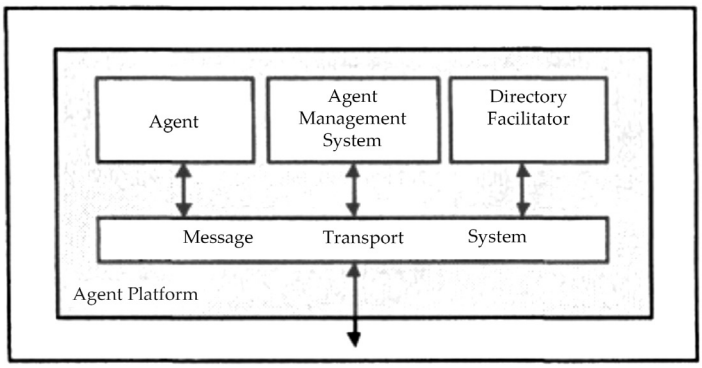


Figure 4. The architecture of a FIPA Agent Platform.

JADE (Java Agent Development framework) is a free software framework for the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. JADE is written in Java language and is made by various Java packages, giving application programmers both ready-made pieces of functionality and abstract interfaces for custom, application dependent tasks. The main tools provided by JADE are the Remote Management Agent (RMA), the Dummy Agent, the Sniffer agent, the Introspector Agent, the Socket Proxy

Agent, theDF GUI (a complete graphical user interface that is used by the default Directory Facilitator). The latest available version is JADE 3.1 (Dec. 2003).

ZEUS is a generic, customizable, and saleable industrial-strength collaborative agent toolkit. It consists in a package of classes implemented in Java, allowing it to run on a variety of hardware platforms. The classes of the ZEUS toolkit are classified in three groups: an agent component library, an agent building tool, and an agent visualization tool. ZEUS toolkit covers all stages of a MAS development, from analysis to deployment. It is limited to a single agent model.

Agent Builder is an integrated software toolkit that allows software developers to quickly implement intelligent agents and agent-based applications. The latest version is Agent Builder 1.3 (based on Java 1.3), Windows XP compatible. Two versions are currently available: LITE, ideal for building single-agent, stand-alone applications and small agencies, and PRO that has all the features of LITE plus an advanced suite of tools for testing and building multi-agent systems. Agent Builder is grounded on Agent0/Placa BDI architecture. It is limited to a single agent model. Almost all stages of a MAS development are covered.

The MadKit toolkit provides a generic, highly customizable and scalable agent platform. It is a generic multi-agent platform based on an organizational model called AGR (agent-group-role). Mad Kit is composed by a set of Java classes that implements the agent kernel, and various libraries of messages, probes and agents. Also, it includes a graphical development environment, system and demonstration agents. The MadKit micro-kernel is a small and optimized agent kernel which handles several tasks (control of local groups and roles, agent life-cycle management, and local message passing). The kernel is extensible through "kernel hooks". MadKit has a good versatility and light methodology (no BDI).

Volcano is a multi-agent platform that is under development, and whose aims are to fulfill the criteria of completeness (e.g. inclusion of MAS analysis and design phases), applicability (e.g. the versatility

of the platform) and complexity (e.g. friendlier user interface, reuse of the platform). It is based on the Agents Environment Interactions Organizations (AEIO) MAS decomposition. In this framework, agents are internal architectures of the processing entities, the environment is composed by the domain dependent elements for structuring external interactions between entities, the interactions are elements for structuring internal interactions between entities, and organizations are elements for structuring sets of entities within the MAS. Volcano has a full analysis-to-deployment chain, including an open library of models and intelligent deployment tools.

2.2. CLASSIFICATION OF MULTI AGENT SYSTEM

The classification of MAS is a difficult task as it can be done based on several different attributes such as Architecture Learning Communication, Coordination. A general classification encompassing most of these features is shown in figure 5.

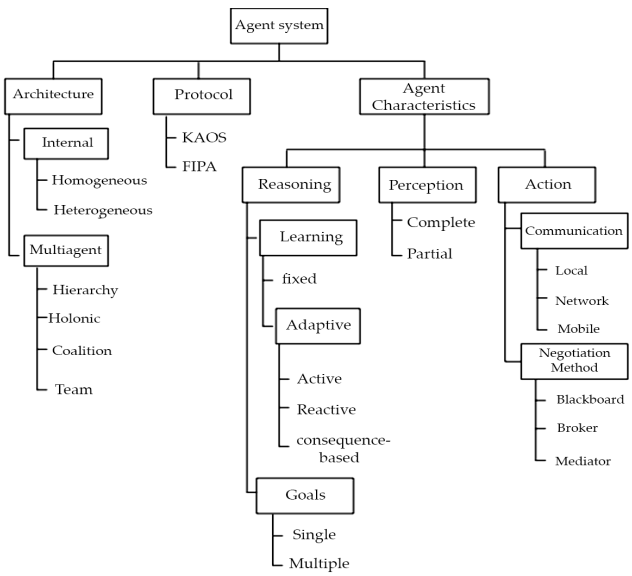


Figure 5. Classification of a multi agent system based on the use of different attributes.

2.2.1. Internal Architecture

Based on the internal architecture of the particular individual agents forming the multi-agent system, it may be classified as two types:

1. Homogeneous structure
2. Heterogeneous structure

a) **Homogeneous Structure:** In a homogeneous architecture, all agents forming the multi-agent system have the same internal architecture. Internal architecture refers to the Local Goals, Sensor Capabilities, Internal states, Inference Mechanism and Possible Actions. The difference between the agents is its physical location and the part of the environment where the action is done. Each agent receives an input from different parts of the environment. There may be overlap in the sensor inputs received. In a typical distributed environment, overlap of sensory inputs is rarely present.

b) **Heterogeneous Structure:** In a heterogeneous architecture, the agents may differ in ability, structure and functionality. Based on the dynamics of the environment and the location of the particular agent, the actions chosen by agent might differ from the agent located in a different part but it will have the same functionality. Heterogeneous architecture helps to make modelling applications much closer to real-world. Each agent can have different local goals that may contradict the objective of other agents. A typical example of this can be seen in the Predator-Prey game. Here both the prey and the predator can be modelled as agents. The objectives of the two agents are likely to be in direct contradiction one to the other.

2.2.2. Overall Agent Organization

Hierarchical Organization

Hierarchical Organization is one of the earliest organizational design in multiagent systems. Hierarchical architecture has been applied to a large number of distributed problems. In the hierarchical agent architecture, the agents are arranged in a typical tree like structure. The agents at different levels on the tree structure have different levels of autonomy. The data from the lower levels of hierarchy typically flow upwards to agents with a higher hierarchy. The Control Signal or Supervisory Signals flow from higher to a lower hierarchy. Figure 6 shows a typical Three Hierarchical Multi-Agent Architecture. The flow of control signals is from a higher to lower priority agents.

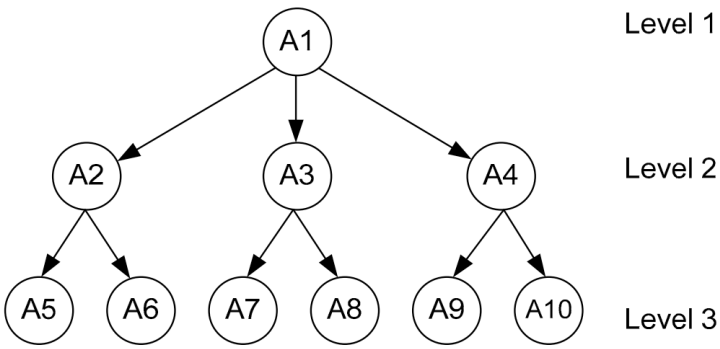


Figure 6. A Hierarchical Agent Architecture.

According to the distribution of the control between the agents, hierarchical architecture can be further classified as being a simple and uniform hierarchy.

Simple Hierarchy: In a simple hierarchy, the decision making authority is bestowed using a single agent of highest level of the hierarchy. The problem with a simple hierarchy is that a single point failure of the agent in the highest hierarchy may cause the entire system to fail.

Uniform Hierarchy: In a uniform hierarchy, the authority is distributed among the various agents in order to increase the efficiency, fault tolerance having a graceful degradation in case of single and multi-point failures. Decisions are made by agents having the appropriate information. These decisions are sent up the hierarchy only where there is a conflict of interest between agents in different hierarchy.

A uniform hierarchical multi-agent system applied to a urban traffic signal control problem. The objective is to provide a distributed control of traffic signal timings. This is to reduce the total delay time experienced by vehicles in a road network. A Three Level Hierarchical Multi-Agent System where each intersection is modelled as an agent forming the agents at lowest hierarchy followed by zonal agents which supervise a group of lower level agents and finally a single apex supervisor agent at the top of hierarchy. The agent in the lower level of the hierarchy decides on the optimal green time. This is based on the local information collected at each of the intersections. The agents at the higher level of the hierarchy modify decision of the lower hierarchical agents. From time to time there may be a conflict of interest or the overall delay experienced at a group of intersections increases due to a selected action. Here, the overall control is uniformly distributed among the agents. A disadvantage is that the amount and the type of information which must be transmitted to agents at higher hierarchy. This is a nontrivial problem which increases as the network size increases.

Holonic Agent Organization

A 'Holon' is a stable and coherent similar or fractal structure that consists of several 'holons' as its sub-structure and is itself a part of a larger framework. The concept of a holon was proposed by Arthur Koestler to explain the social behavior of biological species. However, the hierarchical structure of the holon and its interactions have been used to model a large organizational behaviors in manufacturing and business domains.

In a holonic multi-agent system, an agent that appears as a single entity to may be composed of many sub-agents bound together by commitments. The sub-agents are not bound by a hard constraints or by pre-defined rule but through commitments. These refer to the relationships agreed to by all of the participating agents inside the holon.

Each holon appoints or selects a Head Agent that can communicate with the environment or with other agents located in the environment. The selection of the head agent is usually based on the resource availability, communication capability and the internal architecture of each agent. In a homogeneous multi-agent system, the selection can be random and a rotation policy could be employed similar to that used with distributed wireless sensor networks. In the heterogeneous architecture, the selection is based on the capability of the agent. The holons formed may group further in accordance to benefits foreseen in forming a coherent structure. They form Superholons. Figure 7. Shows a Superholon formed by grouping two holons. Agents A1 and A4 are the heads of the holons and communicate with agent A7. This is the head of the superholon. The architecture appears to be similar to that of hierarchical organization. However in holonic architecture, cross tree interactions and overlapping or agents forming part of two different holons are allowed.

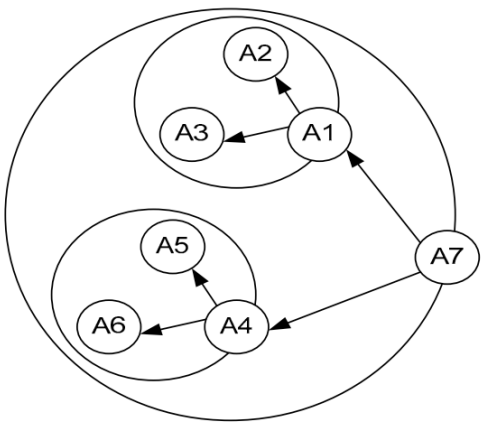


Figure 7. An example of Superholon with Nested Holons resembling the Hierarchical MAS.

In recent times had proved the superiority of the holonic multi-agent organization and how the autonomy of the agents increases when in a holonic group. The abstraction of the internal working of holons provides an increased degree of freedom when selecting the behaviour. A major disadvantage is the lack of a model or of a knowledge of the internal architecture of the holons. This makes it difficult for other agents to predict the resulting actions of the holons.

Coalitions

In coalition architecture, a group of agents come together for a short time to increase the utility or performance of the individual agents in a group. The coalition ceases to exist when the performance goal is achieved. Figure 8. Shows a typical coalition multiagent system. The agents forming the coalition may have either a flat or a hierarchical architecture. Even when using a flat architecture, it is possible to have a leading agent to act as a representative of the coalition group. The overlap of agents among coalition groups is allowed as this increases the common knowledge within the coalition group. It helps to build a belief model. However the use of overlap increases the complexity of computation of the negotiation strategy. Coalition is difficult to maintain in a dynamic environment due to the shift in the performance of group. It may be necessary to regroup agents in order to maximize system performance.

Theoretically, forming a single group consisting of all the agents in the environment will maximize the performance of the system. This is because each agent has access to all of the information and resources necessary to calculate the condition for optimal action. It is impractical to form such a coalition due to restraints on the communication and resources.

The number of coalition groups created must be minimized in order to reduce the cost associated with creating and dissolving a coalition group. The group formation may be pre-defined based on a threshold set for performance measure or alternatively could be evolved online.

A coalition multi-agent architecture for urban traffic signal control was mentioned. Each intersection was modelled as an agent with capability to decide the optimal green time required for that intersection. A distributed neuro-fuzzy inference engine was used to compute the level of cooperation required and the agents which must be grouped together.

The coalition groups reorganize and regroup dynamically with respect to the changing traffic input pattern. The disadvantage is the increased computational complexity involved in creating ensembles or coalition groups. The coalition MAS may have a better short term performance than the other agent architectures.

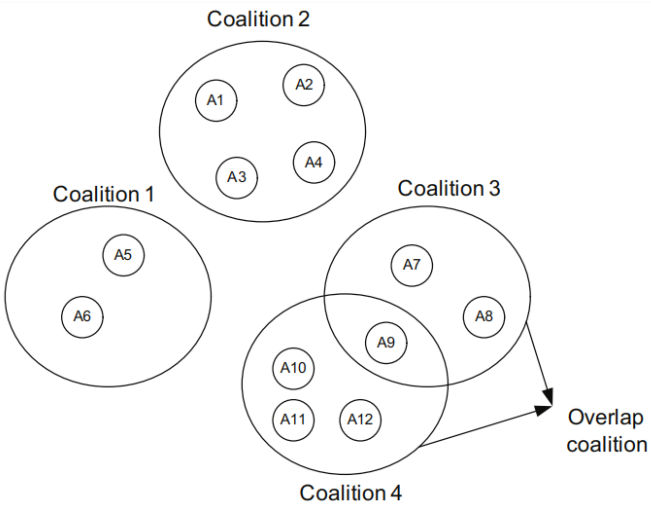


Figure 8. Coalition multi-agent architecture using overlapping groups.

Teams

Team MAS architecture is similar to coalition architecture in design except that the agents in a team work together to increase the overall performance of the group. Rather than each working as individual agents. The interactions of the agents within a team can be quite arbitrary and the goals or the roles assigned to each of the agents can vary with time based on improvements resulting from the team performance. Deals with a team based multi-agent

architecture having a partially observable environment. In other words, teams that cannot communicate with each other has been proposed for the Arthur's bar problem. Each team decides on whether to attend a bar by means of predictions based on the behavioral pattern and the crowd level experienced which is the reward or the utility received associated with the specific period of time. Based on the observations made in, it can be concluded that a large team size is not beneficial under all conditions. Consequently some compromise must be made between the amount of information, number of agents in the team and the learning capabilities of the agents.

Large teams offer a better visibility of the environment and larger amount of relevant information. However, learning or incorporating the experiences of individual agents into a single framework team is affected. A smaller team size offers faster learning possibilities but result in sub-optimal performance due to a limited view of the environment. Tradeoffs between learning and performance need to be made in the selection of the optimal team size. This increases the computational cost much greater than that experienced in coalition multi-agent system architecture. Figure 9. Shows a typical team based on architecture with partial view. The team 1 and 3 can each other but not teams 2, 4 and vice versa. The internal behavior of the agents and their roles are arbitrary and vary with teams even in homogeneous agent structure.

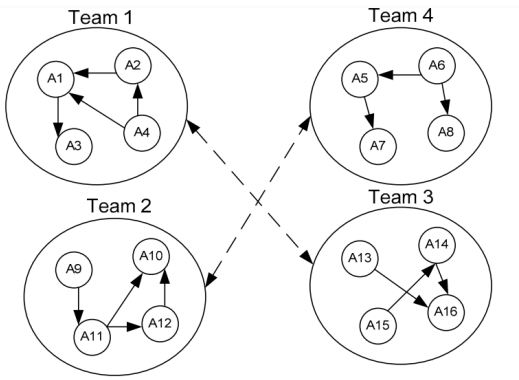


Figure 9. Team based multi-agent architecture with a partial view of the other teams.

Variations and constraints on aspects of the four agent architecture mentioned before can produce other architectures such as federations, societies and congregations. Most of these architectures are inspired by behavioral patterns in governments, institutions and large industrial organizations.

2.3. COMMUNICATION IN MULTI-AGENT SYSTEM

Communication is one of the crucial components in multi-agent systems that needs careful consideration. Unnecessary or redundant intra-agent communication can increase the cost and cause instability. Communication in a multi-agent system can be classified as two types. This is based on the architecture of the agent system and the type of information which is to be communicated between the agents. In the various issues arising in MAS system with homogeneous and heterogeneous architecture has been considered and explained by using a predator/prey and by the use of robotic soccer games. Based on the information communication between the agents MAS can classified as local communication or message passing and network communication or Blackboard. Mobile communication can be categorized into class of local communication.

2.3.1. Local Communication

Local communication has no place to store the information and there is no intermediate communication media present to act as a facilitator. The term message passing is used to emphasize the direct communication between the agents. Figure 10. Shows the structure of the message passing communication between agents. In this type of communication, the information flow is bidirectional. It creates a distributed architecture and it reduces the bottleneck caused by failure of central agents.

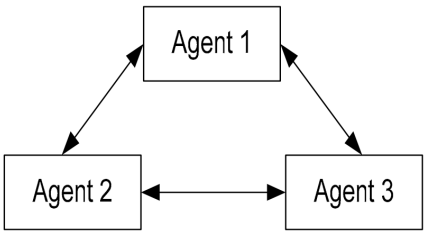


Figure 10. Message Passing Communication between agents.

2.3.2. Blackboards

Another way of exchanging information between agents is through Blackboards. Agent-based blackboards, like federation systems, use grouping to manage the interactions between agents. There are significant differences between the federation agent architecture and the blackboard communication.

In blackboard communication, a group of agents share a data repository which is provided for efficient storage and retrieval of data actively shared between the agents. The repository can hold both the design data as well as the control knowledge that can be accessed by the agents. The type of data that can be accessed by an agent can be controlled through the use of a control shell. This acts as a network interface that notifies the agent when relevant data is available in the repository. The control shell can be programmed to establish different types of coordination among the agents. Neither the agent groups nor the individual agents in the group need to be physically located near the blackboards. It is possible to establish communication between various groups by remote interface communication. The major issue is due to the failure of blackboards. This could render the group of agents useless depending on the specific blackboard. However, it is possible to establish some redundancy and share resources between various blackboards. Figure 11a shows a single blackboard with the group of agents associated with it. Figure 11b shows blackboard communication between two different agent groups and also the facilitator agents present in each group.

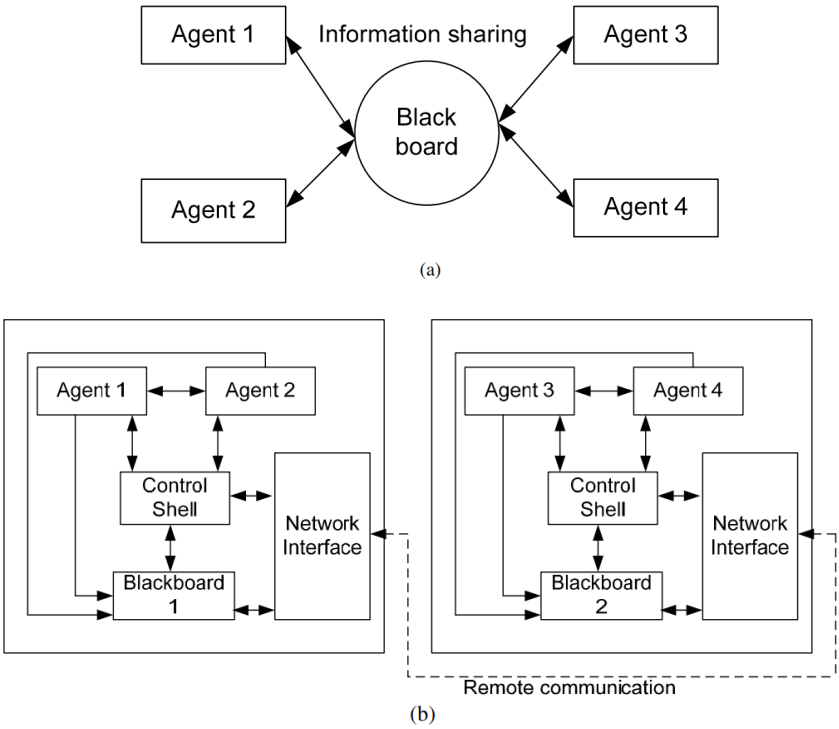


Figure 11. (a) Blackboard type communication between agents. (b) Blackboard communication using remote communication between agent groups.

2.3.3. Agent Communication Language

An increase in the number of agents and the heterogeneity of the group necessitates a common framework to help in proper interaction and information sharing. This common framework is provided by the agent communication languages (ACL). The elements that are of prime importance in the design of ACL were highlighted in. They are the availability of the following.

- A common language and interaction format (Protocol) that can be understood by all of the participating agents.

- A shared Ontology where the message communicated has the same meaning in all contexts and follows agent independent semantics.

There are two popular approaches in the design of an agent communication language. They are Procedural approach and Declarative approach. In Procedural approach, the communication between the agents is modelled as a sharing of the procedural directives. Procedural directives shared could be a part of how the specific agents does a specific task or the entire working of the agent itself. Scripting languages are commonly used in the procedural approach. Some of the most common scripting languages employed are JAVA, TCL, Apple script and Tele script. The major disadvantage of the procedural approach is the necessity of providing information on the recipient agent which in most cases is not known or only partially known. In case of making a wrong model assumption, the procedural approach may have a destructive effect on the performance of the agents. The second major concern is the merging of shared procedural scripts into a single large executable relevant script for the agent. Owing to these disadvantages, the procedural approach is not the preferred method for designing agent communication language.

In the declarative approach, the agent communication language is designed and based on the sharing of the declarative statements that specifies definitions, assumptions, assertions, axioms etc. For the proper design of an ACL using a declarative approach, the declarative statements must be sufficiently expressive to encompass the use of a wide-variety of information. This would increase the scope of the agent system and also avoid the necessity of using specialized methods to pass certain functions. The declarative statements must be short and precise as to increase in the length affects the cost of communication and also the probability of information corruption. The declarative statements also needs to be simple enough to avoid the use of a high level language. This means that the use of the language is not required to interpret the message passed. To meet all of the requirements of the declarative approach based ACL, the ARPA knowledge

sharing effort has devised an agent communication language to satisfy all requirements.

The ACL designed consists of three parts: A Vocabulary part, "Inner language" and "Outer language". The Inner language is responsible for the translation of the communication information into a logical form that is understood by all agents. There is still no consensus on a single language and many inner language representations like KIF (Knowledge Interchange Format), KRSL, LOOM are available. The linguistic representation created by these inner languages are concise, unambiguous and context-dependent. The receivers must derive from them the original logical form. For each linguistic representation, ACL maintains a large vocabulary repository. A good ACL maintains this repository open-ended so that modifications and additions can be made to include increased functionality. The repository must also maintain multiple ontology's and its usage will depends on the application domain.

Knowledge Interchange Format is one of the best known inner languages and it is an extension of the First-Order Predicate Calculus (FOPC). Some of the information that can be encoded using KIF are simple data, constraints, negations, disjunctions, rules, meta-level information that aids in the final decision process. It is not possible to use just the KIF for information exchange as much implicit information needs to be embedded. This is so that the receiving agent can interpret it with a minimal knowledge of the sender's structure. This is difficult to achieve as the packet size grows with the increase in embedded information. To overcome this bottleneck, a high level language that utilizes the inner language as its backbone were introduced. These high-level languages make the information exchange independent of the content syntax and ontology. One well known Outer language that satisfies this category is the KQML (Knowledge Query and Manipulation Language). A typical information exchange between two agents utilizing the KQML and KIF agent communication language is as follows.

```
(ask :Content (geolocation lax(?long ?lat))
            : language KIF
            :ontology STD_GEO
            : from location_agent
            : to location_server
: label Query- "Query identifier")
             (tell : content "geolocation(lax)"
            : language standard_prolog
            : ontology STD_GEO)
```

The KQML is conceived as both message format and message handling protocol to facilitate smooth communication between agents. From the example provided, it can be seen that KQML consists of three layers (Figure 12): A communication layer which indicates the origin and destination agent information and query label or identifier, a message layer that specifies the function to be performed (eg: In the example provided, the first agent asks for the geographic location and the second agent replies to the query), and a content layer to provide the necessary details to perform the specific query.

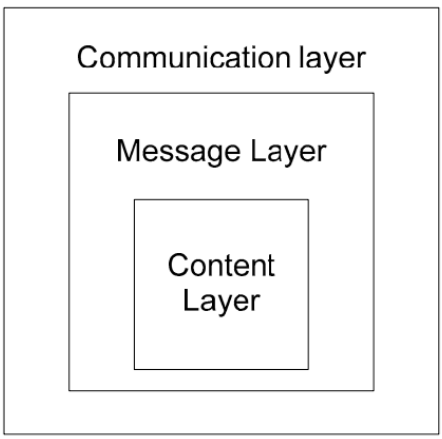


Figure 12. KQML - Layered language structure.

In KQML, the communication layer is at a low level and packet oriented. A stream oriented approach is yet to be developed. The communication streams could be built on TCP/IP, RDP, UDP

or any other packet communication media. The content layer specifies the language to be employed by the agent. It should be noted that agents can use different languages to communicate with each other and interpretation can be performed locally by higher level languages.

2.4. COORDINATION IN MULTI-AGENT SYSTEM

Coordination is the central issue in the design of multi-agent systems. Agents are seldom stand-alone systems and usually involve more than one agent working in parallel to achieve a common goal. When multiple agents are employed to achieve a goal, there is a necessity to coordinate or synchronize the actions to ensure the stability of the system. Coordination between agents increases the chances of attaining a optimal global solution. In major reasons necessitating coordination between the agents were highlighted. The requirements are

- To prevent chaos and anarchy
- To meet global constraints
- To utilize distributed resources, expertise and information
- To prevent conflicts between agents
- To improve the overall efficiency of the system

Coordination can be achieved by applying constraints on the joint action choices of each agent or by utilizing the information collated from neighbouring agents. These are used to compute the equilibrium action point that could effectively enhance the utility of all the participating agents. Applying constraints on the joint actions requires an extensive knowledge of the application domain. This may not be readily available. It necessitates the selection of the proper action taken by each agent. It is based on the equilibrium action computed. However, the payoff matrix necessary to compute the utility value of all action choices might be difficult to determine. The dimension of the payoff matrix grows exponentially with the increasing the number of agents and

the available action choices. This may create a bottleneck when computing the optimal solution.

The problem of this dimensional explosion can be solved by dividing the game into a number of sub-games that can be more effectively solved. A simple mechanism which can reduce the number of action choices is to apply constraints or assign roles to each agent. Once a specific role is assigned, the number of permitted action choices is reduced and are made more computationally feasible. This approach is of particular importance in a distributed coordination mechanism. However, in centralized coordination techniques this is not a major concern as it is possible to construct belief models for all agents. The payoff matrix can be computed centrally and provided to all of the agents as shared resource. The centralized coordination is adopted from the basic client/server model of coordination. Most of the centralized coordination techniques uses blackboards as a way in which to exchange information. Master agent schedules of all the connected agents are required to read and write information from and to the central information repository. Some of the commonly adopted client/server models are KASBAH and MAGMA. The model uses a global blackboard to achieve the required coordination. Disadvantage in using the centralized coordination is that of system disintegration resulting from a single point failure of the repository or of the mediating agent. Further use of the centralized coordination technique is contradictory to the basic assumption of DAI.

2.4.1. Coordination through Protocol

A classic coordination technique among agents in a distributed architecture is through the communication protocol. Protocol is usually in high level language which specifies the method of coordination between the agents and is a series of task and resource allocation methods. The most widely used protocol is the Contract Net Protocol which facilitates the use of distributed control of cooperative task execution. The protocol specifies what information is to be communicated between the agents and the format of the information of dissemination is handled by the

protocol. A low-level communication language such as KIF that can handle the communication streams is assumed to be available. The protocol engages in negotiation between the agents to arrive at an appropriate solution. The negotiation process must adhere to the following characteristics.

1. Negotiation is a local process between agents and it involves no central control
2. Two way communication is available between all participating agents exists
3. Each agent makes its evaluation based on its own perception of the environment
4. The final agreement is made through a mutual selection of the action plan

Each agent assumes the role of Manager and Contractor as necessary. The manager essentially serves to break a larger problem into smaller sub-problems and finds contractors that can perform these functions effectively. A contractor can become a manager and decompose the sub-problem so as to reduce the computational cost and increase efficiency. The manager contracts with a contractor through a process of bidding. In the bidding process, the manager specifies the type of resource required and a description of the problem to be solved. Agents that are free or idle and have the resources required to perform the operation submits a bid indicating their capabilities. The manager agent then evaluates the received bids, chooses an appropriate contractor agent and awards the contract. In case of non-availability of any suitable contracting agent, the manager agent waits for a pre-specified period before rebroadcasting the contract to all agents. The contracting agent may negotiate with the manager agent seeking an access to a particular resource as a condition before accepting the contract.

The FIPA model is the best example of an agent platform that utilizes the contract net protocol to achieve coordination in between the agents. FIPA - Foundation for Intelligent Physical Agents is a model developed to standardize agent technology. The FIPA has

its own ACL (Agent Communication Language) that serves as the backbone for the high-level contract net protocol.

Disadvantage of the protocol based coordination is the assumption of the existence of an cooperative agent. The negotiation strategy is passive and does not involve any punitive measures which attempts to force an agent to adopt a specific strategy. Usually a common strategy is achieved through iterative communications where the negotiation parameters are modified progressively to achieve equilibrium. This makes the contract net protocol to be communication intensive.

2.4.2. Coordination via Graphs

Coordination graphs were introduced in to serve as a framework to solve large scale distributed coordination problems. In coordination graphs, each problem is subdivided into smaller problems that are easier to solve. The main assumption with coordination graphs is that the payoffs can be expressed as a linear combination of the local payoffs of the sub-game. Based on this assumption, algorithm such as variable elimination method can compute the optimal joint actions by iteratively eliminating agents and creating new conditional functions that compute the maximal value the agent can achieve given the action of other agents on which it depends. The joint action choice is only known after the completion of the entire computation process, which scales with the increase in agents and available action choices and is of concern in time critical processes. An alternate method using max-plus which reduces the computation time required was used. This was to achieve coordination in multi-agent system when applied to urban traffic signal control.

2.4.3. Coordination through Belief Models

In scenarios where time is of critical importance, coordination through protocols fail to succeed when an agent with a specific resource to solve the sub-problem reject the bid. In such scenarios,

agents with an internal belief model of the neighboring agents could solve the problem. The internal belief model could be either evolved by observing the variation in the dynamics of the environment or developed based on heuristic knowledge and domain expertise. When the internal model is evolved, the agent has to be intelligent enough to differentiate between the change in environment due to other agent actions and natural variations occurring in the environment. In a heuristics based belief model has been employed to create coordination between agents and to effectively change the green time. In evolutionary methods combined with neural networks have been employed to dynamically compute the level of cooperation required between the agents. This is based on the internal state model of the agents. The internal state model was updated using reinforcement learning methods.

2.5. APPLICATIONS OF MULTI-AGENT SYSTEMS

Multi-Agent Systems appeared as new software technologies that integrate a variety of Artificial Intelligence techniques from different subfields (reasoning, knowledge representation, machine learning, planning, coordination, communication and so on), and which offer an efficient and more natural alternative to build intelligent systems, thus giving a solution to the current complex real world problems that need to be solved. For example, a complex system could be decomposed in components and again the components in sub-components and so on, till some primitive entities are obtained. Some of these primitive entities could be viewed as being agents that solve their local problems and interact between them in order to solve the goal of the initial complex system. However, most of the real world complex systems are only nearly decomposable, and a solution would be to endow the components with the ability of making decisions about the nature and the scope of their interactions at run time. Still, from this simplistic view, we could figure out a new type of computing, based on agents. In Jennings argued that agent of the Fish market system is that it is left to the buyers and sellers to encode their

own bidding strategies. Also, the auctions could be monitored by the FM Monitoring Agent that keeps track of every single event taking place during a tournament. In Fish market each agent in a MAS is dynamically attached to a controller module, and it's in charge of controlling its external actions (i.e. protocol execution).

Sardine

In it is described an alternative airline flight bidding prototype system, called SARDINE (System for Airline Reservations Demonstrating the Integration of Negotiation and Evaluation), which offers better choices in comparison to the Priceline system. The SARDINE system uses software agents to coordinate the preferences and interests of each party involved. The buyer agent takes the buyer's preferences and correlates these parameters with the available flights from a reservation database. The user then tells the buyer agent how much to bid and the airline agents accept the ticket bids from the buyer agent. Finally, the airline agents consider individual bids based on flight yield management techniques and specific buyer information. The SARDINE system uses the OR combinatorial auction. A combinatorial auction is one in which the user submits simultaneous multiple bids. The bids are mutually exclusive of one another, thus an OR combinatorial auction is used.

eMediator

The eMediator is a next generation electronic commerce server that has three components: an auction house (eAuctionHouse), a leveled commitment contract optimizer (eCommitter), and a safe exchange planner (eExchangeHouse). The eAuction House allows users from Internet to buy and sell goods as well as to set up auctions. It is a third party site, and therefore both sellers and buyers can trust that it executes the auction protocols as stated. It is implemented in Java and uses some of the computationally

intensive matching algorithms in C++. In order to increase reliability the information about the auctions is stored in a relational database. The server is the first Internet auction that supports combinatorial auctions, bidding via graphically drawn price-quantity graphs, and by mobile agents oriented approaches can significantly enhance our ability to model, design and build complex (distributed) software systems.

A natural way to modularize a complex system is in terms of multiple, interacting autonomous components that have particular goals to achieve, i.e. of a multi-agent system (MAS). A multi-agent approach is an attempt to solve problems that are inherently (physically or geographically) distributed where independent processes can be clearly distinguished. Such problems include, for example, decision support systems, networked or distributed control systems, air traffic control. Therefore, multi-agent systems approach is appropriate for distributed intelligence applications: network based, human involved, physically distributed, decentralized controlled, etc.

The basic notion of agent-computing is the agent with its derivation, the software agent. Several definitions has been given to the notion of agent. According to Michael Wooldridge, an agent is a computer system that is situated in some environment, and is capable of flexible, autonomous action in that environment in order to meet its design objectives. The flexibility characteristic means that the agent is reactive, pro-active and social. Therefore, the key characteristics of agents are autonomy, proactivity, situatedness, and interactivity. More characteristics could be added, such as mobility, locality, openness, believability, learning, adaptation capabilities, comprehensibility, etc. A software agent is an independently executing program able to handle autonomously the selections of actions when expected or limited unexpected events occur.

Summarizing, an agent need to have computational abilities (reasoning, searching, etc) and can use its knowledge and rationality models to map inputs to outputs that would maximize

its utility (its performance measure according to the rationality). According to the interaction strategy that is used, an agent could be cooperative, self-interested, and hostile. Cooperative agents could work together with other agents and humans with the intention of solving a joint problem. Self-interested agents try to maximize their own goods without any concern for the global good, and will perform services for other agents only for compensation (e.g. in terms of money). Hostile agents have a utility that increases with their own gains, and increases also with the competitor's losses.

The agents can be viewed as living in a society where they have to respect the rules of that society. They also live in an organization, which can be effectively executed only in respect with organizational patterns of interactions. In general, multi-agent systems represent institutions where agents must fulfill a set of expected behaviors in their interactions.

2.5.1. Multi-agent systems infrastructures

Each multi-agent system architecture has its specific features: agent registration, agent capability advertisements, strategy for finding agents, agent communication language, agent dialogue mediation, agent content language, and default agent query preference, etc. As multi-agent systems are open, in complex applications, homogeneity cannot be achieved with respect to the MAS architecture specific features. Thus, interoperation mechanisms must be designed and implemented.

RETSINA

RETSINA (Reusable Task Structure-based Intelligent Network Agents) multi-agent infrastructure has been developed at the Carnegie Mellon University in Pittsburgh, USA. It is composed by four different reusable agent types that can be adapted to different applications, the interface agents, task agents, information/ resource agents, and middle agents. A collection of RETSINA

agents forms an open society of reusable agents that self-organize and cooperate in response to task requirements. The RETSINA framework was implemented in Java. It is constructed on the principle that all agents should communicate directly with each other, if necessary. Agents find each other through a Matchmaker agent, who does not manage the transaction between the two agents, it just allow the direct communication between the two agents.

RETSINA is an open MAS infrastructure that supports communities of heterogeneus agents. It does not employ any centralized control on the MAS, rather implements distributed infrastructural services that facilitate the relations between the agents instead of managing them.

The RETSINA-OAA Inter Operator acts as a connection between two MASs with two radically different agent architectures: the RETSINA capability-based MAS architecture and SRI' Open Agent Architecture (OAA). Agents in the OAA system "speak" Prolog-based OAA ICL, while agents in RETSINA system use KQML. The two languages has very different syntactic and semantic structures. OAA is organized around an agent called the Facilitator, which manages all the communications between agents in such a way that OAA agents cannot communicate directly.

SICS

SICS Market Space is an agent-based market infrastructure implemented in Java. The goal of SICS is to enable automation of consumer goods markets distributed over the Internet. It consists in an information model for participant interests, and an interaction model that defines a basic vocabulary for advertising, searching, negotiating and settling deals. The interaction model is asynchronous message communication in a simple speech act based language, the Market Interaction Format (MIL).

2.5.2. Application Areas

We have selected various MAS application areas (which are not disjunctive) and for each area a brief presentation of some MASs developments (the majority being simulations or prototypes) is made. The general application domains selected for this presentation are resource management (ADEPT business management, FACTS telecommunication service, Tele MACS, Challenger, Meta Morph II, MACIV); manufacturing planning, scheduling and control (TELE TRUCK); monitoring, diagnosis and control (ARCHON energy management); electronic commerce (Fish market, SARDINE, eMediator, SMACE, COM_ELECTRON), and virtual enterprise (VIRT_CONSTRUCT).

2.5.3. ARCHON's electricity transportation management application

Energy management is the process of monitoring and controlling the cycle of generating, transporting and distributing electrical energy to industrial and domestic customers. A Spanish company, Iberdrola, that works in the energy domain decided to develop a set of decision support systems (DSS) in order to reduce the operators' cognitive load in critical situations, and to decrease the response time for making decisions. The DSS were interconnected and extended using the ARCHON technology. In it is discussed the problem of development and deployment of MASs in real world settings, and it is analyzed under the ARCHON project and applied to electricity transport management. ARCHON provides a decentralized software platform which offers the necessary control and level of integration to help the subcomponents to work together. Each agent consists of an ARCHON Layer (AL) and an application program (Intelligent System - IS).

Seven agents are running on five different machines. The agents are: BAI (Black-out Area Identifier), CSI-D and CSI-R (pre-existing Control System Interface), BRS (Breaks and Relays Supervisor), AAA (Alarms Analysis Agent), SRA (Service Restoration Agent),

and UIA (User Interface Agent). The BAI agent identifies which elements of the network are initially out of service. CSI is the application's front end to the control system computers and consists of two agents: CSI-D detects the occurrence of disturbances and preprocesses the chronological and non-chronological alarm messages which are used by the agents AAA, BAI and BRS; and CSI-R detects and corrects inconsistencies in the snapshot data file of the network, computes the power flowing through it and makes it available to SRA and UIA. The BRS agent detects the occurrence of a disturbance, determine the type of fault, generates an ordered list of fault hypotheses, validates hypotheses and identifies malfunctioning equipment. The AAA agent has similar goals with BRS. The SRA agent devises a service restoration plan to return the network to a steady state after a blackout has occurred. The UIA agent implements the interface between the users and the MAS.

Due to parallel activation of tasks, efficiency is achieved. Reliability is increased because even if one of the agents break down, the rest of agents can produce a result (not the best one) that could be used by the operator.

The application is operational since 1994. The MAS gives better results because it takes multiple types of knowledge and data into account and integrates them in a consistent manner. Also, the system is robust because there are overlapping functionalities which means that partial results can be produced in the case of agent failure. The system is open, so new agents could be added in an incremental manner.

2.5.4. ADEPT business process management application

An agent-based system developed for managing a British Telecom (BT) business process is presented in. The business process consists in providing customers with a quote for installing a network to deliver a particular type of telecommunications service. The process is dynamic and unpredictable, it has a high degree of natural concurrency, and there is a need to respect departmental and organizational boundaries. In this process, the following

departments are involved: the customer service division (CSD), the design division (DD), the surveyor department (SD), the legal department (LD), and the organizations that provide the out-sourced service of vetting customers (VCs). In the multi-agent system, each department is represented by an agent, and all the interactions between them take the form of negotiations (based on a service-oriented negotiation model). All negotiations are centered on a multi-attribute object, where attributes are, for instance, price, quality, duration of a service.

2.5.5. FACTS telecommunication service management

In the FACTS telecommunication service management the problem scenario is based on the use of negotiation to coordinate the dynamic provisioning of resources for a Virtual Private Network (VPN) used for meeting scheduling by end users. A VPN refers to the use of a public network (as is the Internet) in a private manner. This service is provided to the users by service and network providers. The multi-agent system consists in a number of agents representing the users (Personal Communication Agents), and the service and network providers.

2.5.6. Challenger

In it is described a MAS for distributed resource allocation, Challenger. The MAS consists of agents which individually manage local resources and which communicate with one another in order to share their resources (e.g. CPU time) in an attempt to efficiently use of them. Challenger is similar to other market-based control systems as the agents act as buyers and sellers in a marketplace, always trying to maximize their own utility. Experimental results of using the MAS to perform CPU load balancing in a network of computers (small sized networks, e.g. of maximum 10 machines) are presented. Challenger was designed to be robust and adaptive. It is completely decentralized and consist of a distributed set of agents that run locally on every machine in the network. The main agent behavior is based on a market/bidding metaphor with the

following four steps: job origination, making bids, evaluation of bids, and returning results. Several simulations were run, including learning behaviors of the agents in order to improve the performance of the whole system in some critical situations such as large message delays and inaccurate bids made by the agents.

2.5.7. Tele-MACS

Tele-MACS applied a multi-agent control approach to the management of an ATM network. In telecommunications, Tele-MACS considers link bandwidth allocation and dynamic routing. A multi-layered control architecture has been implemented in Java. Tele-MACS consists of multiple layers of MASs, where each layer is defined to conduct the control of the network infrastructure to a certain level of competence.

2.5.8. Tele Truck

Real-life transport scheduling can be solved by a multi-agent system. Each resource could be represented as an agent and market algorithms are applied to find and optimize solutions. The TELE TRUCK system, presented in, can be applied to online dispatching in a logistics management node of a supply web, and uses telecommunication technologies (satellite, GPS, mobile phones). The truck drivers, trucks, (semi)-trailers are autonomous entities with their own objectives, and only an appropriate group of these entities can perform together the transportation task. Thus the whole problem can be modeled as a MAS. Each entity is an intelligent agent, and has its own plan, goal, and communication facilities in order to provide the resources for the transportation plans according to their role in the society. In the TELE TRUCK system different types of negotiation techniques are used for the allocation of transportation tasks in a network of shipping companies. In the case of vertical cooperation, i.e. the allocation of orders within one shipping company, the simulated trading algorithm is used for dynamic optimization, and also an extended contract net protocol is used for fast and efficient

initial solution (e.g. one order can be split to multiple trucks). The simulated trading algorithm is a randomized algorithm that realizes a market mechanism where contractors attempt to optimize a task allocation by successively selling and buying tasks in several trading rounds. In the case of horizontal cooperation, i.e. the order allocation across shipping companies, a brokering mechanism is used for short-term cooperation. The matrix auction is another negotiation technique that is used. This type of auction is truth revealing and applicable for the simultaneous assignment of multiple items or tasks to bidders. For example, in the case of orders assignment to the vehicles, a bidding procedure is used. The dispatch officer in the shipping company interacts with a dispatch agent. The dispatch agent announces the newly incoming orders via an extended contract net protocol.

2.5.9. Meta MorphII

The MetaMorphII system enables the development of a mediator based multi-agent architecture to support enterprise integration and supply chain management. A federated-based approach is applied. A manufacturing system is seen as a set of subsystems that are connected through special interface agents called facilitators or mediators. Each enterprise has at least one mediator. In the supply chain network, partners, suppliers and customers are connected through their mediators. Other levels of mediators can exist inside an enterprise. The coordination mechanisms that are used include communication between agents through facilitators. Local agents use a restricted subset of an Agent Communication Language (ACL) to inform facilitators about their needs and services they offer. Facilitators use this information as well as their knowledge of the global MAS network to transform local agents' messages and route them to other facilitators. In this way, local agents give a part of their autonomy to facilitators and in turn, the facilitators satisfy their requirements.

2.5.10. Fish market

The Fish market project conducted at the Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona, developed an agent-mediated electronic institution. FM100 is a Java-based version of the Fish market auction house, that allows to define auction-based trading scenarios where goods can be traded using the classical auction protocols (Dutch, English, Vickrey, and First-price Sealed Bid). It has a library of agent templates written in Java, Lisp, and C.

Fishmarket is one of the most popular simulations of an agent-mediated auction house, which offers a convinient mechanism for automated trading due to the simplicity of the conventions used for the interaction when multiparty negotiations are involved, and to the fact that on-line auctions may successfully reduce storage, delivery or clearing house costs in the fish market. FM was designed for the Spanish fish market.

2.5.11. MACIV

MACIV is a multi-agent system for resource management on civil construction companies, developed in Java as an academic prototype used for demonstration of negotiation techniques. In order to achieve an adequate solution and take into account the specific characteristics of the problem, it was decided to adopt a decentralized solution based on multiagent systems techniques. The agent's behaviors were improved through reinforcement learning. A multi-criteria negotiation protocol is used for a society of buyers and sellers, where buyer agents represent human operators requesting for tasks to be executed and seller agents represent resources competing for being used for those task execution.

2.5.12. SMACE

SMACE is a MAS for e-commerce that supports and assists the creation of customized software agents to be used in agent-mediated ecommerce transactions. It has been used for testing automated negotiation protocols, including those based on the continuous double auction that support a multi-issue approach in the bargaining process. Learning is also included as a capability that enhance the MAS performance. The agents may have any negotiation strategy. SMACE was implemented in Java and JATLite was used to provide the communication infrastructure.

2.5.13. COM_ELECTRON

COM_ELECTRON is a multi-agent system developed at University of Ploiesti, which is dedicated to second hand electronic products selling. It has been implemented in JADE as a simulation. For the shopping agents architecture we have used the Smart Agent architecture. The role of a Smart Agent is to assist users while doing electronic shopping in the Internet. The shopping agent may receive proposals from multiple seller agents. Each proposal defines a complete product offering including a product configuration, price, warranty, and the merchant's value-added services. The shopping agent evaluates and orders these proposals based on how they satisfy its owner's preferences (expressed as multi-attribute utilities). He negotiates over a set of issues that describe the characteristics of a good (such as: type of processor, memory capacity, price, hard disk capacity, in the case of a second hand laptop).

The main purpose of the agent-mediated electronic commerce system COM_ELECTRON is the maximization of the profit viewed as an increased number of transactions and deals agreed after some rounds of bilateral negotiation. In order to achieve this purpose, the negotiation model that was adopted by an agent (buyer/seller) is the service-oriented negotiation model described in extended with an adaptability component capability

implemented by a feed-forward artificial neural network, that allows him to model the other agent's negotiation strategy, thus doing an adaptive negotiation in order to make a better deal. The adaptive negotiation model was included in the architecture of the Smart Agent.

This history of price proposals is a time series and the prediction of the next price proposal is made by using a feed forward artificial neural network. The learning capability is activated during the process of proposals and counterproposals exchanging and will influence the way in which the negotiation will evolve to an agreement. For example, the seller agent will reason about the buyer agent based solely on his observations of buyer's actions.

Currently, the COM_ELECTRON system uses a non-mediated interaction coordination model. If a coordination trace inspection it is needed, the architecture of the MAS can be modified by including some mediator agents which will capture the state of the interaction.

2.5.14. VIRT_CONSTRUCT

The agent-based virtual enterprise, VIRT_CONSTRUCT is under development at University of Ploiesti. The implementation of the system is made in JADE. The goal of VIRT_CONSTRUCT is the construction of private houses. Figure 13 presents the MAS view of VIRT_CONSTRUCT.

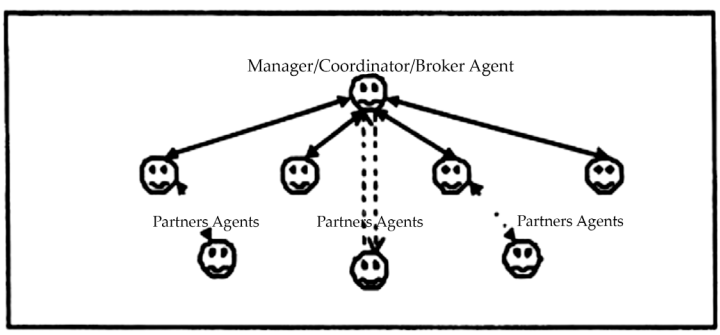


Figure 13. The MAS view of VIRT_CONSTRUCT.

In the case of an agent-based VE, each partner is an agent that will act on behalf of the partner via delegation or in negotiation processes. In the context of an electronic marketplace, the creation of a VE involves the initiation of a competition between different agents that send bids in order to become the VE partners. Figure 14 describes an example of negotiation process between two agents, the Broker-Agent (A) and a potential partner's agent (B) that has the capability of specialized roof construction.

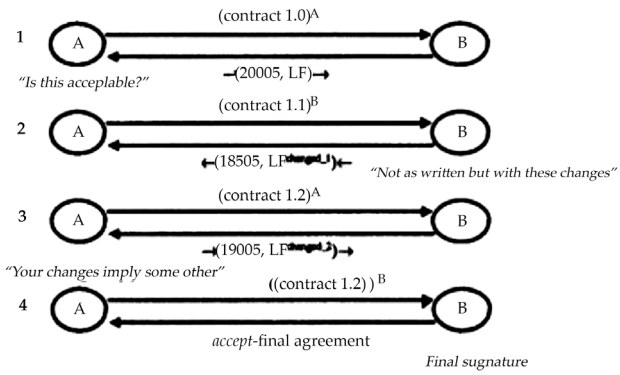
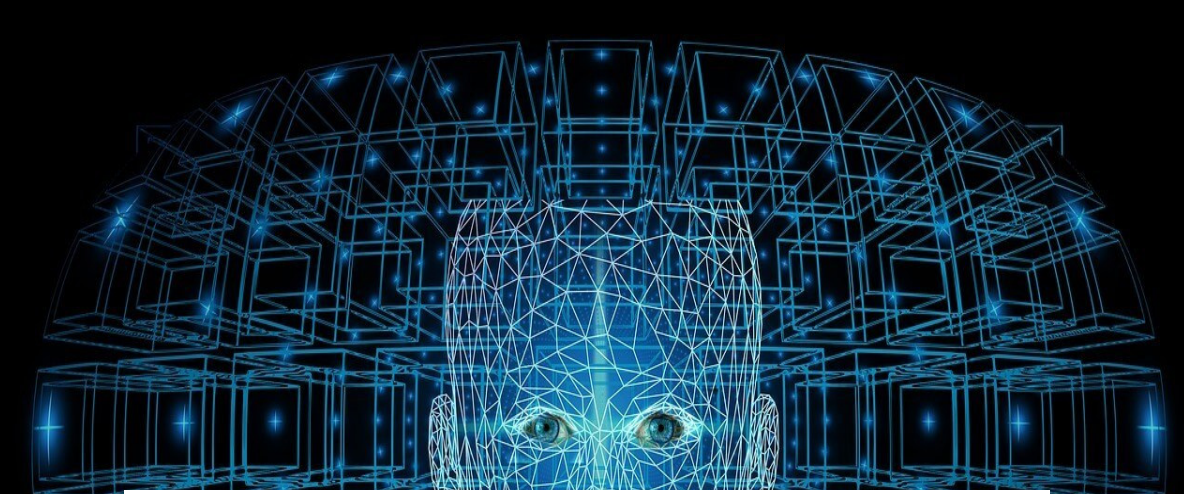


Figure 14. The negotiation process between two agents.

REFERENCES

1. Balaji P.G and D.Srinivasan, "Distributed multi-agent type-2 fuzzy architecture for urban traffic signal control," In IEEE Internationa Conference on Fuzzy Systems, 2009
2. Bergenti, Federico and Ricci, Alessandro, "Three approaches to the coordination of multiagent systems," In Proceedings of the 2002 ACM Symposium on Applied Computing, Mar 2002.
3. Fulcher, J and Jain, L.C. (Editors), Computational Intelligence: A Compendium, Springer-Verlag, 2008.
4. Gabel.T and Riedmiller. M, "On a successful application of multi-agent reinforcement learning to operations research benchmarks," IEEE International Approximate Dynamic Programming and Reinforcement learning 2007.
5. Jain, L.C and De Wilde, P. (Editors), Practical Applications of Computational Intelligence Techniques, Kluwer Academic Publishers, USA, 2001.
6. Jain, L.C., Chen, Z. And Ichalkaranje, N. (Editors), Intelligent Agents and Their Applications, Springer-Verlag, Germany, 2002.
7. Jain, L.C., Sato,M., Virvou, M., Tsihrintzis, G., Balas, V. And Abeynayake, C. (Editors), Computational Intelligence Paradigms: Volume 1 – Innovative Applications, SpringerVerlag, 2008.
8. Khosla, R., Ichalkaranje, N. And Jain, L.C. (Editors), Design of Intelligent Multi-Agent Systems, Springer-Verlag, Germany, 2005.
9. M.C.Choy, D.Srinivasan and R.L.Cheu, " Cooperative, hybrid agent architecture for realtime traffic signal control," IEEE Trans. On Systems, Man and Cybernetics-Part A: Systems and Humans, 2003.
10. Mumford, C. And Jain, L.C.(Editors), Computational intelligence: Collaboration, Fusion and Emergence, Springer-Verlag, 2009.

11. Nikos Vlassis, "A Concise introduction to multiagent systems and distributed artifical intelligence," Synthesis Lectures On Artificial Intelligence And Machine Learning, 1st edition, 2007
12. O.Yadgar, S.Kraus and C.Oritz, "Scaling up distributed sensor networks: Cooperative large scale mobile agent organizations," Distributed Sensor Networks: A Multiagent Perspective, 2003.
13. Paulo Leitao, Paul Valckenaers, and E. Adam, "Self-adaptation for robustness and cooperation in holonic multi-agent systems," Trans. On large-Scale Data- &Knowl.-Cent. Syst. I, LNCS 5740, 2009
14. Resconi, G. And Jain, L.C.,(Editors), Intelligent Agents: Theory and Applications, Springer-Verlag, Germany, 2004



CHAPTER 3

PATTERN RECOGNITION, SIGNAL AND IMAGE PROCESSING

INTRODUCTION

Pattern is everything around in this digital world. A pattern can either be seen physically or it can be observed mathematically by applying algorithms. Pattern recognition is the process of recognizing patterns by using machine learning algorithm. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation. One of the important aspects of the pattern recognition is its application potential.

Signal Processing incorporates all aspects of the theory and practice of signal processing. It features original research work covering novel signal processing tools with a focus on the signal processing issues. It is intended for a rapid dissemination of knowledge to

engineers and scientists working in the research, development or practical application of signal processing.

Signal processing consists of various manipulations or transformations performed on a measured signal. Origin provides a wide array of tools for your signal processing tasks.

Image processing and analysis often require fixed sequences of local operations to be performed at each pixel of an image. Such sequences of operations can be performed in parallel using a pipeline of processors, each operating on the output of the preceding one. The first processor performs the first operation on the image, pixel by pixel. As soon as the first pixel and its neighbors have been processed, the second processor begins to perform the second operation, and so on. Since each processor has available to it the output value of its operation at every pixel, it can also compute statistics of these values, if desired.

3.1 CONCEPT OF PATTERN RECOGNITION

Pattern recognition, in computer science, the imposition of identity on input data, such as speech, images, or a stream of text, by the recognition and delineation of patterns it contains and their relationships. Stages in pattern recognition may involve measurement of the object to identify distinguishing attributes, extraction of features for the defining attributes, and comparison with known patterns to determine a match or mismatch. Pattern recognition has extensive application in astronomy, medicine, robotics, and remote sensing by satellites.

Many modern pattern recognition theories that concentrate on the visual process take for granted that, if the image is appropriately represented, the problem is essentially solved, the association of the appropriately represented image with a particular name being a trivial final step. (Of course, for those interested in the higher cognitive and linguistic processes, this second stage of the pattern recognition problem is central.) Much of the emphasis by many contemporary pattern recognition theorists is, therefore, on image

transformations and representations prior to the comparison process.

The image transformation process is by no means simple or immediate. It is, itself, a major challenge to explanatory theory. Human vision is wonderfully adaptive. At some stage, it seems that the human pattern recognizer normalizes the stimulus so that, even when an object is rotated, translated, or magnified over wide ranges, it can still be recognized. (This invariance is another way of defining stimulus equivalence.)

Most computer models cum theories, as well as psychological models of perception, usually include some preliminary normalization to a canonical configuration or to an invariant representation. (This is especially true for connectionist or neural net models.) If the normalization is done properly, recognition is not dependent upon the particular situational properties of the stimulus, and the model mimics human recognition invariance in a reasonably complete way. For example, simply transforming a stimulus to a polar, as opposed to a Cartesian, coordinate system is one useful means of establishing invariance to rotation and magnification. It is also possible to transform a stimulus to a completely different representational system such as a spectrum of spatial frequencies, for example, a Fourier or a Walsh transform, to provide another means of precluding any sensitivity to irrelevant spatial translations.

In spite of the ubiquitous nature of this theoretical approach, whether such a standardization or canonical transformation of the stimulus actually occurs in human pattern recognition remains an unresolved question. At a behavioral level, human recognition skills exhibit a profound insensitivity to an object's location or its size. It is conceivable, however, that the organic analysis system is so powerful that this minor miracle can be accomplished without any image normalization of the kind to which computational theorists usually retreat. Such a preliminary modification of the image may merely be a convenience, if not a necessity, for the computer modeler or psychological theoretician because of our incomplete understanding of the later stages of processing. The

difficulty of solving problems without some kind of a fixed frame of reference may be much less for the human visual system than for the computer program.

The mathematical problem of defining a canonical coordinate system to achieve good invariance to the various distortions and displacements is not trivial. However effortlessly nervous systems seem to adjust to changes in stimulus position and shape, the general problem posed to the modeler or theoretician whose goal is to describe human pattern recognition is profound, refractory, and clearly not yet solved.

Before searching for a pattern there are some certain steps and the first one is to collect the data from the real world. The collected data needs to be filtered and pre-processed so that its system can extract the features from the data. Then based on the type of the data system will choose the appropriate algorithm among Classification, Regression, and Regression to recognize the pattern.

Classification. In classification, the algorithm assigns labels to data based on the predefined features. This is an example of supervised learning.

Clustering. An algorithm splits data into a number of clusters based on the similarity of features. This is an example of unsupervised learning.

Regression. Regression algorithms try to find a relationship between variables and predict unknown dependent variables based on known data. It is based on supervised learning.

Features can be represented as continuous, discrete, or discrete binary variables. A feature is basically a function of one or more measurements, computed to quantify the significant characteristics of the object. The feature is one of the most important components in the Pattern Recognition system. **Example:** consider a football, shape, size and color, etc. are features of the football.

A feature vector is a set of features that are taken together.

Example: In the above example of football, if all the features (shape, size, color etc.) taken together then the sequence is feature vector ([shape, size, color]). The feature vector is the sequence of features represented as an n-dimensional column vector. In the case of speech, MFCC (Mel-frequency Cepstral Coefficient) is the spectral features of the speech. The sequence of the first 13 features forms a feature vector.

3.1.1 Features of Pattern Recognition

Pattern recognition is a process of finding regularities and similarities in data using machine learning data. Now, these similarities can be found based on statistical analysis, historical data, or the already gained knowledge by the machine itself.

A pattern is a regularity in the world or in abstract notions. If we discuss sports, a description of a type would be a pattern. If a person keeps watching videos related to cricket, YouTube wouldn't recommend them chess tutorials videos.

Examples: Speech recognition, speaker identification, multimedia document recognition (MDR), automatic medical diagnosis.

The features are:

1. Speed and accuracy for the familiar is high
2. It can recognize an unfamiliar object
3. It has the ability to recognize different shapes and object from all angles.
4. It identifies the patterns and objects when partly hidden.
5. During analysis quickly catch the patterns with automaticity.

3.1.2 Training a Pattern Recognition system

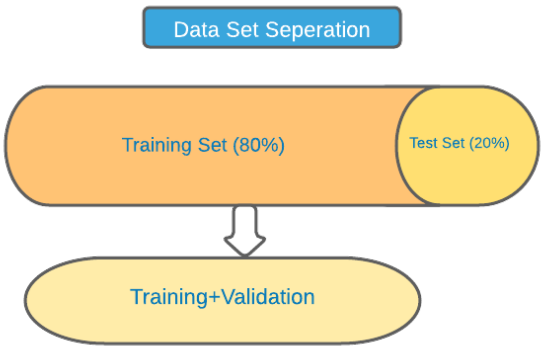


Figure1. Dataset.

Training Set: The training set plays an important part to train the model. Program process this dataset by using training rules. To get the better result one need to collect quite a large dataset because the program will always give better results with a handful of training data. But it may not give the same results in the case of the test dataset. If someone is building a masked face recognizer then he/she will need a lot of images of people wearing a mask. From that dataset, the necessary information will be gathered by the program. Generally, 80% of the total dataset is used as the training dataset.

Validation Set: Fine-tuning helps to train the model. If for the training dataset the accuracy is increasing then a certain portion of data from the training dataset which is unknown to the model is selected to check that for that dataset also the accuracy is increasing. If accuracy is not increasing for the validation set then the program is over fitting the model. In that case, the developer needs to check the value of the parameters or he/she may have to reconsider the model.

Test Set: The test set is used to take the output from the model. After the training, it is used to check how accurate the model is. The rest of the 20% of the dataset is used as a test set.

3.1.3 How does it work?

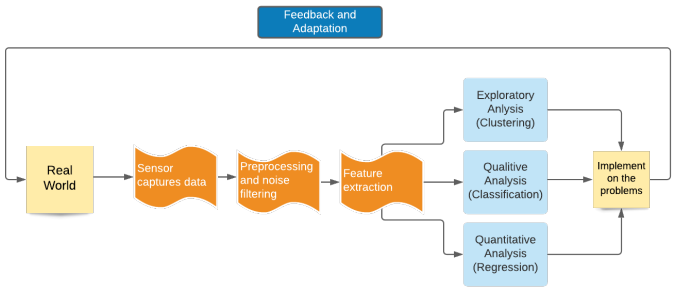


Figure 2. Step by step process.

A pattern recognition system will perceive some input from the real world with sensors. Such a system can work with any type of data: images, texts, videos, or numbers.

After receiving some information as the input, the algorithm starts to pre-process the data. That is segmenting something interesting from the background. For example, when you are given a photo of a park and a familiar face or any object that attracts the user's attention, this is pre-processing.

While the data is in the pre-processing phase it is important to filter the noise from the main dataset. Depending on the working function of the application, the filter algorithm will change. For example, consider a face recognition system where the system is collecting the images for training purposes.

In order to process the data, it will first convert the images from RGB to greyscale. Also, the system doesn't need other areas than the face. So to filter out unwanted portions of the images and replace them with white or black background some filter mechanisms are required. Once those filter mechanisms are used on the data it will be easier for the system to extract features from the filtered images.

Feature extraction is a process of uncovering some characteristic traits that are similar to more than one data sample. The derived

information may be general features, which are evaluated to ease further processing. For example, in image recognition, the extracted features will contain information about grey shade, texture, shape, or context of the image. This is the main information used in image processing. The methods of feature extraction and the extracted features are application dependent.

After extracting the features from the processed data the result of a pattern recognition system will be either a class assignment (labeled dataset), or cluster assignment (dataset without labels), or predicted values (where regression is applied).

3.1.4 Neural Approach

There are quite a few approaches for pattern recognition like Statistical, Syntactical, and Neural. The statistical approach is nothing but to collect historical data and based on the observations and analyses from those data new patterns are recognized. The syntactical approach is also known as the structural approach as it mainly relies upon sub-patterns called primitives like words.

The pattern recognition approaches discussed so far are based on direct computation through machines. Direct computations are based on math and stats related techniques. Other than those techniques another one is the neural approach, neural networks related topics are discussed here to recognize the patterns. As it is known to all neuron is the basic unit of brain cells and together these neurons create networks to control the specific tasks. This neural network is implemented in systems. The outcome of this effort is the invention of artificial neural networks.

An artificial neural network is a computing system that tries to stimulate the working function of a biological neural network of human brains. In this network, all the neurons are well connected and that helps to achieve massive parallel distributing. The input units receive various forms and structures of information based on an internal weighting system and the neural network attempts to learn about the information presented to produce one output report.

The advantages of neural networks are their adaptive-learning, self-organization, and fault-tolerance capabilities. For these outstanding capabilities, neural networks are used for pattern recognition applications. An ANN initially goes through a training phase where it learns to recognize patterns in data, whether visually, aurally, or textually. Some of the best neural models are back-propagation, high-order nets, time-delay neural networks, and recurrent nets.

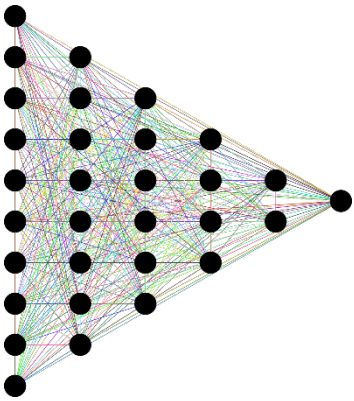


Figure 3. Basic structure of a feed-forward neural network

Normally, only feed-forward networks are used for pattern recognition. Feed-forward means that there is no feedback to the input. Similar to the way that human beings learn from mistakes, neural networks also could learn from their mistakes by giving feedback to the input patterns. This kind of feedback would be used to reconstruct the input patterns and make them free from error; thus increasing the performance of the neural networks. Of course, it is very complex to construct such types of neural networks. These kinds of networks are called auto-associative neural networks. This complexity of constructing the network can be avoided by using back-propagation algorithms. During this supervised phase, the network compares its actual output produced with what it was meant to produce—the desired output.

The difference between both outcomes is adjusted using backpropagation. This means that the network works backward, going from the output unit to the input units to adjust the weight

of its connections between the units until the difference between the actual and desired outcome produces the lowest possible error. Local minima is one of the main problems associated with back-propagation algorithms. In addition, neural networks have issues associated with hyper-parameters like learning rate, architecture selection, feature representation, modularity, and scaling. Though there are problems and obstacles, the application of neural networks has spread everywhere.

3.1.5 Applications

The pattern is the most basic thing for anyone to learn anything. When a baby starts learning he/she tries to search for patterns to identify different objects. Many people use face recognition in photos when posting to social media. Pattern recognition is used to build this face recognition system similar to fingerprint identification.

Basically, a machine tries to capture features of the object and store those features into a vector. These elements in vectors are the attributes of the pattern. Example**:** While representing different types of balls, (circumference, weight, shape, and class) will be Vector and each feature is an element. If the first ball belongs to class 1, the vector would be (22.4cm, 163gm, round, 1), where the value of the last element represents the "cricket ball". Each type of ball falls under a separate class and is denoted by a number.

The field where we use the pattern recognition process:

- Image processing, segmentation, and analysis

Pattern Recognition is efficient enough to give machines human recognition intelligence. This is used for image processing, segmentation, and analysis. For example, computers can detect different types of insects better than humans.

- Computer Vision

Using a pattern recognition system one can extract important features from the images and videos. This is helpful in computer

vision which is applied in different fields', especially biomedical imaging.

- Seismic Analysis

Decision-theoretic and syntactic pattern recognition techniques are employed to detect the physical anomalies (bright spots) and to recognize the structural seismic patterns in two-dimensional seismograms. Here, decision-theoretic methods include Bayes classification, linear and quadratic classifications, tree classification, partitioning-method, and tree classification, and sequential classification.

- Radar Signal Classification

Pattern recognition and signal processing methods are used in a large dataset to find similar characteristics like amplitude, frequencies, type of modulation, scanning type, pulse repetition intervals, etc. Basically, it helps to classify the radio signals, and based upon their class the conversion to digital form is accomplished.

- Speech Recognition

All of us have heard the names Siri, Alexa, and Cortona. These are all the applications of speech recognition. Pattern recognition plays a huge part in this technique.

- Fingerprint Identification

Many recognition approaches are there to perform Fingerprint Identification. But pattern recognition system is the most used approach.

- Medical Diagnosis

Algorithms of pattern recognition deal with real data. It has been found that pattern recognition has a huge role in today's medical diagnosis. From breast cancer detection to covid-19 checking algorithms are giving results with more than 90% accuracy.

- Stock Market Analysis

Patterns are everywhere and nobody can ignore that. Though the stock market is hard to predict still some AI-based applications

are there which are doing using a pattern recognition approach to predict the market. Example: Blumberg, Tinkoff, SofiWealth, and Kosho.

3.1.6 Pattern Recognition Image Processing

Pattern recognition image processing is a method used to scan for similarities in images based on a pixel-by-pixel scan of the source image. While pattern recognition image processing may find exact matches, the software is geared toward finding similarities. The primary use of this type of software is for law enforcement agencies to check for forgeries or for suspects, but it also can be used to search images for something difficult to find or in video editing. This tool is not set up to find exact matches, so it is possible that there may be inaccuracies between two patterns.

To find a specific pattern using pattern recognition image processing, the software performs a pixel-by-pixel scan of a source image and compares it to a second image. A pixel-by-pixel scan means the process looks at every pixel and detects similarities between the two images. It judges the pixels' colors, along with general outlines and features found in the source image; color can change in different environments, so this generally is a secondary metric used in finding similarities. If the pattern recognition software is specialized, such as for finding similarities between faces, then there will be extra metrics for comparing common aspects of the human face, such as the ears and the eyes.

There is software capable of finding exact matches between images, and pattern recognition image processing may be able to find an exact match. At the same time, this process is geared more toward finding similarities, which generally makes the tool more versatile. For example, if the user is looking for similarities between two images of the same beach, but one image has a slightly different perspective, then the exact-matching system will say they have nothing in common.

This process is most often used by law enforcement agencies to identify suspects or to compare signatures and forgeries. With forgeries, the pattern recognition image processing program is not quite as effective as exact-match technology, but it may be able to detect a fake signature. To find a suspect, this program would compare a picture of the suspect with images taken from security cameras in stores and other areas.

While pattern recognition image processing software generally is accurate, it can have problems. For example, if there is someone who looks similar to a suspect, then this software will recognize that similarity and law enforcement may arrest the wrong person. This is because the program is set up more for similarities, but most users know that, so most matches are treated with some caution to ensure that inaccuracies remain minimal.

3.1.7 Types

Pattern recognition is the scientific discipline that allows us to classify objects into several categories or classes that can be further used to perform analysis and improve certain things. The three best-known approaches for pattern recognition are:

1) **Template matching.** Template Matching is used to determine the similarity between two entities (points, curves, or shapes) of the same type. The pattern to be recognized is matched with a stored template along with geometrical transformations. This approach has some obvious disadvantages of being too rigid and having the need for lots of templates
2) **Statistical classification.** In this method, each pattern is represented in terms of some features or measurements. The main objective of this approach is to establish decision boundaries in the feature space. This separates patterns belonging to different classes creating some rules for an inter-class boundary.
3) **Syntactic or structural matching.** This method works on a hierarchy framework where a pattern is said to be

composed of simple sub-patterns that are themselves built from yet simpler sub-patterns. Considered equivalent to languages where primitives are alphabets which make words then lines than the page and then documents.

3.1.8 Pattern recognition methods

Pattern recognition is a computational algorithm used to classify raw data (sometimes appropriate action choice is included in the definition). The term is from machine learning, but has been adapted by cognitive psychologists to describe various theories for how the brain goes from incoming sensory information to action selection. Pattern recognition undergoes an important developing for many years. Pattern recognition include a lot of methods which impelling the development of numerous applications in different filed. The practicability of these methods is intelligent emulation.

Statistical pattern recognition

Statistical decision and estimation theories have been commonly used in PR for a long time. It is a classical method of PR which was found out during a long developing process, it based on the feature vector distributing which getting from probability and statistical model. The statistical model is defined by a family of class-conditional probability density functions Pr(x|ci)(Probability of feature vector x given class ci) In detail, in SPR, we put the features in some optional order, and then we can regard the set of features as a feature vector. Also statistical pattern recognition deals with features only without consider the relations between features.

Data clustering

Its aim is to find out a few similar clusters in a mass of data which not need any information of the known clusters. It is an unsupervised method. In general, the method of data clustering can be partitioned two classes, one is hierarchical clustering, and the other is partition clustering.

The application of fuzzy sets

The thinking process of human being is often fuzzy and uncertain, and the languages of human are often fuzzy also. And in reality, we can't always give complete answers or classification, so theory of fuzzy sets come into being. Fuzzy sets can describe the extension and intension of a concept effectively. The application of fuzzy sets in pattern recognition started in 1966, where the two basic operations –abstraction and generalization the role of fuzzy sets in PR. The PR system based on fuzzy sets theory can imitate thinking process of human being widely and deeply.

Neural networks

Neural networks is developing very fast since the first neural networks model MP was proposed since 1943, especially the Hopfield neural networks and famous BP arithmetic came into being after. It is a data clustering method based on distance measurement; also this method is model-irrespective. The neural approach applies biological concepts to machines to recognize patterns. The outcome of this effort is the invention of artificial neural networks which is set up by the elicitation of the physiology knowledge of human brain. Neural networks are composed of a series of different , associate unit. In addition, genetic algorithms applied in neural networks is a statistical optimized algorithms proposed by Holland (1975) NeurPR is a very attractive since it requires minimum a priori knowledge, and with enough layers and neurons, an ANN can create any complex decision region.

Structural pattern recognition

The concept of structural pattern recognition was put for the fourth time. And structural pattern recognition is not based on a firm theory which relies on segmentation and features extraction. Structural pattern recognition emphases on the description of the structure, namely explain how some simple sub-patterns compose one pattern. There are two main methods in structural

pattern recognition, syntax analysis and structure matching. The basis of syntax analysis is the theory of formal language, the basis of structure matching is some special technique of mathematics based on sub-patterns. When consider the relation among each part of the object, the structural pattern recognition is best. Different from other methods, structural pattern recognition handle with symbol information, and this method can be used in applications with higher level, such as image interpretation. Structural pattern recognition always associates with statistic classification or neural networks through which we can deal with more complex problem of pattern recognition, such as recognition of multidimensional objects.

Syntactic pattern recognition

This method major emphasizes on the rules of composition. And the attractive aspect of syntactic methods is its suitability for dealing with recursion. When finish customizing a series of rules which can describe the relation among the parts of the object, syntactic pattern recognition which is a special kind of structural pattern recognition can be used.

Approximate reasoning approach to pattern recognition

This method which uses two concepts: fuzzy applications and compositional rule of inference can cope with the problem for rule based pattern recognition.

3.2 DEVELOPMENT IN SIGNAL PROCESSING

Signal processing is a major activity. It is generally required to be able to extract the individual frequency components which are present in a signal – sometimes because the original signal generating the vibration contains many components (as is the case for periodic, random or transient types of excitation), and other times because it is required to eliminate spurious components of

response, introduced by noise or nonlinearities in the measuring system. These various requirements can be met with the current generation of spectrum and other frequency response analyzers, often based on digital filtering and frequently employing the FFT or similar algorithms developed in the 1960s, and making signal processing very much faster than it had been hitherto. The advent of the FFT was a major development for modal testing.

The primary output from the measurement stage of a modal test consists of a series of response functions, usually – but not exclusively – FRFs, and these are yielded directly from the output of the signal processing devices (analyzers) used to treat the measured time-histories emitted from the transducers. Here, again, a thorough appreciation of the underlying theory becomes essential for the successful use of the various experimental and signal processing devices. Figure 4 illustrates some typical signals recorded during a modal test, and the resulting FRF that is produced as the 'output' from the measurement phase of the test.

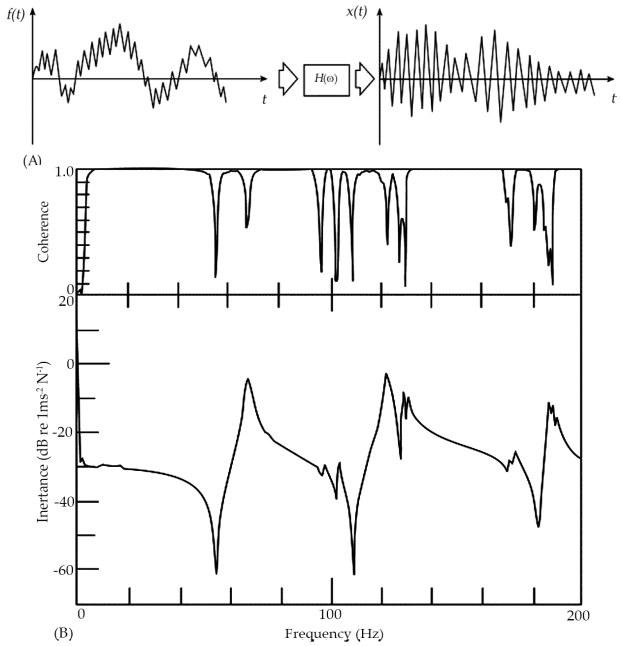


Figure 4. Typical transducer signals and frequency response function produced from modal test. (A) Excitation and response signals under periodic excitation; (B) FRF produced from periodic excitation.

3.2.1 Digital Signal

A digital signal is a way of transmitting data that converts the data to discrete values, usually based on the binary code that computer systems work upon, which consists of packets of information coded as strings of ones and zeros. Using digital signaling allows for an accurate and nearly identical copying of certain types of information like numbers, letters, or the individual pixel colors that make up images, and this information can be stored without long-term degradation of its quality. Where digital signal conversion occurs from what is originally an analog signal, however, such as with music or other natural wave forms, the end result is only an approximation of the original analog signal and some quality in digital format may be lost.

While analog signals are based upon natural processes that utilize the electromagnetic wave forms by which electricity and light are transmitted, digital signal processing requires a digital signal converter. A modulator-demodulator (modem) is such a device. It receives analog signals either from air wave transmissions or telephone lines, and converts them to digital signals that a computer or modern digital television can display as useful information.

Analog signal transmission has been a common form of transmission in technology since the 1800s, but, as of 2007, it is estimated that over 94% of stored and transmitted information has become digital worldwide. This is up from only 3% for digital storage in 1993, and the reasons given for switching to digital signal transmissions is often one of capacity and noise. Analog signals can only be transmitted within a defined range for wavelengths, and, when the signal reaches outside this range or is interfered with by other analog signals along similar wavelengths, distortions and noise can degrade the value of the signal.

Since digital signals are based upon a discrete on/off transmission principle, they have far less susceptibility to corruption over long

distances. A digital signal can also be broken up into separate packets of information known as computer bytes and sent individually to a destination where they are reassembled. This allows for a much more efficient means of transmitting data along randomized networks such as that of the Internet, and it also increases the speed of data transmission over all.

One of the main drawbacks to a digital TV signal or digital cable signal, for instance, is that it is an artificial reproduction of the original data, whereas an analog signal starts out as an exact copy of the original. As a digital signal is translated by multiple devices, encoded as analog and decoded as digital, and reassembled at the end point, quality in the reproduction can be lost. This is due to the fact that digital signals are often copies of copies of copies, and, in the process, approximations must be made by technology to replicate what the original signal was. Wireless digital transmissions can also be corrupted by other wireless activity in the area or radio signals that interfere with them, though this tends to be less of a problem than signal corruption in over the air analog transmissions.

Digital Signal Processing

DSP manipulates different types of signals with the intention of filtering, measuring, or compressing and producing analog signals. Analog signals differ by taking information and translating it into electric pulses of varying amplitude, whereas digital signal information is translated into binary format where each bit of data is represented by two distinguishable amplitudes. Another noticeable difference is that analog signals can be represented as sine waves and digital signals are represented as square waves. DSP can be found in almost any field, whether it's oil processing, sound reproduction, radar and sonar, medical image processing, or telecommunications-- essentially any application in which signals are being compressed and reproduced.

Analog Signal

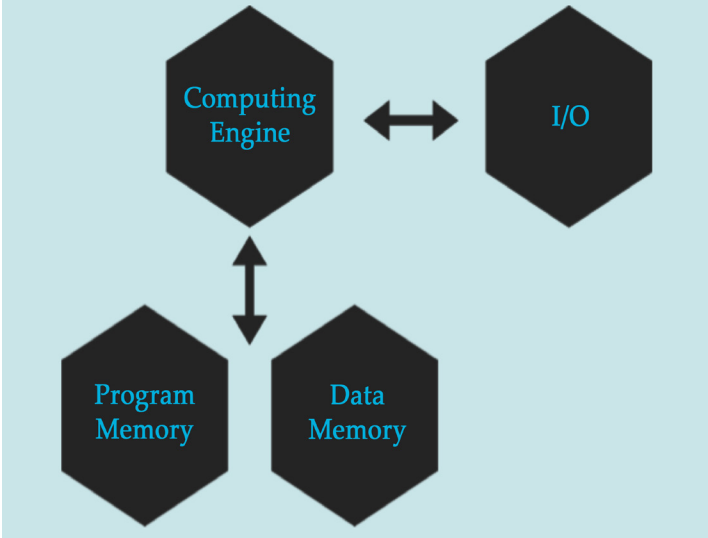
Digital Signal

So what exactly is digital signal processing? The digital signal process takes signals like audio, voice, video, temperature, or pressure that have already been digitized and then manipulates them mathematically. This information can then be represented as discrete time, discrete frequency, or other discrete forms so that the information can be digitally processed. An analog-to-digital converter is needed in the real world to take analog signals (sound, light, pressure, or temperature) and convert them into 0's and 1's for a digital format.

A DSP contains four key components:

- Computing Engine: Mathematical manipulations, calculations, and processes by accessing the program, or task, from the Program Memory and the information stored in the Data Memory.
- Data Memory: This stores the information to be processed and works hand in hand with program memory.
- Program Memory: This stores the programs, or tasks, that the DSP will use to process, compress, or manipulate data.
- I/O: This can be used for various things, depending on the field DSP is being used for, i.e. external ports, serial ports, timers, and connecting to the outside world.

Below is a figure of what the four components of a DSP look like in a general system configuration.



Why Use Digital Signal Processing?

To understand how digital signal processing, or DSP, compares with analog circuitry, one would compare the two systems with any filter function. While an analog filter would use amplifiers, capacitors, inductors, or resistors, and be affordable and easy to assemble, it would be rather difficult to calibrate or modify the filter order. However, the same things can be done with a DSP system, just easier to design and modify. The filter function on a DSP system is software-based, so multiple filters can be chosen from. Also, to create flexible and adjustable filters with high-order responses only requires the DSP software, whereas analog requires additional hardware.

For example, a practical bandpass filter, with a given frequency response should have a stopband roll-off control, passband tuning and width control, infinite attenuation in the stopband, and a response within the passband that is completely flat with zero phase shift. If analog methods were being used, second-order filters would require a lot of staggered high-Q sections, which ultimately means that it will be extremely hard to tune and adjust. While approaching this with DSP software, using a finite impulse

response (FIR), the filter's time response to an impulse is the weighted sum of the present and a finite number of previous input values. With no feedback, its only response to a given sample ends when the sample reaches the "end of the line". With these design differences in mind, DSP software is chosen for its flexibility and simplicity over analog circuitry filter designs.

When creating this bandpass filter, using DSP is not a terrible task to complete. Implementing it and manufacturing the filters is much easier, as you only have to program the filters the same with every DSP chip going into the device. However, using analog components, you have the risk of faulty components, adjusting the circuit and program the filter on each individual analog circuit. DSP creates an affordable and less tedious way of filter design for signal processing and increases accuracy for tuning and adjusting filters in general.

3.2.2 Analog Signal

Analog signals are a representation of time varying quantities in a continuous signal. Basically, a time variance is presented in a manner in which some sort of information is passed using various types of methods, including electrical, mechanical, hydraulic, or pneumatic systems. Unlike digital signals, which use a numeric method of transmitting information, analog signals use small fluctuations in the signal itself to pass information.

These signals act essentially like simulations of a continuous time varying quantity. They duplicate the features of the actual quantity by presenting a different one. In other words, they use one method of recording information and transfer it to a different format that, in turn, presents the information in that medium.

Each analog signal uses a property of the final medium to convey the information for the signal. For example, a thermometer will use the heat of a particular object to determine its temperature. The heat is then transferred to mercury, which changes its position to display the temperature information on the gauge.

The most common analog form of transmission occurs electrically. In order for this to happen, a voltage must be sent at a specific frequency. The flow of this electrical charge is known as current. By controlling the frequency of the current, information can be transmitted to another medium and presented on that medium. For example, magnetic tape on a cassette conveys information to the stereo, which transmits it as electrical signals of specific frequencies, which in turn tell the speakers what noise to make.

Analog signals have a great advantage over digital signals in that they have a much higher density that can present more refined information. Essentially, there is the potential for the signal resolution to be infinite. In addition, the process to create this type of signal is achieved much more simply. By merely adjusting the time quantities, information can be presented.

Disadvantages of the system include the tendency to create unwanted variations in the information transmission such as noise, which can occur in random patterns. When a signal is copied and potentially re-copied, each subsequent version exhibits more of the random patterns, making information transmission harder and ultimately causes signal loss.

In order to avoid these disadvantages, or at least mitigate their effects, the concept of modulation can be used. The base signal is modified in some way to help retain the information as it is transmitted. An example of this is when the amplitude of a waveform is altered in what is known as amplitude modulation. Other options for retaining an electric signal over different generations are by using increased shielding or different cable types twisted together.

Analog Signal Processing

Analog signal processing is the mathematical operation or analysis of analog signals through analog means. This can be conducted in either a discrete or continuous time frame and represents the main way in which these operations perform. Usually this processing involves some form of control, filtering, deblurring or denoising.

In order for the signal to be analog, it must have a continuous value. Basically, the signal must feature a time varying flow of information. Each fluctuation in the signal has important meaning, unlike digital, which deals with numeric representation. The most common context of analog is in electronics, when a signal is sent as physical information.

In terms of audio, analog signal processing is responsible for the changes in bass, treble and volume controls. Video and television use the processing techniques to control the picture's tint. In each of these cases, the voltage and current are controlled by a series of capacitors, resistors, inductors and transistors.

Analog signal processing is defined in the concept of convolution. Convolution defines the parameters in which an input signal combines with the function of a system to determine the accurate output signal. With two analog waveforms, the convolution is the factor in which one of the waveforms is reversed and shifted. To calculate convolution, the first waveform is reversed and shifted to become identical to the second waveform. This creates the analog signal.

The concept of Fourier transformation is also important to analog signal processing. It defines the situation in which the operation transforms a complex analog signal into a series of individual components. This can occur over a period of time or frequency. An example of Fourier transformation is breaking down a musical chord into a number of individual notes.

Frequently, different types of signals are used in analog signal processing. These include sinusoids, impulses and steps. Sinusoids are the basic feature of processing. They demonstrate the deviations in analog through frequency and time variables. Impulses are signals that represent an infinite magnitude and width. The step signals are immediate pulses of information. They demonstrate the sudden input response, similar to the effects of turning a switch.

A number of everyday systems use the concept of analog signal processing. Changing the channel on an analog television requires

the signal be filtered and processed in a certain way. AM/FM radio processes the various analog signals transported throughout the airwaves and creates the output information. Electric guitars also use an analog concept to produce music. The guitar's inductor turns the strings' vibration into electric current.

3.2.3 Signal Processing: Career of the Future

When deciding on a career path, you're likely to have different considerations in mind. You want to have a skillset for which there is a wide demand in growing industries. You also need to be certain your expertise will remain relevant in the technologically uncertain years to come, when AI and automation will change many of our jobs as we know it. Finally,people currently entering the workforce increasingly want to work in fields where they will be able to make a positive social impact. So, is there a way to combine all these needs and desires in one career choice? Yes, if you decide to become a signal processing engineer!

Signal processing the enabling technology for the generation, transformation, extraction and interpretation of information via electronic signals – is essential for our smartphones and wearable devices, as well as the latest health care technologies, digital cameras and our digital assistants like Amazon Echo and Google Home. Taking stock of the immense power and promise of signal processing, it's not difficult to see how it can be the ideal vocational path for a person with an interest in science, technology, or math, and a desire to change the world. The case for a career in signal processing is threefold:

- **Signal Processing Plays a Key Role in Multiple Industries**: Unlike in most fields of study, in signal processing, future jobs are not defined by or restricted to a single professional area. Signals are used to transmit information in nearly every imaginable field. They are used extensively in what will likely be a high-growth industry in years to come: health care. Signal processing is essential for the use of X-rays, MRIs and CT scans,

allowing medical images to be analyzed and deciphered by complex data processing techniques. Signals are used in finance, to send messages about and interpret financial data. This aids decision-making in trading and building stock portfolios. The most exhilarating new movies are made possible by multiresolution signal processing in digital cameras, making entertainment a lucrative market for people with this skillset. And of course, there is the ever-dynamic consumer electronics industry, where smartphones, wearable devices and digital home assistants couldn't exist without processing engineers. Basically, whatever industry interests you probably implements signal processing.

- **Signal Processing is the Technology of the Future:** Many young people are concerned how technology is impacting their future job prospects. But when you're caught in the tide you swim with it, not against it. That's why you should explore the possibilities of signal processing, the technology underpinning most of our disruptive innovations. Just to give a few examples, the latest breakthroughs in health care are enabled by signal processing engineers, who are developing ways to more quickly and accurately process medical images. Hollywood will want to scoop up the best minds in the field to make the most memorable and exciting films. And in Silicon Valley, where disruption is the order of the day, the top tech companies are constantly seeking out experts in signal processing to help develop the latest product or platform. These are the growth industries of the modern economy – and signal processing is the growth skillset in these fields.
- **Signal Processing Can Serve a Social Purpose**: Whether addressing inequality, making our economy more sustainable or combatting epidemics, it's key that researchers and policymakers have access to all the relevant data. Signal processing allows for the expansion of computing power and data storage capabilities,

making signal processing engineers indispensable for understanding and tackling our biggest global problems. A career in this field isn't just about employment opportunities or guarding against your job being automated. It's about contributing to improving the world.

The diversity, relevance and ongoing importance of signal processing makes it an ideal area to study and pursue as a vocation. And just imagine being able to say, in an increasingly digital world, that you - quite literally - make everything possible.

3.2.4 Processing the Interferogram Signal

The signal processing to convert the interferogram into a spectrum usually involves phase correction, zero-filling, and apodization as well as the FT. An ideal interferogram would be perfectly symmetrical about zero path difference. However, in practice, there is some asymmetry because of phase differences between the signals from different wavenumbers. A phase correction routine is applied before the FT. Phase correction can be avoided by scanning the full length of the interferogram on both sides of zero path difference. This allows a magnitude spectrum to be calculated without phase correction. The FT itself uses the fast Fourier transform (FFT) algorithm devised by Cooley and Tukey. This algorithm requires that the number of points to be transformed be equal to a power of two. However, the number of data points collected by scanning to a path difference of 1 cm, for example, does not correspond to a power of two. The interferogram is therefore extended by additional data points with the value of zero to give the required number of points. This process is called zero-filling. It is a common practice to apply further zero-filling to double or quadruple the number of points used. This improves the appearance of the final spectrum by increasing the density of the data points, although it does not improve the resolution. In some systems, interpolation in the spectrum is used to give results that are equivalent to the additional zero-filling.

Apodization is an operation that is necessary because of the finite length of the interferogram. Monochromatic radiation produces a sinusoidal interferogram, but the length of this is limited by the maximum path difference in the interferometer scan. The FT of such a signal is a line of finite width with oscillations on either side. This line has the form of a sinc function (sin x/x) (Figure 5). The oscillations (or sidelobes) that accompany each line are obviously a problem because they could obscure smaller features in the spectrum. They can be reduced at the expense of some broadening of the lines by multiplying the interferogram with a function that decreases with increasing path difference. This process is called apodization. Various functions are used for this, representing different compromises between line broadening and the reduction of the sidelobes. Historically, a simple triangular function was often used, but other functions can reduce the sidelobes further with less broadening. By examining a very large number of alternative functions, Norton and Beer established empirical limits for the reduction in sidelobe intensity that could be achieved for a given increase in linewidth. Three functions identified by Norton and Beer are in common use: weak, medium, and strong, corresponding to increasing degrees of sidelobe reduction and line broadening (Figure 6). The use of no apodization to achieve the narrowest possible lines is called boxcar truncation.

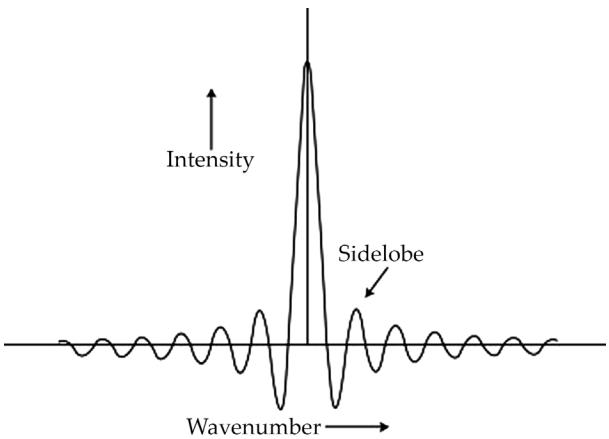


Figure 5. The sinc function generated by Fourier transformation of a cosine wave of finite length.

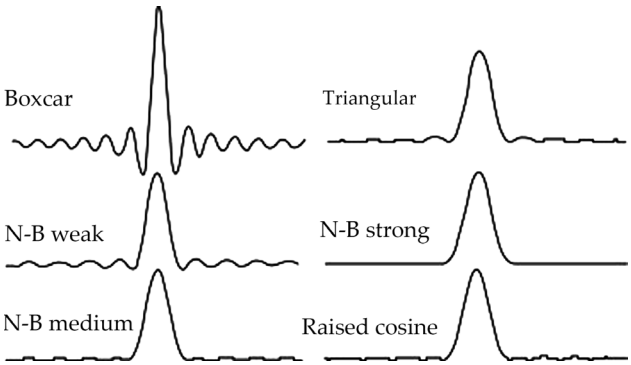


Figure 6. Some common apodization functions and the resulting line-shapes.

3.2.5 Signal and Data Processors

The signal processor is the part of the receiver that extracts the desired target signal from unwanted clutter. It is not unusual for these undesired reflections to be much larger than desired target echoes, in some cases more than one million times larger. Large clutter echoes from stationary objects can be separated from small moving target echoes by noting the Doppler frequency shift produced by the moving targets. Most signal processing is performed digitally with computer technology. Digital processing has significant capabilities in signal processing not previously available with analog methods.

Pulse compression is sometimes included under signal processing. It too benefits from digital technology, but analog processors (e.g., surface acoustic wave delay lines) are used rather than digital methods when pulse compression must achieve resolutions of a few feet or less.

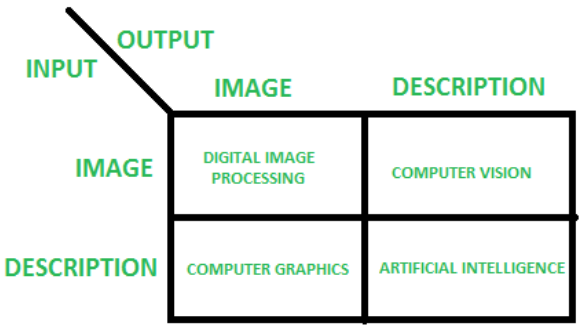
3.3 IMAGE PROCESSING

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful

information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools;
- Analyzing and manipulating the image;
- Output in which result can be altered image or report that is based on image analysis.



There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

3.3.1 Types of Image Processing Applications

Different types of image processing applications include those used in the fields of medicine, digital art, meteorology, law enforcement

and more. Doctors use radiology equipment built with image processing technology for the detection of health problems such as cancerous tumors and blockages in blood vessels. Graphic designers and animation artists use image processing to create illustrations and computer game characters. Meteorologists are able to detect and predict weather patterns through remote sensing technology that uses digital signal processing. Police detectives use digital photo processing technology that is designed to detect specific faces, which helps them in apprehending criminals.

X-ray technology has been around for decades in the healthcare field, and it has been improved through computer processing techniques that allow doctors to view clear and detailed images of internal body systems. Angiography is a specific application of image signal processing that renders highly contrasted images of a patient's blood vessels and any potentially dangerous clots or plaques within them. Image processing applications can also be found in computerized axial tomography (CAT) scans, which have improved the rates of early cancer detection and have thus increased patients' chances of recovery.

Digital photo processing is one of the foundations of computerized graphic arts. Evolutions of dynamic, interactive websites have created a demand for more sophisticated illustrations and animations in order for these types of sites to stand out from the rest. Image processing applications are used in both realistic and non-realistic digital painting and drawing techniques. Artists and animators also use digital filtering to alter and enhance their creations, including rendering them in three dimensions (3D). Computer game design incorporates advanced animation methods to bring characters to life, and these games have become much more realistic than in the past because of improvements in graphics processing.

Image processing applications also have uses in areas of environmental science, particularly in monitoring and reporting weather patterns. Advancements in image processing have led to further developments in remote sensing, which uses satellites

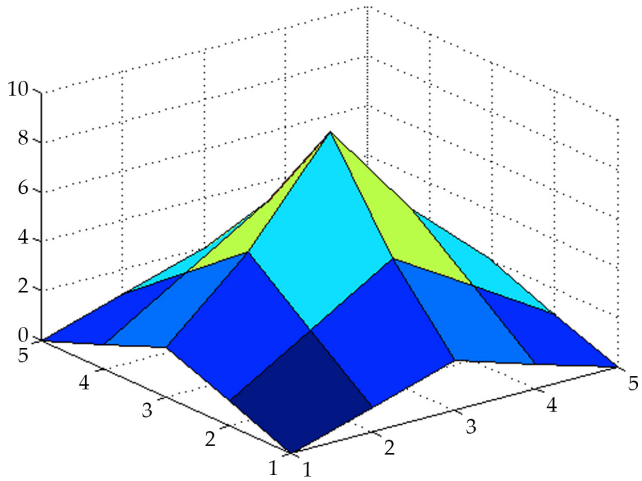
to record light spectrum and pressure changes that would not otherwise be visible to the human eye. This kind of signal processing can be used to create infrared images of storage systems as well as to track their movements over specific time periods.

The mug shots that have been traditionally used in law enforcement have been taken to a new level thanks to image processing. Face recognition technology is able to capture images of suspects through video surveillance and automatically match them to their mug shot images in an existing criminal database. Numbers of captured offenders have increased thanks to this image processing application.

3.3.2 2D Image Processing

A two-dimensional (2D) image processing program is made to work on 2D images instead of three-dimensional (3D) images, which have different axes. One task of 2D image processing is to manipulate an image, either through a filter or by performing some other change to the image. Another part of 2D image processing is to analyze an image, which can be as specific as a pixel analysis or a broader color or light analysis. When a change or analysis is made, the image processor has to output these changes so they are visible on the computer screen or it can make a report about the image itself.

If someone takes a photo of a real-life area, then that area is in 3D, but the image is only in 2D. This is because the image only has an X- and Y-axis, while a 3D computer image is programmed to have a X-, Y- and Z-axis. While 2D image processing programs may have some 3D features, they are primarily made to work with images that only have two axes. Images can be added to the program through a digital camera, which automatically compresses the image into a workable file, or through a scanner.



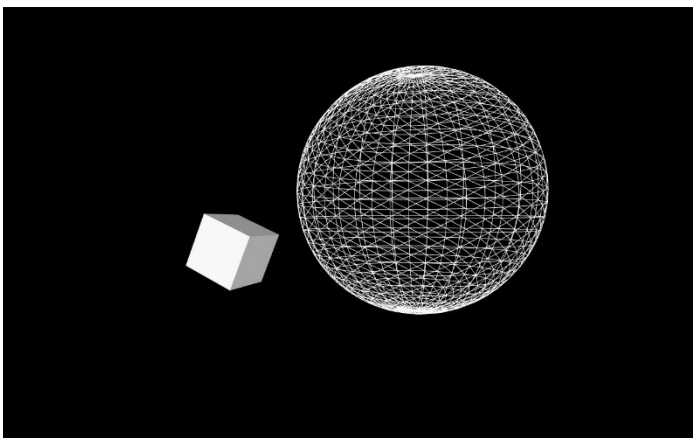
Manipulation is one of the main tasks of 2D image processing. This can be any change, such as drawing on the image, changing the color, adding text or applying a gradient. Many changes are made through use of a filter, which goes through the pixels and changes them. For example, a sepia filter takes the original picture's colors and changes them into a reddish-brown.

Analysis is another import function of 2D image processing. By analyzing the image, the user may be able to uncover hidden information or clear up the image to make it easier to view. For example, medical image processing produces high-contrast images to make it easier to look at blood vessels and bones; meteorologists use filters to change light and wind patterns into visible colors so they can analyze the weather. This helps the user see things the human eye cannot perceive, at least not easily.

After an image is changed or analyzed, it must be outputted. If a screen output is needed, then a 2D image processing program displays the changes on the screen. Report output creates a text report about what was changed or found during the processing; for example, if the 193rd pixel was changed from red to green, the report would highlight that change.

3.3.3 3D Image Processing

Three-dimensional (3D) image processing is the method by which a two-dimensional (2D) image becomes a 3D image, usually from model building and rendering. To create the image, 3D image processing starts with an object's mesh skeleton, which contains many different lines and volume data to correctly represent the 3D space. After the model is built, it is rendered and many different 2D views are captured to create the 3D effect. Entertainment and architecture workers use 3D image processing to build realistic models for movies and buildings, respectively. Doctors also use 3D image processing, because it helps doctors visualize problems, whether internal issues with a patient or for research purposes.



To start the image processing, a mesh object is required. This can either be created from an image processing program, in which users create lines to build up the mesh skeleton, or a 3D scanner can be used to capture the information. Regardless of the technique, the mesh skeleton contains volume and depth information that the computer understands, making it into a 3D model. At this stage, the model does not have any color or texture; it is just a bunch of lines that represent the model's shape and size.

Rendering is the next stage. Designers place colors and textures over the 3D model to make it look realistic. This makes it easier for people to see and understand the image. To make this 3D, the

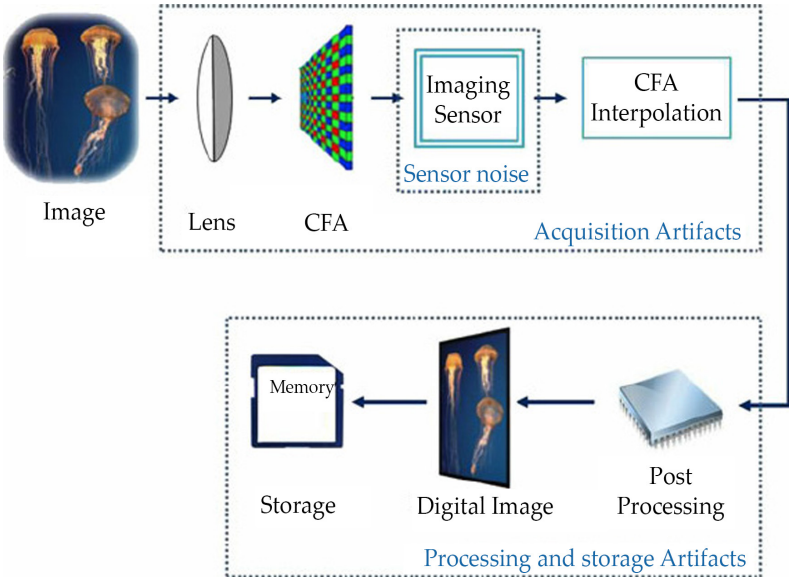
computer takes many different 2D screenshots until it captures every angle, so when the user moves the object, it appears 3D.

The entertainment and architecture industries extensively use 3D image processing to build models for use. Both go through the same process of creating a model and rendering it, but the difference is in how the model is used. In entertainment, the model is meant to move around and interact with actors. Architects use the model so clients can easily visualize the building when it is finished and to make construction easier.

Medical science also makes use of 3D image processing, for both diagnosis and research. In diagnosis, a camera will take pictures of someone's insides, and the camera will be able to create a 3D model of an organ or section that doctors can examine. For research, doctors will be able to watch and study models to see how they react over time; this also helps newcomers to the medical field visualize how internal parts look.

3.3.4 Image Acquisition in Image Processing

Image acquisition in image processing can be broadly defined as the action of retrieving an image from some source, usually a hardware-based source, so it can be passed through whatever processes need to occur afterward. Performing image acquisition in image processing is always the first step in the workflow sequence because, without an image, no processing is possible. The image that is acquired is completely unprocessed and is the result of whatever hardware was used to generate it, which can be very important in some fields to have a consistent baseline from which to work. One of the ultimate goals of this process is to have a source of input that operates within such controlled and measured guidelines that the same image can, if necessary, be nearly perfectly reproduced under the same conditions so anomalous factors are easier to locate and eliminate.



Depending on the field of work, a major factor involved in image acquisition in image processing sometimes is the initial setup and long-term maintenance of the hardware used to capture the images. The actual hardware device can be anything from a desktop scanner to a massive optical telescope. If the hardware is not properly configured and aligned, then visual artifacts can be produced that can complicate the image processing. Improperly setup hardware also may provide images that are of such low quality that they cannot be salvaged even with extensive processing. All of these elements are vital to certain areas, such as comparative image processing, which looks for specific differences between image sets.

One of the forms of image acquisition in image processing is known as real-time image acquisition. This usually involves retrieving images from a source that is automatically capturing images. Real-time image acquisition creates a stream of files that can be automatically processed, queued for later work, or stitched into a single media format. One common technology that is used with real-time image processing is known as background image acquisition, which describes both software and hardware that can quickly preserve the images flooding into a system.

There are some advanced methods of image acquisition in image processing that actually use customized hardware. Three-dimensional (3D) image acquisition is one of these methods. This can require the use of two or more cameras that have been aligned at precisely describes points around a target, forming a sequence of images that can be aligned to create a 3D or stereoscopic scene, or to measure distances. Some satellites use 3D image acquisition techniques to build accurate models of different surfaces.

3.3.5 Machine Vision Image Processing

Machine vision image processing is the analysis of images by a machine. Those with expertise in certain industries use this specific term to refer to extracting data from images for specific purposes. When professionals refer to machine vision image processing, they are often referring to the use of image analysis for controlling industrial activities or processes. While there are many other applications of machine vision, mass production is a prime example of how businesses use this kind of new technology.

Although some associate machine vision image processing with robotics, it is also frequently used in other kinds of technologies. Some of the common tasks for this technique include quality assurance, as well as sorting or handling data for all kinds of process enhancement. In general, this technology helps businesses to enhance their automated processes through automated analysis. For example, when tactile methods are not enough for robotic systems to sort through various shapes and sizes of parts, specialized machine vision image processing methods can often sort parts more efficiently, through very specific algorithms that take in the parameters of the colors or greyscale values in the image to accurately define contours or sizing for an object.

Apart from its industrial applications, using technology to understand images is very valuable for radically different kinds of technology projects. Machine vision image processing exists within a larger field sometimes known as **computer vision** image processing. Its use may be related to military applications, the

consumer product environment, or in specific fields from medicine or transportation to asset maintenance, mechanical production, and much more.

Some of the basic tasks involved in machine vision image processing include the construction of or transformation between black-and-white or grayscale images. Engineers also work on how to achieve pattern recognition and images, or match images up with each other. It is also useful for other specific kinds of projects, such as bar coding or biometrics.

Those interested in what happening in this field can gets more current information from leading industry periodicals, and other venues reporting on how the technique works within the larger field of artificial intelligence. Like other kinds of technology fields, the processing of images for machine vision relies on a supply chain, with hardware like 3-d lasers and software products used to create specific, scalable machine vision applications. Detailed analysis of this field can show IT experts how the vanguard is supporting progress in machine vision through this kind of image analysis and artificial data handling.

3.3.6 Image Processing Techniques

Generally image processing consists of several stages: image import, analysis, manipulation and image output. There are two methods of image processing: digital and analogue. In particular, digital image processing and its techniques.

Computer algorithms play a crucial role in digital image processing. Developers use multiple algorithms to solve different tasks, including digital image detection, analysis, reconstruction, restoration, image data compression, image enhancement, image estimation and image spectral estimation.

Major techniques of digital image processing are as follows:

- **Image Editing**, which basically means altering digital images by means of graphic software tools.

- **Image Restoration**, which refers to the estimation of a clean original image out of the corrupt image taken in order to get back the information lost.
- **Independent Component Analysis**, which separates a multivariate signal computationally into additive subcomponents.
- **Anisotropic Diffusion**, which is often known as Perona-Malik Diffusion, makes it possible to reduce image noise without having to remove important parts of the image.
- **Linear Filtering**. It's another digital image processing technique, which refers to processing time-varying input signals and producing output signals that are subject to the constraint of linearity.
- **Neural Networks**, which are computational models widely used in machine learning for solving various tasks.
- **Pixelation**, which often refers to turning printed images into digitized ones (such as GIF).
- **Principal Components Analysis**, a digital image processing technique that can be used for feature extraction.
- **Partial Differential Equations**, which also is dealing with effectively de-noising images.
- **Hidden Markov Models**, a technique used for image analysis in two dimensions.
- **Wavelets**, which stands for a mathematical function that's used in image compression.
- **Self-organizing Maps**, a digital image processing technique for classifying images into a number of classes.

3.3.7 Image Processing Technique

Classification of Image Processing techniques are given below

1. Image representation
2. Image preprocessing

3. Image enhancement
4. Image analysis
5. Image compression

Image representation

In computer science, the representation of an image can take many forms. Most of the time, it refers to the way that the conveyed information, such as color, is coded digitally and how the image is stored, i.e., how is structured an image file. Several open or patented standards were proposed to create, manipulate store and exchange digital images. They describe the format of image files, the algorithms of image encoding such as compression as well as the format of additional information often called metadata. Differently, the visual content of the image can also take part in its representation.

An image defined as in the "real world" is considered to be a function of two real variables, such example, f(x,y) with f as the amplitude of the image at the real coordinate position (x,y). An image processing operation typically defines a new image g in terms of an existing image f. The effect of digitization is given figure.

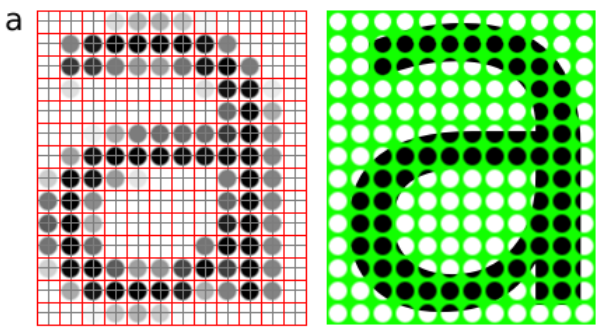


Figure 7. A digital Image Representation.

Image Preprocessing

Preprocessing indicates that the same tissue type may have a different scale of signal intensities for different images. Preprocessing functions involve those operations that are normally required prior to the main data analysis and extraction of information and are generally grouped as radiometric or geometric corrections. Radiometric corrections include correcting the data for sensor irregularities and unwanted sensor or atmospheric noise, removal of non-brain vowels and converting the data so they accurately represent the reflected or emitted radiation to find out a transformation between two images precisely. The preprocessed images will have some noise which should be removed for the further processing of the image. Image noise is most apparent in image regions with low signal level such as shadow regions or under exposed images. There are so many types of noise like salt and pepper noise, film grains .All these noise are removed by using algorithms. Among the several filters, median filter is used.

As a Machine Learning Engineer, data pre-processing or data cleansing is a crucial step and most of the ML engineers spend a good amount of time in data pre-processing before building the model. Some examples for data pre-processing includes outlier detection, missing value treatments and remove the unwanted or noisy data.

Similarly, Image pre-processing is the term for operations on images at the lowest level of abstraction. These operations do not increase image information content but they decrease it if entropy is an information measure. The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task.

There are 4 different types of Image Pre-Processing techniques and they are listed below.

- Pixel brightness transformations/ Brightness corrections
- Geometric Transformations

- Image Filtering and Segmentation
- Fourier transform and Image restoration

Image enhancement

Image enhancement is the procedure of improving the quality and information content of original data before processing. Common practices include contrast enhancement, spatial filtering, density slicing, and FCC. Contrast enhancement or stretching is performed by linear transformation expanding the original range of gray level. Spatial filtering improves the naturally occurring linear features like fault, shear zones, and lineaments. Density slicing converts the continuous gray tone range into a series of density intervals marked by a separate color or symbol to represent different features.

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further image analysis. For example, you can remove noise, sharpen, or brighten an image, making it easier to identify key features.

Here are some useful examples and methods of image enhancement:

- Filtering with morphological operators
- Histogram equalization
- Noise removal using a Wiener filter
- Linear contrast adjustment
- Median filtering
- Unsharp mask filtering
- Contrast-limited adaptive histogram equalization (CLAHE)
- Decorrelation stretch

FCC is commonly used in remote sensing compared to true colors because of the absence of a pure blue color band because further scattering is dominant in the blue wavelength. The FCC is standardized because it gives maximum identical information

of the objects on Earth and satisfies all users. In standard FCC, vegetation looks red (Fig. 8) because vegetation is very reflective in NIR and the color applied is red. Water bodies look dark if they are clear or deep because IR is an absorption band for water. Water bodies give shades of blue depending on their turbidity or shallowness because such water bodies reflect in the green wavelength and the color applied is blue.

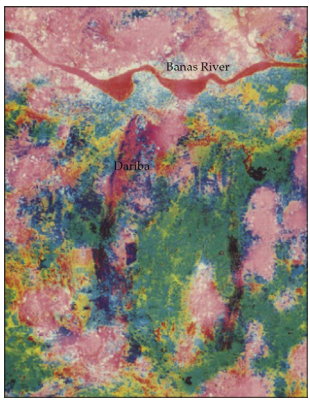


Figure 8. False-color composite image of aeromagnetic and satellite data over Rajpura-Dariba sulfide belt, Rajasthan, India.

Image enhancement techniques can be divided into two wide categories:

- Spatial domain methods, which function directly on pixels
- Frequency domain methods, which function on the Fourier transform of an image.

Image analysis

Image analysis methods extract information from an image by using automatic or semiautomatic techniques termed as scene analysis, image description, image understanding, pattern recognition, computer/machine vision. Image analysis differs from other types of image processing methods, such as enhancement or restoration in that the final result of image analysis procedures is a numerical output rather than a picture.

The image analysis system used comprises primarily an optical microscope and a video camera, with computer control and output devices. The supporting software for image capturing, processing and data sorting, was used to calculate the pore area fraction (porosity) and other measurements characterizing the pore structure, such as the mean size and the number of the pores.

Image analysis is the extraction of meaningful information from images; mainly from digital images by means of digital image processing techniques. Image analysis tasks can be as simple as reading bar coded tags or as sophisticated as identifying a person from their face.

Computers are indispensable for the analysis of large amounts of data, for tasks that require complex computation, or for the extraction of quantitative information. On the other hand, the human visual cortex is an excellent image analysis apparatus, especially for extracting higher-level information, and for many applications — including medicine, security, and remote sensing — human analysts still cannot be replaced by computers. For this reason, many important image analysis tools such as edge detectors and neural networks are inspired by human visual perception models.

Image compression

The objective of image compression is toreduce the size of digital images to save storage space and transmission time. Lossless compression is preferred for artificial images like technical drawings, icons and also be preferred for high value content, such as medical imagery or image scans made for archival purposes. Lossy methods are especially suitable for natural images such as photos in applications where minor loss of fidelity is acceptable to achieve a substantial reduction in bit rate. The lossy compression that produces unnoticeable differences can be called visually lossless. Run-length encoding, Huffman encoding and Lempel Ziv encoding are the methods for lossless image compression.

Transform coding such as DCT, Wavelet transform are applied followed by quantization and symbol coding can be cited as a method for lossy image compression.

Image compression may be lossy or lossless. Lossless compression is preferred for archival purposes and often for medical imaging, technical drawings, clip art, or comics. Lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossy methods are especially suitable for natural images such as photographs in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate. Lossy compression that produces negligible differences may be called visually lossless.

Why Do We Need Image Compression?

Consider a black and white image that has a resolution of 1000*1000 and each pixel uses 8 bits to represent the intensity. So the total no of bits req= 1000*1000*8 = 80,00,000 bits per image. And consider if it is a video with 30 frames per second of the above-mentioned type images then the total bits for a video of 3 secs is: 3*(30*(8, 000, 000))=720, 000, 000 bits

As we see just to store a 3-sec video we need so many bits which is very huge. So, we need a way to have proper representation as well to store the information about the image in a minimum no of bits without losing the character of the image. Thus, image compression plays an important role.

REFERENCES

1. C.M. Bishop: Pattern Recognition and Machine Learning. Springer Verlag, Singapore, 2006.
2. ETSI TS 126 404 V6.0.0. UMTS. General audio codec audio processing functions; Enhanced aacPlus general audio codec; Encoder specification; SBR PART - 3GPP TS 26.404 version 6.0.0 Release 6 (09/2004).
3. Manjunath B.S., Salembier P., and Sikora T. Introduction to MPEG-7: Multimedia Content Description Interface. Wiley & Sons, 2002.
4. R.O.Duda, P.E. Hart, and D.G. Storck: Pattern Classification. 2nd ed., Wiley, 2001.
5. Salomon D., Motta G., and Bryant D. Data Compression: The Complete Reference. Springer, 4th edition, 2006.
6. Sh.Casey. x86 and SSE Floating Point Assists in IA-32: Flush-To-Zero (FTZ) and Denormals-To-Zero (DAZ). Intel Software Network, October 2008. http://software.intel.com/en-us/articles/x87-and-sse-floating-point-assists-in-ia-32-flush-to-zero-ftz-and-denormals-are-zero-daz/
7. Taubman D. S. and Marcellin M. W. JPEG 2000: Image Compression Fundamentals, Standards and Practice. Kluwer International Series in Engineering and Computer Science, Secs 642, 52001

CHAPTER 4

COMPUTER NETWORK AND DISTRIBUTED SYSTEM

INTRODUCTION

A computer network is a system in which multiple computers are connected to each other to share information and resources. A computer network is a group of computers that use a set of common communication protocols over digital interconnections for the purpose of sharing resources located on or provided by the network nodes. The interconnections between nodes are formed from a broad spectrum of telecommunication network technologies, based on physically wired, optical, and wireless radio-frequency methods that may be arranged in a variety of network topologies. The nodes of a computer network may be classified by many means as personal computers, servers, networking hardware, or general purpose hosts. They are identified by hostnames and network addresses. Hostnames serve as memorable labels for the

nodes, rarely changed after initial assignment. Network addresses serve for locating and identifying the nodes by communication protocols such as the Internet Protocol. Computer networks may be classified by many criteria, for example, the transmission medium used to carry signals, bandwidth, and communications protocols to organize network traffic, the network size, the topology, traffic control mechanism, and organizational intent. Computer networks support many applications and services, such as access to the World Wide Web, digital video, digital audio, shared use of application and storage servers, printers, and fax machines, and use of email and instant messaging applications.

Computer network, two or more computers that are connected with one another for the purpose of communicating data electronically. Besides physically connecting computer and communication devices, a network system serves the important function of establishing a cohesive architecture that allows a variety of equipment types to transfer information in a near-seamless fashion. Two popular architectures are ISO Open Systems Interconnection (OSI) and IBM's Systems Network

Architecture (SNA). Two basic network types are local-area networks (LANs) and wide-area networks (WANs). LANs connect computers and peripheral devices in a limited physical area, such as a business office, laboratory, or college campus, by means of links (wires, Ethernet cables, fiber optics, Wi-Fi) that transmit data rapidly. A typical LAN consists of two or more personal computers, printers, and high-capacity disk-storage devices called file servers, which enable each computer on the network to access a common set of files. LAN operating system software, which interprets input and instructs networked devices, allows users to communicate with each other; share the printers and storage equipment; and simultaneously access centrally located processors, data, or programs (instruction sets). LAN users may also access other LANs or tap into WANs. LANs with similar architectures are linked by "bridges," which act as transfer points. LANs with different architectures are linked by "gateways," which convert data as it passes between systems. WANs connect computers and smaller networks to larger networks over greater geographic areas, including different continents. They may link the computers by means of cables, optical fibers, or satellites, but their users commonly access the networks via a modem (a device that allows computers to communicate over telephone lines). The largest WAN is the Internet, a collection of networks and gateways linking billions of computer users on every continent.

4.1 OVERVIEW OF COMPUTER NETWORKS

A computer network consists of a collection of computers, printers and other equipment that is connected together so that they can communicate with each other. Figure gives an example of a network in a school comprising of a local area network or LAN connecting computers with each other, the internet, and various servers. Broadly speaking, there are two types of network configuration, peer-to-peer networks and client/server networks. Computer networking is the practice of interfacing two or more computing

devices with each other for the purpose of sharing data. Computer networks are built with a combination of hardware and software.

Computer network is a telecommunication channel through which we can share our data. It is also called data network. The best example of computer network is Internet. Computer network does not mean a system with control unit and other systems as its slave. It is called a distributed system. A network computer, or NC, is one of multiple diskless computers dependent upon and connected to another computer that holds hard drive space or processes information. The term was originally trademarked by Oracle in 1996, but now refers to any arrangement of computers connected to a common server. Also called a diskless node, hybrid client, or thin client, a network computer is a diskless computer that lacks disk drives, depends entirely upon a server for information storage, and must boot from a central server each time it starts.

Although similar, diskless nodes and thin clients have distinct differences. A diskless node may do its own information processing, relying on the server at boot-ups and for more complex information processing. A thin client essentially relies on the server for boot-ups and all information processing. Thin clients process only their own user interface. Network computer arrangements have various advantages over a computer network of workstations with hard drives. Network computers are cheaper to manufacture and less expensive to run and maintain. They are also easier to update because only the server needs to be updated

rather than an entire network.

A network computer can be run with only basic software, cutting costs for the entire network. This efficiency may come with a price. Diskless workstations and thin clients may work more slowly when many workstations are in use, taxing the central server. The original concept of a network computer came about shortly after the release of Microsoft Windows 95. Oracle attempted to develop a concept that would render Microsoft's operating system obsolete by introducing a computer that had none. Instead, Oracle's network computer would rely on Oracle databases for accessing applications and storing information. Oracle's network computer would be much cheaper than Microsoft's personal computer.

Unfortunately, the product was released before it was ready. Network computers ran slowly, and Oracle lacked sufficient infrastructure to support the product. Personal computer prices dropped, making them even more attractive to consumers and more competitive with network computers. Oracle's network computers were unable to compete, and the product was deemed a failure. Still, the concept was not completely abandoned, and other developers have used aspects of network computer technology to develop newer, network computer–like technologies, like the diskless node, thin client, and hybrid client. Netbooks and smartphones are examples of devices that use scaled-back operating systems and are used primarily for Internet access and applications that rely on Internet access.

4.1.1 History of Computer Networking

Computer networking may be considered a branch of computer science, computer engineering, and telecommunications, since it relies on the theoretical and practical application of the related disciplines. Computer networking was influenced by a wide array of technology developments and historical milestones.

- In the late 1950s, a network of computers was built for the U.S. military radar system Semi-Automatic Ground Environment (SAGE) using the Bell 101 modem. It was

the first commercial modem for computers, released by AT&T Corporation in 1958. The modem allowed digital data to be transmitted over regular unconditioned telephone lines at a speed of 110 bits per second (bit/s).

- In 1959, Christopher Strachey filed a patent application for time-sharing and John McCarthy initiated the first project to implement time-sharing of user programs at MIT. Stratchey passed the concept on to J. C. R. Licklider at the inaugural UNESCO Information Processing Conference in Paris that year. McCarthy was instrumental in the creation of three of the earliest time-sharing systems (Compatible Time-Sharing System in 1961, BBN Time-Sharing System in 1962, and Dartmouth Time Sharing System in 1963).
- In 1959, Anatolii Ivanovich Kitov proposed to the Central Committee of the Communist Party of the Soviet Union a detailed plan for the re-organization of the control of the Soviet armed forces and of the Soviet economy on the basis of a network of computing centers, the OGAS.
- In 1959, the MOS transistor was invented by Mohamed Atalla and Dawon Kahng at Bell Labs. It later became one of the basic building blocks and "work horses" of virtually any element of communications infrastructure.
- In 1960, the commercial airline reservation system semi-automatic business research environment (SABRE) went online with two connected mainframes.
- In 1963, J. C. R. Licklider sent a memorandum to office colleagues discussing the concept of the "Intergalactic Computer Network", a computer network intended to allow general communications among computer users.
- Throughout the 1960s, Paul Baran and Donald Davies independently developed the concept of packet switching to transfer information between computers over a network. Davies pioneered the implementation of the concept. The NPL network, a local area network at the National Physical Laboratory (United Kingdom)

used a line speed of 768 kbit/s and later high-speed T1 links (1.544 Mbit/s line rate).

- In 1965, Western Electric introduced the first widely used telephone switch that implemented computer control in the switching fabric.
- In 1969, the first four nodes of the ARPANET were connected using 50 kbit/s circuits between the University of California at Los Angeles, the Stanford Research Institute, the University of California at Santa Barbara, and the University of Utah. In the 1970s, Leonard Kleinrock carried out mathematical work to model the performance of packet-switched networks, which underpinned the development of the ARPANET. His theoretical work on hierarchical routing in the late 1970s with student Farouk Kamoun remains critical to the operation of the Internet today.
- In 1972, commercial services were first deployed on public data networks in Europe, which began using X.25 in the late 1970s and spread across the globe. The underlying infrastructure was used for expanding TCP/IP networks in the 1980s.
- In 1973, the French CYCLADES network was the first to make the hosts responsible for the reliable delivery of data, rather than this being a centralized service of the network itself.
- In 1973, Robert Metcalfe wrote a formal memo at Xerox PARC describing Ethernet, a networking system that was based on the Aloha network, developed in the 1960s by Norman Abramson and colleagues at the University of Hawaii. In July 1976, Robert Metcalfe and David Boggs published their paper "Ethernet: Distributed Packet Switching for Local Computer Networks" and collaborated on several patents received in 1977 and 1978.
- In 1974, Vint Cerf, Yogen Dalal, and Carl Sunshine published the Transmission Control Protocol (TCP)

specification, RFC 675, coining the term Internet as a shorthand for internetworking.

- In 1976, John Murphy of Datapoint Corporation created ARCNET, a token-passing network first used to share storage devices.
- In 1977, the first long-distance fiber network was deployed by GTE in Long Beach, California.
- In 1977, Xerox Network Systems (XNS) was developed by Robert Metcalfe and Yogen Dalal at Xerox.
- In 1979, Robert Metcalfe pursued making Ethernet an open standard.
- In 1980, Ethernet was upgraded from the original 2.94 Mbit/s protocol to the 10 Mbit/s protocol, which was developed by Ron Crane, Bob Garner, Roy Ogus, and Yogen Dalal.
- In 1995, the transmission speed capacity for Ethernet increased from 10 Mbit/s to 100 Mbit/s. By 1998, Ethernet supported transmission speeds of 1 Gbit/s. Subsequently, higher speeds of up to 400 Gbit/s were added (as of 2018). The scaling of Ethernet has been a contributing factor to its continued use.

A computer network extends interpersonal communications by electronic means with various technologies, such as email, instant messaging, online chat, voice and video telephone calls, and video conferencing. A network allows sharing of network and computing resources. Users may access and use resources provided by devices on the network, such as printing a document on a shared network printer or use of a shared storage device. A network allows sharing of files, data, and other types of information giving authorized users the ability to access information stored on other computers on the network. Distributed computing uses computing resources across a network to accomplish tasks.

4.1.2 Network Concept

The generic term "network" refers to a group of entities (objects, people, etc.) which are connected to one another. A network, therefore, allows material or immaterial elements to be circulated among all of these entities, based on well-defined rules.

- ***Network***: A group of computers and peripheral devices connected to each other. Note that the smallest possible network is two computers connected together.
- ***Networking***: Implementing tools and tasks for linking computers so that they can share resources over the network.
- ***Transportation Network***: A combination of infrastructure and vehicles used for transporting people and goods between different geographic areas.
- ***Telephone Network***: Infrastructure for transporting voice signals from one telephone station to another.
- ***Neural Network***: A group of brain cells connected to each other
- ***Criminal Network***: A group of con artists in cahoots (wherever there is one con artist, there is usually another!)
- ***Computer Network***: A group of computers linked to each other with physical lines, exchanging information as digital data (binary values, i.e. values encoded as a signal which may represent either 0 or 1)

There is not just one kind of network, as there have historically been different kinds of computers, which communicate using various different languages. The need for multiple types of networks also arises from the heterogeneity of the physical transmission media that link them together, whether that means the data is transferred the same way (such as by electrical pulses, light beams, or electromagnetic waves) or uses the same kind of physical medium.

4.1.3 Network Accessories

Network accessories are as follows:

c

Network Cables

Network cables are used to connect one network device to other network devices or to connect two or more computers to share printer, scanner etc. Different types of network cables like coaxial cable, optical fiber cable, and twisted pair cables are used depending on the network's topology, protocol and size.

Hub

This is a hardware device that is used to network multiple computers together. It is a central connection for all the computers in a network, which is usually Ethernet-based. Information sent to the hub can flow to any other computer on the network. If one needs to connect more than two computers together, a hub will allow one to do so. If one only needs to network two computers together, a simple crossover Ethernet cable will do the trick.

Routers

The router is connected to at least two networks and decides which way to send each information packet based on its current understanding of the state of the networks it is connected to. A router is located at any gateway (where one network meets another), including each point-of-presence on the Internet. A router

is often included as part of a network switch. A router may create or maintain a table of the available routes and their conditions and use this information along with distance and cost algorithms to determine the best route for a given packet. Typically, a packet may travel through a number of network points with routers before arriving at its destination. Routing is a function associated with the Network layer (layer 3) in the standard model of network programming, the Open Systems Interconnection (OSI) model. A layer-3 switch is a switch that can perform routing functions.

An edge router is a router that interfaces with an asynchronous transfer mode (ATM) network. A router is a network bridge combined with a router.

In programming, the Open Systems Interconnection, a bridge is a product that connects a local area network (LAN) to another local area network that uses the same protocol (, Ethernet or token ring). One can envision a bridge as being a device that decides whether a message from one to someone else is going to the local area network in building or to someone on the local area network in the building across the street. A bridge examines each message on a LAN, "passing" those known to be within the same LAN, and forwarding those known to be on the other interconnected LAN (or LANs).

In bridging networks, computer or node addresses have no specific relationship to location. For this reason, messages are sent out to every address on the network and accepted only by the intended destination node. Bridges learn which addresses are on which network and develop a learning table so that subsequent messages can be forwarded to the right network. Bridging networks are generally always interconnected local area networks since broadcasting every message to all possible destinations would flood a larger network with unnecessary traffic. For this reason, router networks such as the Internet use a scheme that assigns addresses to nodes so that a message or packet can be forwarded only in one general direction rather than forwarded in all directions. A bridge works at the data-link (physical network) level of a network, copying a data frame from one network to the next network along the communications path.

Network Software

The broader concept of networking encompasses any interaction among two or more people or machines. The software included in the networking category relates to the systems that allow computers, portable devices, and other machines to connect to each other and other such networks, including the globally connected system of servers and computers commonly known as the internet.

The subject of computer networking often falls into the category of information technology or information systems, though it is not restricted to the business and enterprise IS/IT audience. Today, even novice PC users are constructing and maintaining their own home networks, and many workers access their business computers from other locations using remote access. Successful networks depend on a large number of variable factors, and managing those elements is a key ingredient in maintaining a reliable and effective system. Products such as Bandwidth Monitor Pro, Packet Analyzer Enterprise Edition, and SysAid Help Desk and Inventory offer assistance in tracking and monitoring the network.

As the growth of Wi-Fi technologies such as the 802.11g standard continues, more products are beginning to offer security for wireless networks and devices, including Trend Micro PC-cillin and McAfee Wireless Home Network Security. One can only expect the trend to accelerate as more cities, cafes, and other public and private spaces adopt Wi-Fi. A general phrase for software that is designed to help set up, manage, and/or monitor computer networks. Networking software applications are available to manage and monitor networks of all sizes, from the smallest home networks to the largest enterprise networks.

Network Protocol

The software on the individual personal computers and the software on the server have to cooperate with each other to provide access to the shared resources. This is done by designing the software to use an agreed set of conventions (a "protocol") that is standardized so that multiple vendors can provide interoperable systems. There are several different network protocols in use, and they can co-exist on the same network interconnection if they are properly designed. , an Ethernet network can simultaneously support general-purpose protocols, such as LAN Manager, DECnet, TCP/IP, AppleTalk, Novell, and at the same time special-purpose protocols, such as LAT and XNS.

Protocols specify, , that each data packet will have a header that includes agreed numbers of bits specifying:

- The sender's address
- The intended recipient's address
- The type of packet
- The length of the data segment in the packet and so on.

The protocols also specify the circumstances, if any, under which the recipient is to send an acknowledgment of receipt of the data packet. Some protocols include a sequence number that permits the recipient to re-assemble a longer message that has been split into parts; even if those parts are not delivered in the same order

that they were sent. This can easily happen in an overloaded local-area network, or in a complex wide-area network that includes alternative pathways connecting the two machines.

Network Topologies

Network topology refers to the way that the computer network is arranged. The network can have a physical or a logical topology. The physical topology describes the layout of computers and where the workstations are positioned. The logical network topology describes how the information flows through the network. Choosing the physical topology is important because if it is not chosen correctly, this could cause the network to not operate properly. There are several terms that describe the type of physical topology that a network can have. The most common topologies are bus, ring, star, and mesh.

In a bus topology, all of the computers are attached to a single cable using terminators. The terminators work to absorb the energy from the signals in the network. The bus topology is easy to install, but it is not reliable because a single default can bring down network. In a ring topology, each computer is connected directly to two other computers on the network. As with a bus topology, a single fault can disrupt a ring network. This type of network does have advantages, however, and does not require a network server.

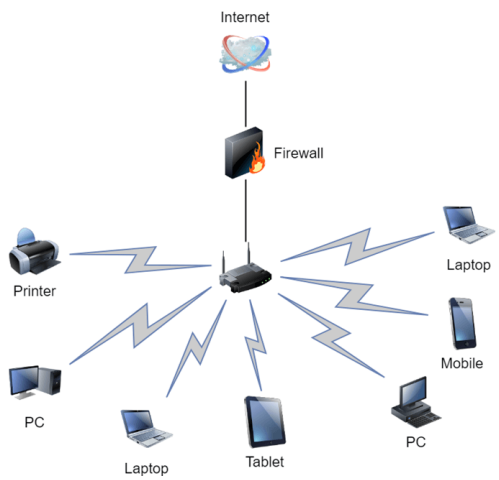
In a star topology, each computer is connected using its own separate cable. This set up is more reliable than a bus topology because its design makes it fault tolerant and susceptible to errors. The information on the network is transmitted from one system to another and the data flows in one single direction. The topology is expensive to maintain and is not reliable because removing one computer can disrupt the entire network.

In a mesh topology, a path is present from one computer to another computer in the network. The mesh topology is usually used in internet structure. The mesh topology can be complicated

to construct because it has multiple connection between locations. For every computer that one has one will need at least one and a half connections for each one. This contributes to the expensive structure. When setting up a network at home, consider a network topology that is simple and easy to maintain. One will also want a topology that is inexpensive to set up. If one is choosing a topology for a business, one will want to consider a topology that is reliable and resistant to errors. The customers will need a network that is free of downtime; therefore, be sure that the topology is one that withstands disruption.

4.2 THE CLIENT-SERVER MODEL IN A DISTRIBUTED NETWORK COMPUTING SYSTEM

A distributed computing system is a set of application and system programs, and data dispersed across a number of independent personal computers connected by a communication network. In order to provide requested services to users the system and relevant application programs must be executed. Because services are provided as a result of executing programs on a number of computers with data stored on one or more locations, the whole computing activity is called distributed computing



A distributed system is a system consisting of a collection of autonomous machines connected by communication networks and equipped with software systems designed to produce an integrated and consistent computing environment. Distributed systems enable people to cooperate and coordinate their activities more effectively and efficiently. The key purposes of the distributed systems can be represented by: resource sharing, openness, concurrency, scalability, fault-tolerance and transparency

Resource sharing. In a distributed system, the resources - hardware, software and data can be easily shared among users. , a printer can be shared among a group of users.

Openness. The openness of distributed systems is achieved by specifying the key software interface of the system and making it available to software developers so that the system can be extended in many ways.

Concurrency. The processing concurrency can be achieved by sending requests to multiple machines connected by networks at the same time.

Scalability. A distributed system running on a collection of a small number of machines can be easily extended to a large number of machines to increase the processing power.

Fault-tolerance. Machines connected by networks can be seen as redundant resources, a software system can be installed on multiple machines so that in the face of hardware faults or software failures, the faults or failures can be detected and tolerated by other machines.

Transparency. Distributed systems can provide many forms of transparency such as:

- Location transparency, which allows local and remote information to be accessed in a unified way
- Failure transparency, which enables the masking of failures automatically; and

- Replication transparency, which allows duplicating software/data on multiple machines invisibly.

Computing in the late 1990s has reached the state of Web-based distributed computing. A basis of this form of computing is distributed computing which is carried out on distributed computing systems. These systems comprise the following three fundamental components:

- personal computers and powerful server computers,
- local and fast wide area networks, internet, and
- Systems, in particular distributed operating systems, and application software.

4.2.1 Concepts of Distributed Network Computing

The problem is how to formalize the development of distributed computing. The above shows that the main issue of distributed computing is programs in execution, which are called processes. The second issue is that these processes cooperate or compete in order to provide the requested services. This means that these processes are synchronized.

A natural model of distributed computing is the client-server model, which is able to deal with the problems generated by distribution, could be used to describe computation processes and their behavior when providing services to users, and allows design of system and application software for distributed computing systems.

According to this model there are two processes, the client, which requests a service from another process, and the server, which is the service provider. The server performs the requested service and sends back a response. This response could be a processing result, a confirmation of completion of the requested operation or even a notice about a failure of an operation.

From the user's point of view a distributed computing system can provide the following services: printing, electronic mail, file service, authentication, naming, database service and computing service. These services are provided by appropriate servers. Because of the restricted number of servers (implied by a restricted number of resources on which these servers were implemented), clients compete for these servers.

An association between this abstract model and its physical implementation is shown in Figure 1. In particular the basic items of the model: the client and server, and request and response are shown. In this case, the client and server processes execute on two different computers. They communicate at the virtual (logical) level by exchanging requests and responses. In order to achieve this virtual communication, physical messages are sent between these two processes. This implies that operating systems of computers and a communication system of a distributed computing system are actively involved in the service provision.

A more detailed client-server model has three components:

- *Service*: A service is a software entity that runs on one or more machines. It provides an abstraction of a set of well-defined operations in response to applications' requests.
- *Server*: A server is an instance of a particular service running on a single machine.

- *Client:* A client is a software entity that exploits services provided by servers. A client can but does not have to interface directly with a human user.

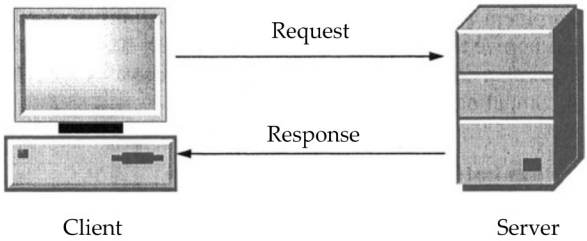


Figure 1: The basic client-server model.

4.2.2 Features and Problems of the Client-Server Model

The most important features of the client-server model are simplicity, modularity, extensibility and flexibility. Simplicity manifests itself by closely matching the flow of data with the control flow. Modularity is achieved by organizing and integrating a group of computer operations into a separate service. Also any set of data with operations on this data can be organized as a separate service. The whole distributed computing system developed based on the client-server model can be easily extended by adding new services in the form of new servers. The servers which do not satisfy user requirements can be easily modified or even removed. Only the interfaces between the clients and servers must be maintained.

There are three major problems of the client-server model:

- The first is due to the fact that the control of individual resources is centralized in a single server. This means that if the computer supporting a server fails, then that element of control fails. Such a solution is not tolerable if a control function of a server is critical to the operation of the system (e.g., a name server, a file server, an authentication server). Thus, the reliability and availability of an operation depending on multiple

servers is a product of reliability of all computers and devices, and communication lines.

- The second problem is that each single server is a potential bottleneck. The problem is exacerbated as more computers with potential clients are added to the system.
- The third problem arises when multiple implementations of similar functions are used to improve the performance of a client-server based system because of a need to maintain consistency. Furthermore, this increases the total costs of a distributed computing system.

4.2.3 Cooperation between Clients and Servers

Cooperation Type and Chained Server

A system in which there is only one server and one client would not be able to provide high performance, and reliable and cost effective services to users. It is necessary to use one server to provide services to more than one client. The simplest cooperation between clients and servers based on sharing allows for lowering the costs of the whole system and more effective use of resources. An example of a service based on this cooperation is a printing service. Figure 2 shows a printer server providing services to n clients, which all are connected by a local area network.

In a distributed computing system there are two different types of cooperation between clients and servers. The first type assumes that a client requests a temporary service. The second one is generated by a client that wants to arrange a number of calls to be directed to a particular serving process. This implies a need for establishing long term bindings between this client and a server.

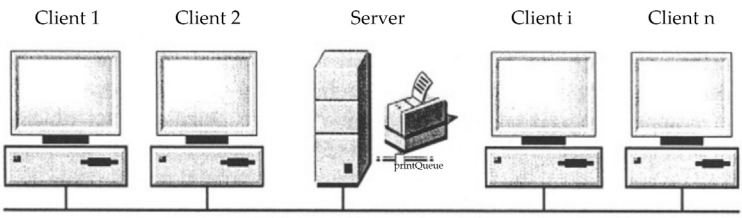


Figure 2: Printing service (a service example).

Processes can act as either clients or servers, depending on the context. A file server that receives a request to read a file from a user's client process must check on the access rights of this user. For this purpose the file server sends a request to an authentication server and waits for its response. The response of the file server to the client depends on the response from the authentication server. This implies that the file server acts as a client of the authentication server. Thus, a service provided to the user by a distributed computing system based on the client-server model could require a chain of cooperating servers.

Multiple Servers

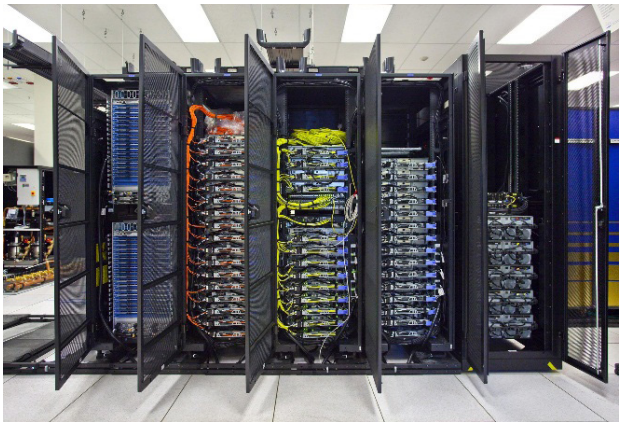
A distributed computing system has the following functions:

Improve performance through parallel execution of programs on a cluster (sometimes called network) of workstations,

Decrease response time of databases through data replication,

- Support synchronous distant meetings,
- Support cooperative workgroups, and
- Increase reliability by service multiplication, etc.

To perform these functions, many servers must contribute to the overall application. This implies a need to invoke multiple services. Furthermore, it would require in some cases simultaneous requests to be sent to a number of servers. Different applications will require different semantics for the cooperation between clients and servers, as illustrated in the following paragraphs.



Cooperation in the Systems Supporting Parallel Execution

In a distributed computing system supporting parallel execution, there are some parts of a program which could be executed as individual processes on separate computers. For this purpose a process (parent), which coordinates such parallel processing of individual processes (children), causes them to execute on selected idle computers and waits for computational results. In this case the parent process acts as a client and the child processes as servers, in a one-to-many communication pattern. However, the parent process cannot proceed any further unless all children send back responses.

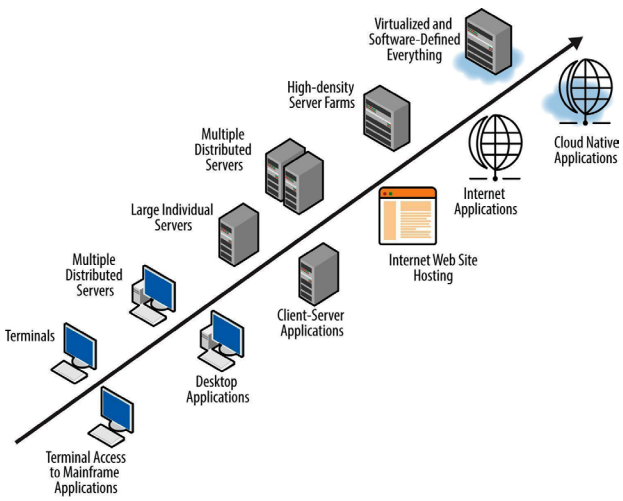
This example shows that there are two questions which should be answered in order to improve the cooperation between the client and servers: who is responsible for locating idle computers on which servers can run, and who is responsible for setting up those servers on remote computers and coordinating responses.

Cooperation in the Systems Supporting a Distributed Database

Similar semantics of cooperation between a client and multiple servers in a distributed computing system occur in supporting a distributed database. To commit a transaction all operations

performed on a set of databases must be completed successfully. Thus, a client process which executes a transaction sends operation requests to relevant databases (servers) and waits for the results of the operations. The client process can be involved in other operations, however, responses from all database servers must be received to commit the transaction.

In this case there is no need to set up servers on idle computers — there are servers which already run on dedicated computers. However, there is an issue of who can deal with these database servers, the client process or another entity working on behalf of this client.



Cooperation in the Systems Supporting a User Application

There are different semantics of cooperation between a client and multiple servers in a distributed computing system supporting a user application. This requires identifying a database server which manages relevant data, accesses that database server and carries out some operations on data, and prints that data. There are three servers which must be engaged in supporting the application: a service discovery server, a database server and a printer server. The client process running the application software invokes

each server in a sequential order and waits for a response before accessing the next server.

In this case there is again no need to set up servers on idle computers. However, the same issue exists, i.e., who can deal with these database servers, the client process or another entity working on behalf of this client.

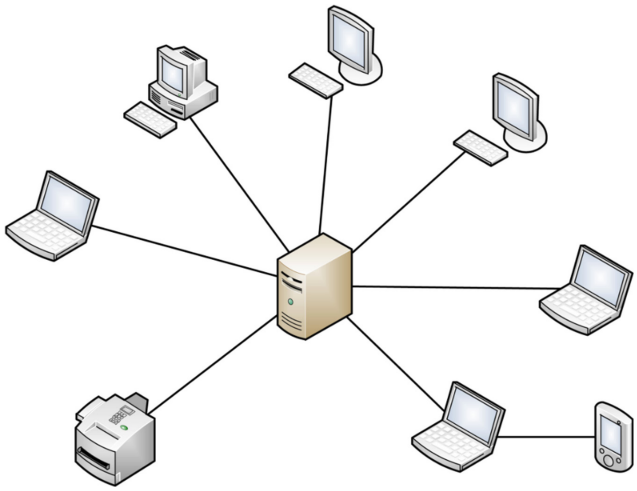
Cooperation in the Systems Supporting Mission Critical Applications

A different form of cooperation between a client and multiple servers is required in reliable distributed computing systems, in particular those which support mission critical applications. In this case a request sent to a group of servers should generate identical responses from all of them. An example of such a system is a redundant computational server on board a space ship, or a fault-tolerant transaction oriented database. In any case only identical operation results are accepted by the client.

These examples show that distributed computing systems have moved from the basic one-to-one client-server model to the one-to-many and chain models in order to improve performance and reliability. Furthermore, the issues identified when discussing the one-to-many communication pattern of the client-server model demonstrate that client and servers cooperation can be strongly influenced and supported by some active entities which are extensions to the client-server model.

4.3 EXTENSIONS TO THE CLIENT-SERVER MODEL

The need for extending the client-server model can be specified as an outcome of a study into involvement of other entities in the provision of services, an interface between a client and server, and the behavior of a client after sending of a request to a server.



Agents and Indirect Client-Server Cooperation

A client and server can cooperate either directly or indirectly. In the former case there is no additional entity that participates in exchanging requests and responses between a client and a server. Indirect cooperation in the client-server model requires two additional entities, called agents, to request a service and to be provided with the requested service. Figure 3 shows such an extension.

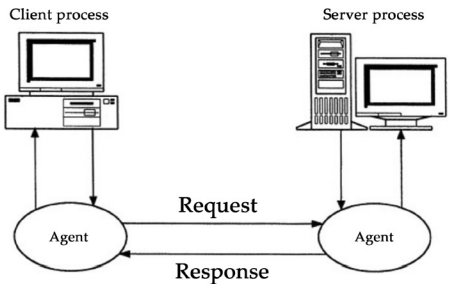


Figure 3: Indirect client-server cooperation.

The role of these agents can vary from a simple communication module which hides communication network details to an entity which is involved in mediating between clients and servers, resolving heterogeneity issues, and managing resources and cooperating

servers. A client can invoke desired servers explicitly by sending direct requests to multiple servers. In this case the programmer of a user application must concentrate both on an application and on managing server cooperation and communication. Writing resource management and communication software is expensive, time consuming and error prone. The interface between the client and the server is complicated, differs from one application to another, and the whole service provided is not transparent to the client process (user).

Clients can also request multiple services implicitly. This requires the client to send only one request to a general server. A requested service will be composed by this invoked server cooperating with other servers, based on information provided in the request. After completion of necessary operations by involved servers, the invoked server sends a response back to the client. This coordination operation can be performed by a properly designed agent. Despite the fact that such an agent is quite complicated, the cooperation between the client and the server is based on a single, well-defined interface. Furthermore, transparency is provided to the client which reduces the complexity of the application.

Cooperation between a client and multiple servers can be supported by a simple communication system which employs a direct, one-to-one message protocol. Although this communication model is simple, its performance is poor because each server involved must be invoked by sending a separate message. The overall performance of a communication system supporting message delivery in a client-server based distributed computing system can be dramatically improved if a one-to-many communication pattern is used. In this case a single request is sent by the client process to all servers, specified by a single group name. The use of multicast at the physical/data link layer does improve this system, but it is not essential.

4.3.1 The Three-Tier Client-Server Architecture

Agents and servers acting as clients can generate different architectures of distributed computing systems. The three-tier client-server architecture extends the basic client-server model by adding a middle tier to support the application logic and common services. In this architecture, a distributed application consists of the following three types of components:

User interface and presentation processing: These components are responsible for accepting inputs and presenting the results. They belong to the client tier;

Computational function processing: These components are responsible for providing transparent, reliable, secure, and efficient distributed computing. They are also responsible for performing necessary processing to solve a particular application problem. We say these components belong to the application tier;

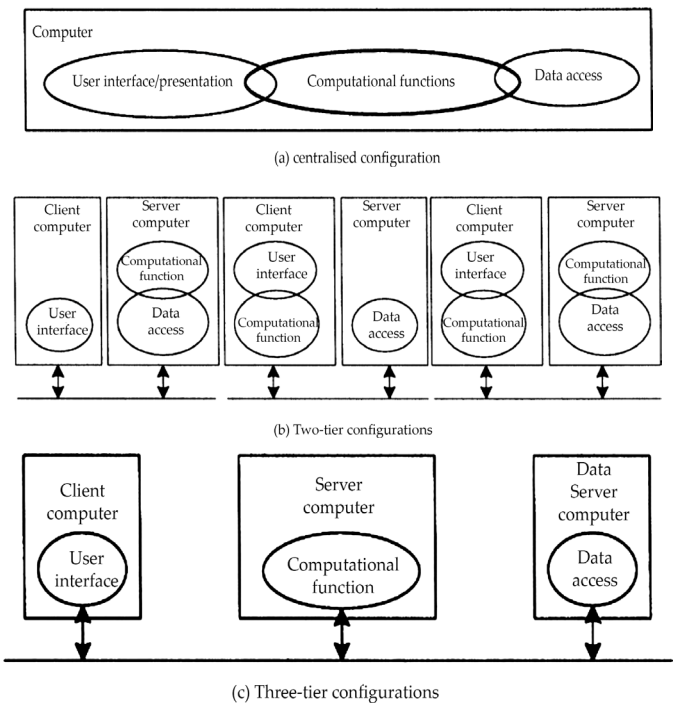


Figure 4: Examples of three-tier configurations.

Data access processing: These components are responsible for accessing data stored on external storage devices (such as disk drives). They belong to the back-end tier.

These components can be combined and distributed in various ways to create different configurations with varying complexity. Figure 4 shows some examples of such configurations ranging from centralized processing to three-tier distribution. In particular, Figure 4(a) shows a centralized configuration where all the three types of components are located in a single computer. Figure 4(b) shows three two-tier configurations where the three types of components are distributed on two computers. Figure 4(c) shows a three-tier configuration where all the three types of components are distributed on different computers.

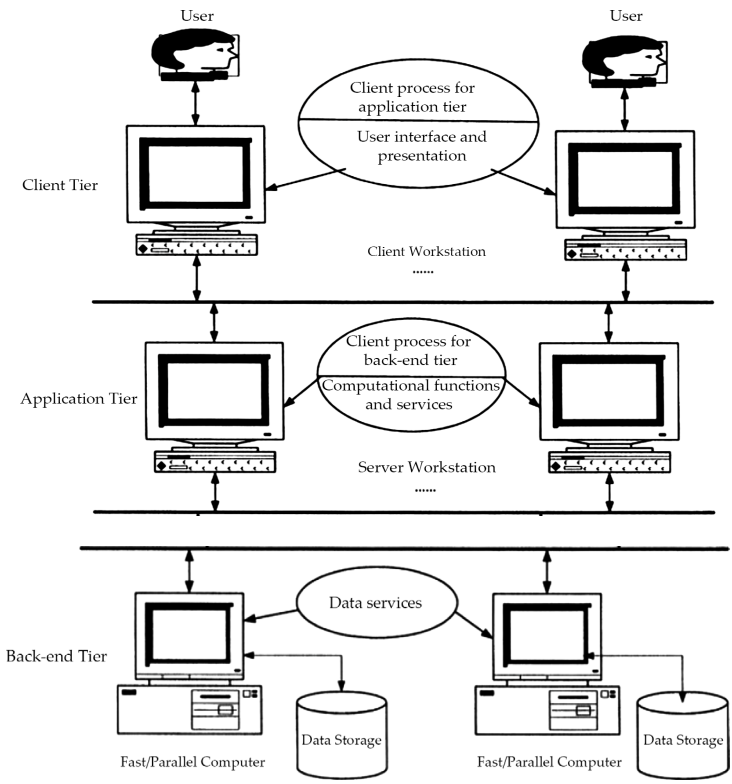


Figure 5: An example implementation of the three-tier architecture.

Figure 5 illustrates an example of implementation of the three-tier architecture. In this example, the upper tier consists of client computers that run user interface processing software. The middle tier contains computers that run computational function processing software. The bottom tier includes back-end data servers. In a three-tier client-server architecture, application clients usually do not interact directly with the data servers, instead, they interact with the middle tier servers to obtain services. The middle tier servers will then either fulfil the requests themselves, sending the result back to the clients, or more commonly, if additional resources are required, servers in the middle tier will act (as clients themselves) on behalf of the application clients to interact with the data servers in the bottom tier or other servers within the middle tier.

Compared with a normal two-tier client-server architecture, the three-tier client-server architecture has the following two important advantages:

Better Transparency: The servers within the application tier of the three-tier architecture allow an application to detach user interface from back-end resources and therefore provide better location and migration transparency. That is, the location or implementation of back-end resources can be changed without affecting the programs within the client tier;

Better scalability: The centralized and two-tier architectures do not scale well to support large applications. The servers within the application tier of the three tier architecture, however, inject another dimension of scalability into the client-server environment.

Other benefits that the three-tier client-server architecture may achieve include better concurrency, flexibility, reusability, load balancing, and reliability.

In addition to providing services for business logic and applications, the application tier should provide the following key services (they sometimes are called middleware):

Directory services: These services are required to locate application services and resources and route messages there.

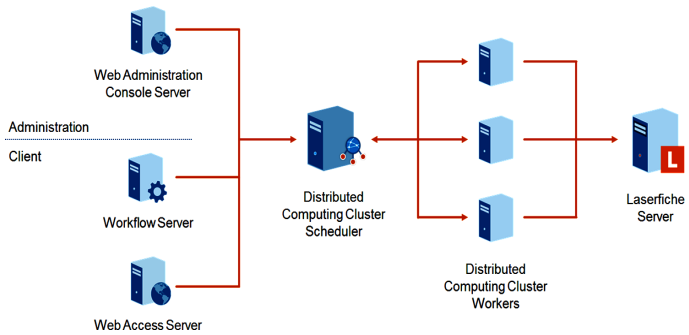
Security services: An integrated security service is needed to provide a comprehensive inter-application client-server security mechanism.

Time services: These services provide a universal format for representing time on different platforms running in different countries in various time zones. This is critical in maintaining error logs and timestamps, and in keeping synchronization among application processes.

Transaction services: These services provide transaction semantics to support commit, rollback, and recovery mechanisms. These mechanisms are critical to ensure that updates across one or more databases are handled correctly and that data integrity is not jeopardised.

4.3.2 Service Discovery

To invoke a desired service a client must know whether there is a server which is able to provide this service and its characteristics if it exists, and its name and location. This is the issue of service discovery. In the case of a simple distributed computing system, where there are only a few servers, there is no need to identify an existence of a desired server. Information about all available servers is a priori. This implies that service discovery is restricted to locating the server, which provides the desired service. On the other hand, in a large distributed computing system which is a federation of a set of distributed computing systems, with many service providers who offer and withdraw these services dynamically, there is a need to learn both whether a proper service (e.g., a very fast color printer of high quality) is available at a given time, and if so its name and location.



It is worth mentioning that a client in a distributed computing system managed by a distributed operating system, which provides transparency (one of the most important features of a distributed computing system), should only know a name of either a server or an agent working on behalf of the server. On the other hand, a client in a distributed computing system managed by a set of centralized operating systems and their extensions to access remote resources and services must know both its name and location. The reason is that transparency is not provided.

Service discovery is achieved through the following modes:

- Server computer address is hardwired into client code;
- Broadcast is used to locate servers;
- Name server is used to locate services; and
- Brokers are used to locate servers.

4.3.3 Hardwiring Computer Address

This approach requires only a location of the server, in the form of computer address, to be provided. However, it is only applicable in very small and simple systems, where there is only one server process running on the destination computer. Thus, an operating system knows where to deliver an incoming request.

Another version of this approach is based on a much more advanced naming system, where requests are sent to processes rather than to computers. In this case each process is named by a pair . < computer_address, process_name >. A client is provided with not only a name of a server, but also with the address of a server computer. This solution is not location transparent as the user is aware of the location of the server. The lack of transparency can create a problem when there is a need to move a server to another computer, and a pair < computer_address, process_name > has been hardwired in client code.

4.3.4 Broadcast Approach

According to this approach each process has a unique name (e.g., a very long identifier can be used for this purpose). In order to send a request a client knows a name of a destination, in particular of a server. However this is not enough because an operating system of a computer where the server runs must know an address of the server's computer. For this purpose the client's operating system broadcasts a special locate request containing the name of the server, which will be received by all computers on a network. An operating system that finds the server's name in the list of its processes, which means that the named server runs on its computer, sends back a 'here I am' response containing its address (location). The client's operating system receives the response and can store (cache) the server's computer address for future communication. This approach is transparent, however the broadcast overhead is high as all computers on a network are involved in the processing of the locate request.

The cooperation between clients, servers and operating systems supporting them in a distributed computing system using the broadcast approach to locate servers is illustrated in Figure 6.

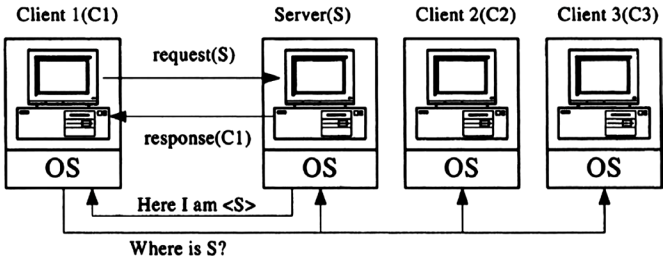
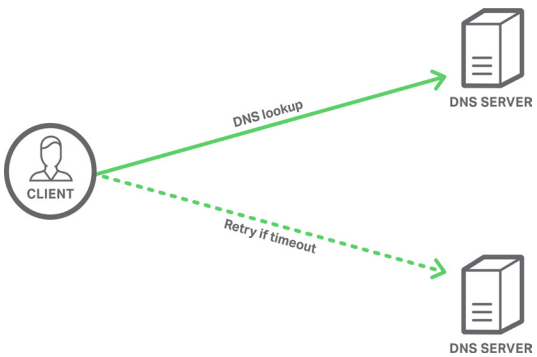


Figure 6: Service discovery -- broadcast approach.

4.3.5 Name Server Approach

This approach is very similar to the broadcast based approach, however it reduces the broadcast overhead. In order to learn the address of a desired server, an operating system of the client's computer sends a 'where is' request to a special system server, called a name server, asking for the address of a computer where the desired server runs. This means that the name and location (computer address) of the name server are known to all operating systems. The name server sends back a response containing an address of the desired server. The client's operating system receives the response and can cache the server's computer address for future communication.



This approach is transparent and much more efficient than the broadcast based approach. However, because the name server is centralized, the overall performance of a distributed computing system could be degraded, as the name server could be a bottleneck.

Furthermore, reliability of this approach is low; if a name server computer crashes, a distributed computing system cannot work.

Figure 7 illustrates the cooperation between clients, servers and operating systems supporting them in a distributed computing system using the approach of server

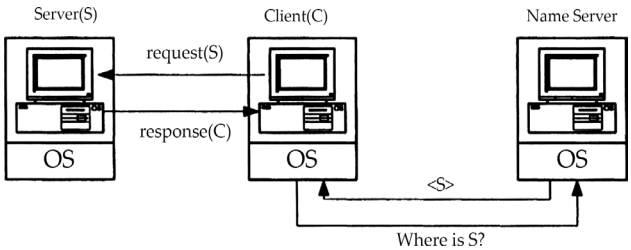


Figure 7: Service discovery -- name server and server location lookup.

4.3.6 Broker-Based Location Lookup

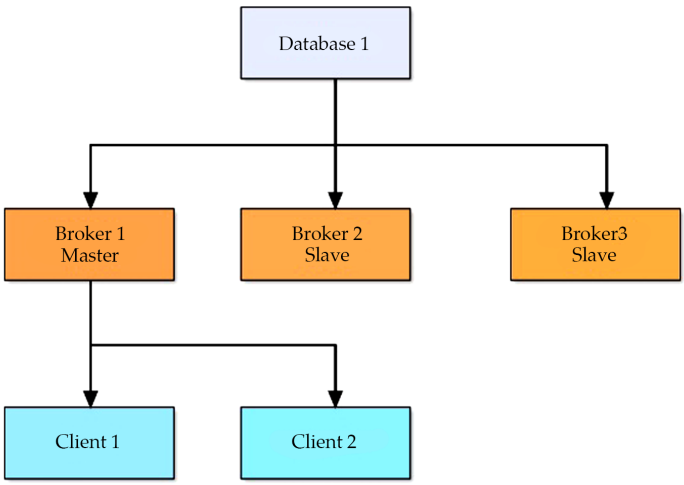
A client in a distributed computing system supported by any of the above server location approaches must know all servers available in that system and their names (also the server locations (addresses) in systems which do not provide transparency) in order to invoke a desired service. In a large distributed computing system there could be a large number of servers. Moreover, servers of the same type can be characterized by different attributes describing the services they provide (e.g., one laser printer is a color printer, another is a black and white printer). Furthermore, servers can be offered by some users and revoked dynamically. A user is not able to know names and attributes of all these servers, and their dynamically changing availability. There must be a server which could support users to deal with these problems. This server is called a broker. Thus, a broker is a server that

- allows a client to identify available servers which can be characterized by a set of attributes that describe the properties of a desired service;
- mediates cooperation between clients and servers;
- allows service providers to register the services they

support by providing their names, locations and features in the form of attributes;

- advertises registered services and makes them available to clients; and
- withdraws services dynamically.

A broker-based approach is very similar to the server location lookup performed via a name server approach. However, there are real conceptual differences between a broker and a name server which frees clients from remembering ASCII names or path names of all servers (and eventually the server locations), and allows clients to identify attributes of servers and learn about their availability. Thus, a broker is a server, which embodies both service management and naming services. There are two basic broker classes, which form two different forms of cooperation between clients and servers:



Forwarding Broker

Cooperation between a client and a server mediated by this broker is as follows:

Step 1: the broker receives from a client a service enquiry in a form of a set of attributes that characterize a desired service, and a server operation request;

Step 2: if a matching server is available, the broker sends the server operation request to that found server; otherwise, it sends a failure response to the client;

Step 3: the server sends back a response to the broker;

Step 4: the broker passes the response to the client.

The forwarding broker possesses advantages and disadvantages of a system with a centralized server. This means that all requests to servers and their responses are going through this broker.

Direct Broker

Cooperation between a client and a server mediated by this broker is as follows:

Step 1: the broker receives from a client a service enquiry in a form of a set of attributes that characterize a desired service;

Step 2: if a matching server is available, the broker sends back a name and a server computer address to the client; otherwise, it sends a failure response;

Step 3: the client sends the server operation request to the server;

Step 4: the server sends back a response to the client.

Despite the fact that the direct broker also possesses advantages and disadvantages of a system with a centralized server, its performance is better than that of the forwarding broker, because only service enquiry messages are sent to this broker.

4.4 DISTRIBUTED SYSTEM

Distributed networking is a distributed computing network system where components of the program and data depend on multiple sources. Distributed networking, used in distributed computing, is the network system over which computer

programming, software, and its data are spread out across more than one computer, but communicate complex messages through their nodes (computers), and are dependent upon each other. The goal of a distributed network is to share resources, typically to accomplish a single or similar goal. Usually, this takes place over a computer network, however, internet-based computing is rising in popularity. Typically, a distributed networking system is composed of processes, threads, agents, and distributed objects. Merely distributed physical components is not enough to suffice as a distributed network; typically distributed networking uses concurrent program execution. For example, Distributed systems have endless use cases, a few being electronic banking systems, massive multiplayer online games, and sensor networks. StackPath utilizes a particularly large distributed system to power its content delivery network service. Every one of our points of presence (PoPs) has nodes that form a worldwide distributed system. And to provide top notch content delivery, StackPath stores the most recently and frequently requested content in edge locations closest to the location it is being used.

4.4.1 Types of Distributed Systems

A distributed network is a type of computer network that is spread over different networks. This provides a single data communication network, which can be managed jointly or separately by each network. Besides shared communication within the network, a distributed network often also distributes processing.

Distributed networks are part of distributed computing architecture, in which enterprise IT infrastructure resources are divided over a number of networks, processors and intermediary devices. A distributed network is powered by network management software, which manages and monitors data routing, combining and allocating network bandwidth, access control and other core networking processes.

Distributed networks and processing work together to deliver specialized applications to different remote users. This means

that an application may be hosted and executed from a single machine but accessed by many others. A client/server computing architecture is an example of a distributed network where the server is the producer of a resource and many interconnected remote users are the consumers who access the application from different networks.

Distributed systems generally fall into one of four different basic architecture models:

Client/server

Client/server computing is a type of distributed computing where one computer, a client, requests data from the server, a primary computing center, which responds to the client directly with the requested data, sometimes through an agent. Client/server distributed networking is also popular in web-based computing. Client/Server is the principle that a client computer can provide certain capabilities for a user and request others from other computers that provide services for the clients. The Web's Hypertext Transfer Protocol is basically all client/server.

- **Three-tier**—Information about the client is stored in a middle tier rather than on the client to simplify application deployment. This architecture model is most common for web applications.
- ***n*-tier**—Generally used when an application or server needs to forward requests to additional enterprise services on the network.

Agent-based

A distributed network can also be agent-based, where what controls the agent or component is loosely defined, and the components can have either pre-configured or dynamic settings.

Decentralized

Decentralization is where each computer on the network can be used for the computing task at hand, which is the opposite of the client/server model. Typically, only idle computers are used, and in this way, it is thought that networks are more efficient. Peer-to-peer (P2P) computation is based on a decentralized, distributed network, including the distributed ledger technology blockchain.

Peer-to-peer

There are no additional machines used to provide services or manage resources. Responsibilities are uniformly distributed among machines in the system, known as peers, which can serve as either client or server.

Mesh

Mesh networking is a local network composed of devices (nodes) that was originally designed to communicate through radio waves, allowing for different types of devices. Each node is able to communicate with every other node on the network.

Cloud computing

Enterprises with rapid growth and scaling needs may find it challenging to maintain their own distributed network under the traditional client/server computing model. Cloud Computing is the utility of distributed computing over Internet-based applications, storage, and computing services. A cloud is a cluster of computers or servers that are closely connected to provide scalable, high-capacity computing or related tasks.

4.4.2 Advantages of Distributed Networking

Prior to the 1980's, computing was typically centralized on a single low-cost desktop computer. But today, computing resources (computers or servers) are typically physically distributed in many places, which distributed networking excels at. Some types of computing does not scale well past a certain level of parallelism and the gains of superior hardware components, and thus is bottle-necked, such as by Very Large Scale Instruction Words. By increasing the number of computers rather than the power of their components, these bottlenecks are overcome. Situations where resource sharing becomes an issue, or where higher fault tolerance is needed also find aid in distributed networking. Distributed networking is also very supportive of higher levels of anonymity.

Some advantages of Distributed Systems are as follows −

- All the nodes in the distributed system are connected to each other. So nodes can easily share data with other nodes.
- More nodes can easily be added to the distributed system i.e. it can be scaled as required.
- Failure of one node does not lead to the failure of the entire distributed system. Other nodes can still communicate with each other.
- Resources like printers can be shared with multiple nodes rather than being restricted to just one.

4.4.3 Disadvantages of Distributed Systems

Some disadvantages of Distributed Systems are as follows:−

- It is difficult to provide adequate security in distributed systems because the nodes as well as the connections need to be secured.
- Some messages and data can be lost in the network while moving from one node to another.

- The database connected to the distributed systems is quite complicated and difficult to handle as compared to a single user system.
- Overloading may occur in the network if all the nodes of the distributed system try to send data at once.

4.4.4 Design Issues with Distributed Systems

While there are many benefits to distributed systems, it's also important to note the design issues that can arise. We've summarized the main design considerations below.

- ***Failure Handling*:** Failure handling can be difficult with distributed systems because some components fail while others continue to function. This can often serve as an advantage to prevent large-scale failures, but it also lead to more complexity when it comes to troubleshooting and debugging.
- ***Concurrency*:** A common issue occurs when several clients attempt to access a shared resource simultaneously. You must ensure that all resources are safe in a concurrent environment.
- ***Security issues*:** Data security and sharing have increased risks in distributed computer systems. The network has to be secured, and users must be able to safely access replicated data across multiple locations.
- ***Higher initial infrastructure costs*:** The initial deployment cost of a distributed system can be higher than a single system. This pricing includes basic network setup issues, such as transmission, high load, and loss of information.

Distributed systems aren't easy to get up and running, and often this powerful technology is too *"overkill"* for many systems. There are many challenges distributing data that ensures various requirements under unexpected circumstances.

Similarly, bugs are harder to detect in systems that are spread across multiple locations.

Cloud vs. distributed systems

Cloud computing and distributed systems are different, but they use similar concepts. Distributed computing uses distributed systems by spreading tasks across many machines. Cloud computing, on the other hand, uses network hosted servers for storage, process, data management.

Distributed computing aims to create collaborative resource sharing and provide size and geographical scalability. Cloud computing is about delivering an on demand environment using transparency, monitoring, and security.

Compared to distributed systems, cloud computing offers the following advantages**:**

- Cost effective
- Access to a global market
- Encapsulated change management
- Access storage, servers, and databases on the internet

However, cloud computing is arguably less flexible than distributed computing, as you rely on other services and technologies to build a system. This gives you less control overall.

Priorities like load-balancing, replication, auto-scaling, and automated back-ups can be made easy with cloud computing. Cloud building tools like Docker, Amazon Web Services (AWS), Google Cloud Services, or Azure make it possible to create such systems quickly, and many teams opt to build distributed systems alongside these technologies.

Examples of Distributed Systems

Distributed systems are used in all kinds of things, everything from electronic banking systems to sensor networks to multiplayer online games. Many organizations utilize distributed systems to power content delivery network services.

In the healthcare industry, distributed systems are being used for storing and accessing and telemedicine. In finance and commerce, many online shopping sites use distributed systems for online payments or information dissemination systems in financial trading.

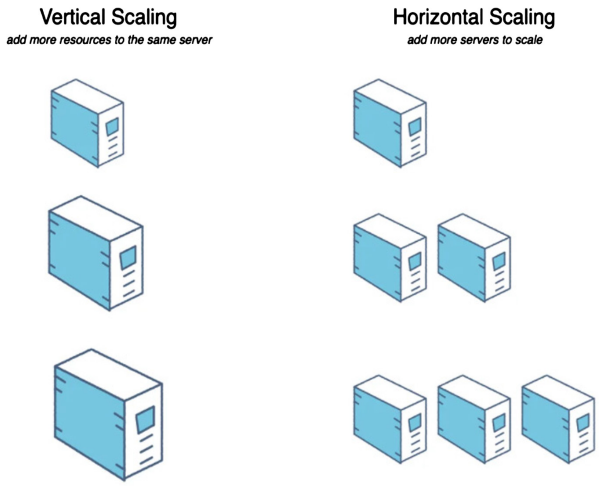
Distributed systems are also used for transport in technologies like GPS, route finding systems, and traffic management systems. Cellular networks are also examples of distributed network systems due to their base station.

Google utilizes a complex, sophisticated distributed system infrastructure for its search capabilities. Some say it is the most complex distributed system out there currently.

4.4.5 Benefits and Challenges of Distributed Systems

Distributed systems can be challenging to deploy and maintain, but there are many benefits to this design. Let's go over a few of those perks.

- ***Scaling*:** A distributed system allows you to scale horizontally so you can account for more traffic.
- ***Modular growth*:** There is almost no cap on how much you can scale.
- ***Fault tolerance*:** Distributed systems are more fault tolerant than a single machine.
- ***Cost effective*:** The initial cost is higher than a traditional system, but because of their scalability, they quickly become more cost effective
- ***Low latency*:** Users can have a node in multiple locations, so traffic will hit the closet node
- ***Efficiency*:** Distributed systems break complex data into smaller pieces
- ***Parallelism*:** Distributed systems can be designed for parallelism, where multiple processors divide up a complex problem into pieces



There are three reasons that teams generally decide to implement distributed systems:

- ***Horizontal Scalability***—Since computing happens independently on each node, it is easy and generally inexpensive to add additional nodes and functionality as necessary.
- ***Reliability***—Most distributed systems are fault-tolerant as they can be made up of hundreds of nodes that work together. The system generally doesn't experience any disruptions if a single machine fails.
- ***Performance***—Distributed systems are extremely efficient because workloads can be broken up and sent to multiple machines.

However, distributed systems are not without challenges. Complex architectural design, construction, and debugging processes that are required to create an effective distributed system can be overwhelming.

Three more challenges you may encounter include:

- ***Scheduling***—A distributed system has to decide which jobs need to run, when they should run, and where they should run. Schedulers ultimately have limitations,

leading to underutilized hardware and unpredictable runtimes.

- ***Latency***—The more widely your system is distributed, the more latency you can experience with communications. This often leads to teams making tradeoffs between availability, consistency, and latency.
- ***Observability***—Gathering, processing, presenting, and monitoring hardware usage metrics for large clusters is a significant challenge.

4.4.6 Distributed Application

A distributed application is a program meant to run on multiple computers at once and can be stored on a server or in **cloud computing**. This is commonly used within a network that has a client-server relationship in which a client computer accesses a program from the server and the server does all the processing. Each computer that accesses the application normally is made for a specific purpose. While there are many different distributed applications, the common ones are general programs, collaboration software, real-time systems and computational systems. Using a distributed application typically is beneficial, but this can pose a problem if the server is weak or slow.

Distributed application systems can be used on many different network types, but they are most often seen in client-server networks. In this type of network, the client computer or the computer people use accesses programs and information from the server. Not only is the application used from the server, but the server is responsible for doing all the processing for the program to work. Cloud computing also can be used for this, in which case the program is stored on a cloud server and client computers access the program.

While each computer or person may use the distributed application for general purposes, the computer or person is normally specialized or instructed to perform a specific task. For example,

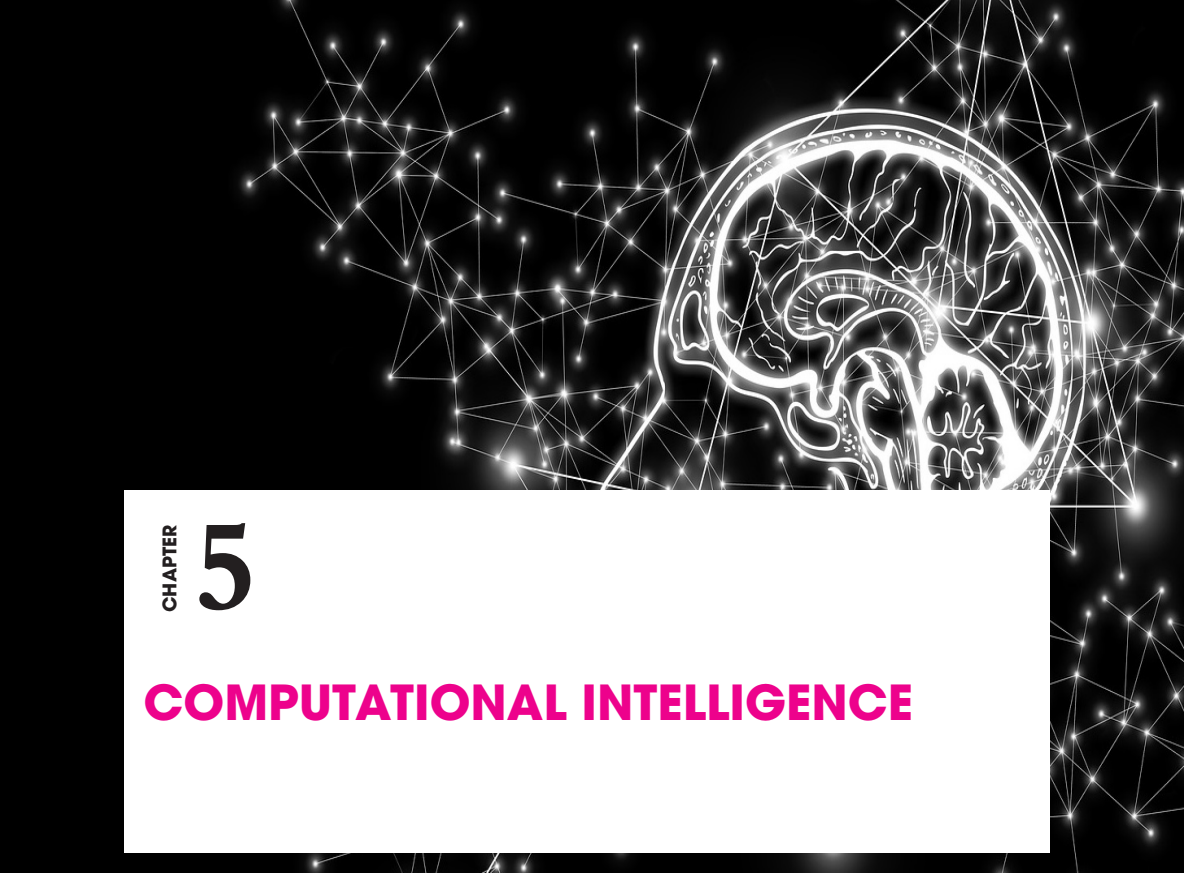
one computer may be used or optimized to create an image while another is used for text. The user also may be instructed to perform a certain task, rather than just performing generic tasks.

There are many different distributed application types, but there are four primary categories. General programs are common programs found on a computer, just they are made to work on several computers at once. Collaboration software is made so several people can work on a single project at the same time; each user typically works on a different section of the project. Real-time systems are chat-and-ticket programs in which users answer customer questions online. Computational programs are made to process code, and using several computers at once optimizes and speeds up processing.

Using a distributed application is generally beneficial, because it makes it easier for many users to access and use a program at once, though there can be a problem if the server is not strong. The server is commonly responsible for performing the processing, so this puts a lot of strain on the server. If the server is weak or slow, then this can lead to lagging or more serious problems on computers accessing the program.

REFERENCES

1. Pelkey, James L. (2007). "6.9 – Metcalfe Joins the Systems Development Division of Xerox 1975-1978". Entrepreneurial Capitalism and Innovation: A History of Computer Communications, 1968-1988. Retrieved 2019-09-05.
2. Piliouras, Teresa C. (Dec 2004). Network Design, Second Edition: Management and Technical Perspectives (2nd ed.). CRC Press. Retrieved 25 September 2018.
3. Simmonds, A; Sandilands, P; van Ekert, L (2004). An Ontology for Network Security Attack. Lecture Notes in Computer Science. 3285.
4. Srinivasa, K.G.; Muppalla, Anil Kumar (Feb 2015). Guide to High Performance Distributed Computing: Case Studies with Hadoop, Scalding and Spark (Computer Communications and Networks). Springer.
5. Tsenov, Martin (June 2007). "Example of communication between distributed network systems using web services". CompSysTech '07 Proceedings of the 2007 International Conference on Computer Systems and Technologies (35): 1.
6. Weija, Jia; Zhou, Wanlei (Dec 2004). Distributed Network Systems: From Concepts to Implementations. Springer.



CHAPTER 5

COMPUTATIONAL INTELLIGENCE

INTRODUCTION

Computational Intelligence (CI) is the theory, design, application and development of biologically and linguistically motivated computational paradigms. Traditionally the three main pillars of CI have been Neural Networks, Fuzzy Systems and Evolutionary Computation. However, in time many nature inspired computing paradigms have evolved. Thus CI is an evolving field and at present in addition to the three main constituents, it encompasses computing paradigms like ambient intelligence, artificial life, cultural learning, artificial endocrine networks, social reasoning, and artificial hormone networks. CI plays a major role in developing successful intelligent systems, including games and cognitive developmental systems. Over the last few years there has been an explosion of research on Deep Learning, in particular

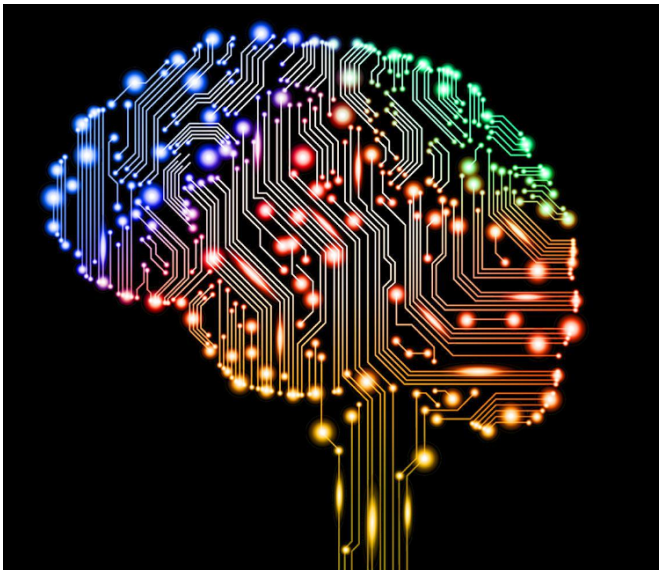
deep convolutional neural networks. Nowadays, deep learning has become the core method for artificial intelligence. In fact, some of the most successful AI systems are based on CI.

The term of CI has been widely used for quite a long time with steadily increasing amount of research and applications, there is no anonymously accepted definition. CI can mean many things to different people and various techniques are considered as belonging to CI. Behind the approaches could be seen different communities, traditionally applying various methodologies and criteria of their evaluation and having some lack of understanding and sometimes even some misunderstanding of each other methods. To make things even more complicated, there exist a lot of variations even within one approach, and different concepts may have the same name in different reports, while the same things could be called distinctly by different researchers. Such a situation can be partly explained by the two following reasons: CI is a shifting phenomenon, changing its study domain in time but staying young forever, on one hand, and CI is a distributed phenomenon, the algorithmic study of processes in every field of enquiry. All of these features impose an extra burden on any attempt to describe the field of CI. This attempts to present and describe a diversity of views, however, giving interpretations, which are considered to be the most popular among the research

community at the present time and the most appropriate for this publication. Note that none of the views considered below is meant to be particularly controversial. Rather, they complement each other in some aspects. The standard way how to distinguish different theories and views is by naming the constituents of a particular methodologies. This way is explored in the following text as well.

5.1 DEFINITION AND UNDERSTANDING OF COMPUTATIONAL INTELLIGENCE

The first view of CI considers it as a replacement term for artificial intelligence (AI) as well. In this very wide interpretation, CI represents the branch of science and engineering concerned with making computers behave like humans. The term of AI was coined in 1956 by J. McCarthy at the Massachusetts Institute of Technology (USA). Since then two complementary views of AI have been developed: one as an engineering discipline dealing with the creation of intelligent machines, the other as an empirical science concerned with the computational modeling of human intelligence. The first branch is a part of computer science and engineering, aimed at programming computers to behave intelligently. The second branch is a part of cognitive science, aimed at simulating with computers the actual processes applied by humans in their intelligent behavior. Within this domain the computer is a tool for modeling theories of human intelligence. When the field was young, these two views were seldom distinguished. However, by the present time a substantial divide has opened up, with the former view dominating modern AI and the latter view characterizing much of modern cognitive science. For this reason the more neutral term of "computational intelligence" has been adopted by many researchers, as both communities are attacking the problem of understanding intelligence in computational terms.



The field of AI (or CI in a wide interpretation as we will apply it in this section) is concerned both with modeling human (in many references it is called natural or biological) intelligence and with solving complex problems not solvable by simple or analytical procedures. For instance, a major, long-range goal of CI is the construction of an intelligent robot, one capable of perceiving, acting, comprehending, reasoning, and learning in complex environments. Among other goals one can find the following:

- ***Automatic Programming*** - the task of describing what a program should do and having the CI system 'write' the program.
- ***Natural Language Processing (NLP)*** - processing and understanding human ("natural") language, also known as computational linguistics.
- ***Knowledge Engineering/Representation*** - turning what we know about a particular domain into a form in which a computer can understand it - see Cybernetics and the Integration of Knowledge.
- ***Planning*** - given a set of actions, a goal state, and a present state, deciding which actions must be taken so that the present state is turned into the goal state.

- ***Search*** - the finding of a path from a start state to a goal state (similar to planning, yet different)
- ***Machine Learning -*** creating of systems that are able to learn from their experience.
- ***Visual Pattern Recognition*** - developing the ability to reproduce the human sense of sight automatically. • Speech Recognition - conversing of speech into text.

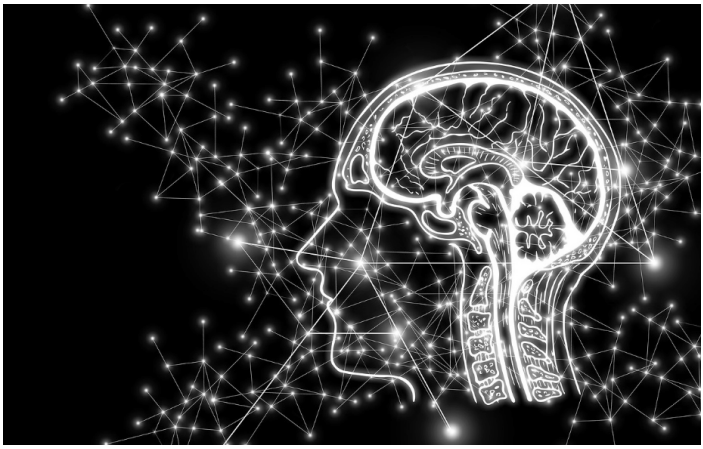
The great variety of CI techniques have been developed and applied over the history for solving the problems mentioned above. Some of these methodologies are:

- ***Automatic Reasoning*** - most human reasoning occurs in task/domains with uncertain, ill-defined and incomplete knowledge. Reasoning in such domains requires techniques such as use of default, probabilistic, and non-monotonic logics.
- ***Bayesian Networks*** - a technique of structuring and interferencing with probabilistic information.
- ***Neural Networks (NN) or Artificial Neural Networks (ANN)*** - the study of programs that function in a manner similar to how animal brains do. The two terms NN and ANN are frequently used interchangeably. They simulate a highly interconnected, parallel computational structure with many relatively simple individual processing elements. - see. Neural Networks for more detail
- ***Fuzzy Logic (FL)*** - is the logic of approximate reasoning. Fuzzy logic comprises operations on fuzzy sets including equality, containment, complementation, intersection and union, and is a generalization of conventional (two-valued, or "crisp") logic - Evolutionary Computation comprises machine learning optimization and classification paradigms based on mechanisms of evolution such as biological genetics and natural selection. The evolutionary computation field includes genetic algorithms, evolutionary programming, genetic programming, and evolution strategies. Most widely used in applications are genetic algorithms. Genetic algorithms

are search algorithms that incorporate natural evolution mechanisms including crossover, mutation and survival of the fittest. GA paradigms work on populations of individuals, rather than on single data points or vectors. They are more often used for optimisation, but also for classification. Evolutionary programming algorithms are similar to genetic algorithms, but do not incorporate crossover. Rather, they rely on survival of the fittest and mutation.

- ***Problem solving and search*** - a fundamental technique in CI is to encode a problem as a state space in which solutions are goal states in that space. Thus, problem solving can be viewed as state space search. To search large, combinatorial state spaces, knowledge (e.g. heuristics) and planning are required.
- ***Simulated Annealing (SA)*** - a technique which can be applied to any minimization or learning process based on successive update steps (either random or deterministic) where the update step length is proportional to an arbitrarily set parameter which can play the role of a temperature. Then, in analogy with the annealing of metals, the temperature is made high in the early stages of the process for faster minimization or learning, then is reduced for greater stability - see Simulated Annealing.

CI problems (speech recognition, NLP, vision, automatic programming, knowledge representation, etc.) can be paired with techniques (NN, search, Bayesian nets, production systems, etc.) to make distinctions such as search-based NLP vs. NN NLP vs. Statistical /Probabilistic NLP. Then one can combine techniques, such as using neural networks to guide search. And one can combine problems, such as posing that knowledge representation and language equivalent.



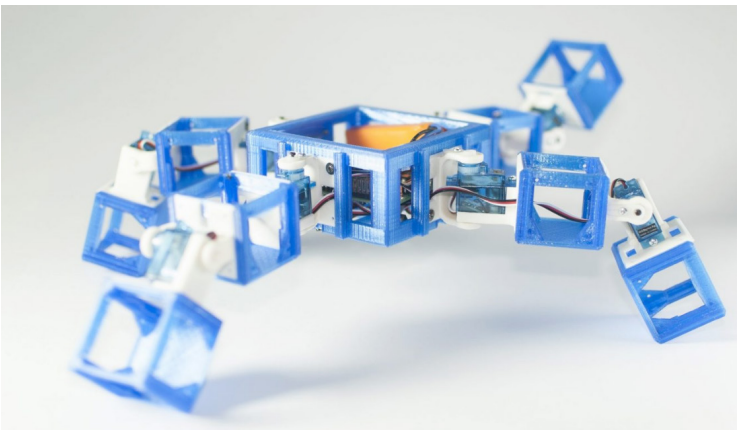
5.1.1 Goals of Computational Intelligence and their Accomplishment to date

Speaking in very general terms, the goal of CI is developing methods, which allow producing thinking machines that can solve problems. One may note that this statement does not limit the choice of those problems by any way. On the contrary, complex problems, which are not typically attempted by computers, should be considered first of all. In 1958 Newell and Simon predicted that computers would by 1970 be capable of composing classical music, discovering important new mathematical theorems, playing chess at grandmaster level, and understanding and translating spoken language. Although these predictions were overly optimistic, they did represent a set of focused goals for the field of CI. What has been achieved so far? Quite a bit of progress has been made, that can be illustrated with the following examples:

- Deployed speech dialog systems by firms like IBM, Dragon and Lernout&Hauspie
- Applications of expert systems/case-based reasoning, for example a computerized Leukemia diagnosis system did a better job checking for blood disorders than human experts.

- Computer translation for Environment Canada: software developed in the 1970s translated natural language weather forecasts between English and French.
- Deep Blue, the first computer system, which has beaten the human chess champion G.Kasparov.
- Intelligent (employing fuzzy, neural and their combinations techniques) systems to control the whole range of the household devices available on the market.

One persistent 'problem' in evaluating the achievements of CI is that often as soon as a CI technique truly succeeds, in the minds of many it ceases to be AI, becoming something else entirely. For example, when Deep Blue defeated Kasparov, there were many who said Deep Blue was not CI, since after all it was just a brute force parallel minimax search. It can be explained partly by the absence of a standard understanding of what intelligence is and how it should be evaluated. However, there exist some methods, the most famous is the so-called Turing test. Alan Turing, one of the founders of computer science, made the claim that by the year 2000, computers would be able to pass the Turing test at a reasonably sophisticated level. Turing test is passed if an independent observer being presented with the results of solving a complex problem is not able to judge if the solution has been achieved by a human or a computer. In particular, that the average interrogator would not be able to identify the computer correctly more than 70 per cent of the times after a five minute conversation.



5.1.2 Goals for Future Research

Such goals have to represent great challenges for CI and expect to produce amazing results when achieved. In particular, they should satisfy to the following conditions:

(a) Represent collaborative community effort: It must span several sub-fields of CI, requiring some degree of collaboration between CI researchers of different specialties. The idea is to help unify the fragmented areas with a common purpose or purposes.

(b) Has high society impact: It must address important problems of widespread interest. Solving the problem must matter to many people. This will help motivate and excite researchers, and justify the field to outsiders.

(c) Has short horizon for progress: It must be possible to have incremental progress and not be an all or nothing problem. For example, problems where we can reasonably expect to make significant measurable progress over the next 10 years or so.

(d) Drive Basic Research: It should involve more than just applying current technology, but should drive basic research and the development of new technology (possibly in completely new directions.

The examples of such problems could be found in different fields. Ones, which a CI community has already intensively been working on, are:

- Knowbots/Infobots or Web agents and intelligent help desks, which require research and development of unified NLP, information retrieval, reasoning, intelligent user interfaces.
- Autonomous Vehicles, which require research and development of unified robotics, machine vision, machine learning, intelligent control, planning.
- Machine Translation, which require research and development of unified NLP, knowledge representation, speech understanding, speech synthesis.

- Intelligent Homes and Services, which require research and development in intelligent control and information systems

5.1.3 Other Views of Computational Intelligence

The second view of CI is based on the definition of this term, put forward by J. Bezdek in 1994. He proposed the narrowest definition, in which he contrasts three ABC's of intelligence: artificial, biological, and computational. In the strictest sense, CI depends on numerical data supplied by manufacturers and does not rely on knowledge. Artificial intelligence, on the other hand, uses what Bezdek called "knowledge tidbits". Heuristically constructed AI systems, such as expert systems, is an example. His understanding of CI and relationship between components of CI systems is summarised in Figure 1, where both natural and artificial intelligence components are considered. Bezdek described

the three levels (A, B, and C) as being three very different levels of system complexity, with complexity increasing from left to right and bottom to top. In the figure, NN refers to neural networks, PR to pattern recognition and I to intelligence. CI thus refers to numeric, or computational, intelligence. The distances between the nodes are skewed to show disparities between terms they represent. Thus, for example, the distinction between computational neural networks (CNNs) and computational pattern recognition (CPR) is less than that between biological neural networks (BNNs) and biological pattern recognition (BPR).

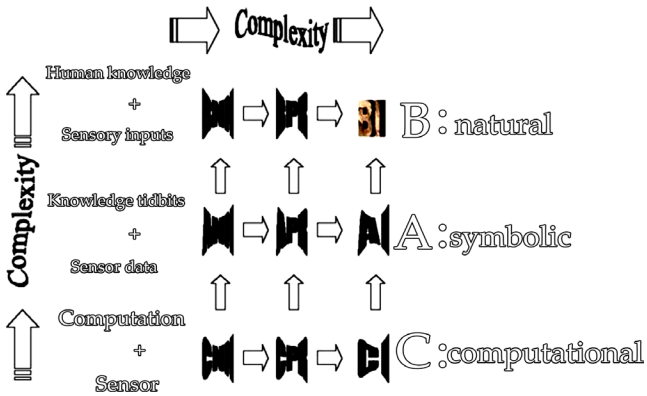


Figure 1: Definition of computational intelligence according to J. Bezdek.

In Bezdek's scheme, then, the top layer represents biological intelligence and the two lower layers represent machine intelligence. As he pointed out, the model presented in the figure is limited to systems used for pattern recognition; there are many alternatives to CNNs appropriate as inputs to CPR. Additionally, he pointed out that there are many kinds of CPR, including deterministic fuzzy and statistical models, some of which do not have biological equivalents. Finally, he suggested that the relationship between nodes at the ends of any arrow is that the node at the tai of the arrow is a subset of the node at the head of the arrow. Thus, CNN is a subset of CPR which is a subset of Cl; and Cl is a subset of AI, which is a subset of BI.

This definition was criticized by R.C. Eberhart who pointed out the following problems:

- the dichotomy of functions (nodes) along biological versus computational lines,
- the statement that some computational models do not have biological equivalents,
- the characterization of a node as a subset of subsequent nodes.

Eberhart's main point is that intelligence exists, and that intelligence is manifested in both artificial and natural systems, and sometimes in hybrids of the two. It does no matter what produces the intelligence. Further, all computational models must have biological analogies, since humans conceived, designed, developed and tested them. We can make only something, what we know. In 1994 he defined CI as a methodology involving computing that exhibits an ability to learn and/or to deal with or adapt to new situations. Such a system is perceived to possess one or more attributes of reasoning, such as generalization, discovery, association and abstraction. CI systems usually comprise hybrids of paradigms such as ANN, FL systems and evolutionary computation systems, augmented with knowledge elements. CI systems are often designed to mimic one or more aspects of the biological intelligence systems.

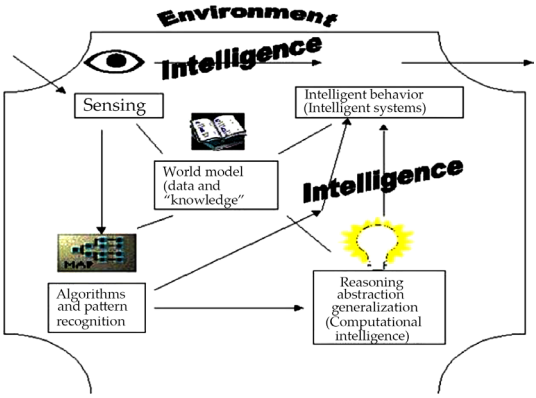


Figure 2: Definition of computational intelligence according to R.C. Eberhart.

The relationships among the components of intelligent systems are, as suggested by Eberhart, represented by Figure 2. Analogous to Bezdek's analysis, the emphasis is on pattern recognition. Other functions are needed to make the figure more complete. The inputs to the system can be sensory inputs in the case of biological systems or a computer keyboard, etc. Then outputs of the intelligent behavior nod to the environment are communications to, and actions upon, the environment. If there is no action that affects the environment, there is no intelligent behavior. One arrow goes directly from sensing to intelligent behavior; another from algorithms and pattern recognition to intelligent behavior. These are meant to represent processes, which include reflex actions related to safety and survival, such as person's actions when touching a hot stove. There may be a better way to represent such processes, but the representation is meant to emphasize that, in general, nodes at the tails of arrows are not subsets of those at the heads, and that all nodes can provide input to the intelligent behavior node.

The diagram in Figure 2 is simplistic, but it is meant to convey the Eberhart's belief that there should be no distinction between artificial and natural intelligence. A system simply possesses one or more of the attributes shown in the Figure 2, and the actions on and communications to the environment are intelligent to some degree, depending upon the system attributes. As represented, CI is buried deeply within the core of the system, be it biological or machine. It is perhaps the furthest from the interface with the environment.

5.2 COMPUTATIONAL INTELLIGENCE TECHNOLOGIES

Basically Computational intelligence techniques include artificial Intelligence (AI) techniques [artificial Neural Networks (ANNs), fuzzy Logic (FL), support Vector Machines (SVM), Self Organizing Maps (SOM) (unsupervised)] ; genetic Algorithms (GA), genetic Programming (GP) and swarm Intelligence or particles swarm optimization (PSO).

CI also makes use of ANN consisting of multi-layer Perceptron concept (MLP), radial Basis Function (RBF), and probabilistic Neural Networks (PNN). Also fuzzy Logic & ANN techniques makes use of adaptive neuro-fuzzy inference system (ANFIS).

The ANN structure consists of an input layer, hidden Layer (s), Output layer, number of nodes in each layer and functions and their parameters.

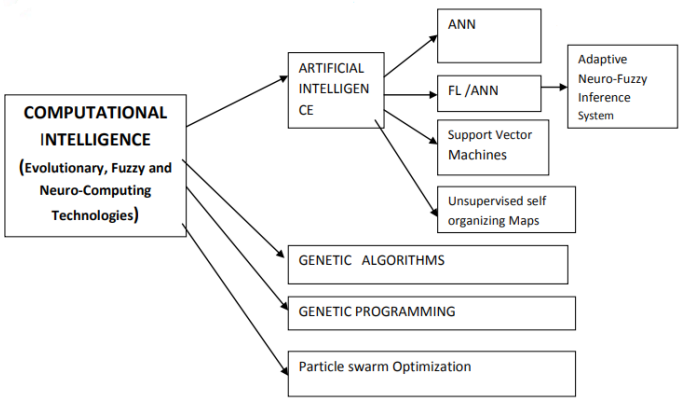
The steps of fuzzy logic approach are Fuzzification using membership functions (MFs)-input, generation of rule base, aggregation and De-fuzzification using MFs –output. The input-output of membership functions need number, type, parameters and a rule base. The Neuro-fuzzy system combines the procedures of fuzzy logic (FL) and ANNs. This actually starts with an initial FL structure. Further the Neuro fuzzy system also uses ANN for adapting the FL (MF) parameters and the rule base to the training data.

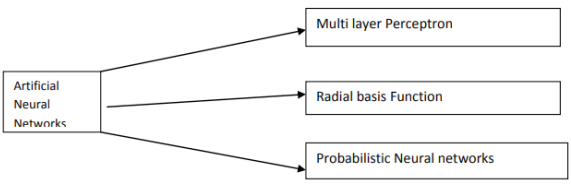
The Genetic algorithms contain the steps

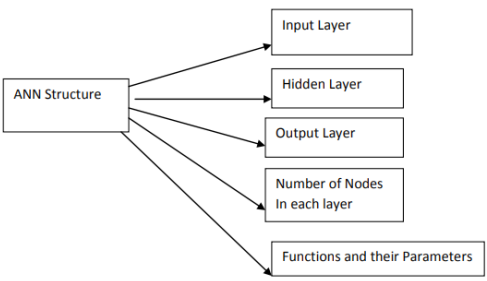
1. Construction of genome (individual),
2. Generation of initial population (group of individuals),
3. Evaluation of individuals,
4. Selection of individuals based on criteria,
5. Generation of new individuals (Mutation, Crossover)
6. Repetition of the process - generation, evaluation, selection,

7. Termination of the process based on max generation no. and/or performance criteria.

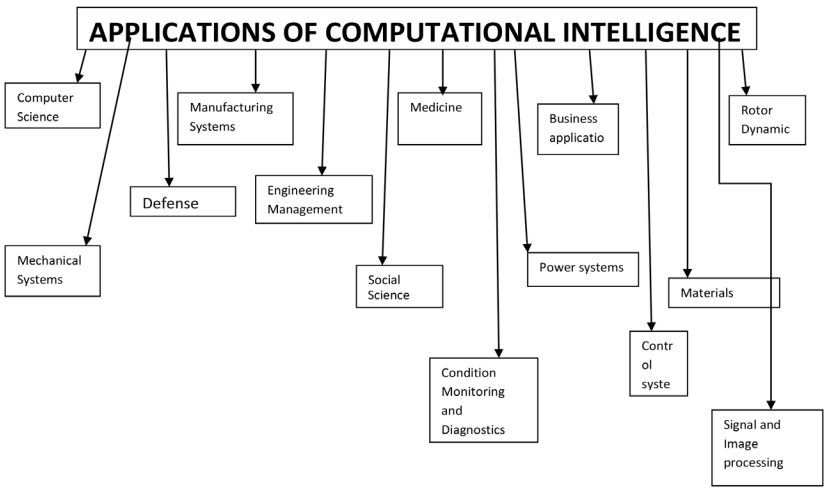
Sometimes we may use combinations where in we Combine advantages of GA and other classifiers, GA and ANN, GA and ANFIS, GA and SVM, automatic selection of classifier structure and parameters, Selection of most important system features from a pool, Selection of most important sensors (in the context of on-line condition monitoring and diagnostics)- sensor fusion. In fact automatic selection of classifier structure and parameters consist of ANNs -Number of neurons in hidden layer, ANFIS - Number of MFs and their parameters and SVM parameters.





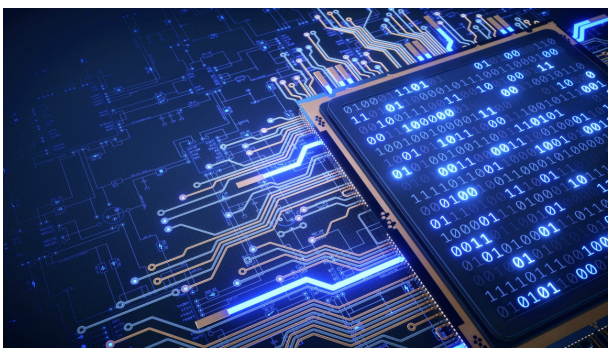


Genetic Programming (GP) is a branch of GA with a lot of similarities. The main difference of GP and GA is in the representation of the solution. In GA the output is in form of a string of numbers representing the solution. GP produces a computer program in the form of a tree-based structure[relating to the inputs or leaves, the mathematical functions (nodes) and the output (root node)].



5.3 THE MAIN CI TECHNIQUES

We will look briefly at the following key CI paradigms: Evolutionary Algorithms, Neural Networks, Fuzzy Systems and Multi-Agent Systems. This will be followed by a short summary covering some other important techniques included in this collection.



5.3.1 Evolutionary Algorithms

Evolutionary algorithms (EAs) comprise a class of techniques inspired by evolution and natural selection. The best known EAs are undoubtedly the genetic algorithms (GAs) developed by John Holland in the 1960's and 70's. Contemporaries of Holland independently developed some similar techniques however, for example of Rechenberg introduced evolution strategies (ES) and Fogel, Owen and Walsh developed evolutionary programming (EP). Since these early days, interest in evolutionary-inspired algorithms has grown extensively, and many new variations have appeared, often very different from the original models conceived by Holland, Rechenberg or Fogel. For example, in the early 1990s, John Koza proposed genetic programming: an evolutionary style technique for evolving effective computer programs, mostly using the LISP programming language. Other popular paradigms to have been derived from the more generic approach include artificial life, evolvable hardware, ant systems and particle swarms, to name but a few. Artificial Immune Systems have also become a popular topic for research in recent years, drawing analogies with some of the ingenious problem-solving mechanisms observed in natural immune systems and applying them to a broad range of real-world problems. In addition, there are many examples of hybrid (or memetic) approaches where problem specific heuristics, or other techniques such as neural networks, fuzzy systems, or simulated annealing, have been incorporated into a GA framework. Thus, due to the growth in popularity of search and optimization techniques inspired by natural evolution during the last few decades, it is now common practice to refer to the field as evolutionary computing and to the various techniques as evolutionary algorithms. In addition, evolutionary techniques for simultaneously optimizing several objectives have recently become popular. These approaches, collectively known as multiobjective evolutionary algorithms are very effective at balancing the frequently conflicting objectives to produce excellent trade-off solutions, from which a human decision maker can make an informed choice.

The analogy with natural population structures and their geographical distributions make parallel implementations highly desirable, to speed up processing and to facilitate complex emergent behaviour from simple components within the distributed populations.

Given the range of EAs mentioned above, it is not perhaps surprising that there is no rigorous definition of the term "evolutionary algorithm" that everyone working in the field would agree on. There are, however, certain elements that the more generic types of EA tend to have in common:

1. a population of chromosomes encoding candidate solutions to the problem in hand,
2. a mechanism for reproduction,
3. selection according to a fitness, and
4. genetic operators.

Figure 3 gives an outline of a generic EA. The process is initialized with a starting population of candidate solutions. The initial population is frequently generated by some random process, but may be produced by constructive heuristic algorithms, or by other methods. Once generated, the candidate solutions are evaluated to establish the quality of each solution, and based on this quantity, a fitness value will be computed, in such a way that better quality solutions will be assigned higher values for their fitness. Individuals will next be selected from the population to form the parents of the next generation, and these will be duplicated and placed in a mating pool. The selection process is frequently biased, so that fitter individuals are more likely to be chosen than their less fit counterparts. Genetic operators are then applied to the individuals in the mating pool. The idea is to introduce new variation, without which no improvement is possible. Recombination (also known as crossover) is achieved by combining elements of two parents to form new offspring. Mutation, on the other hand, involves very small random changes made to solutions. The final stage in the cycle requires the population is updated with new individuals.

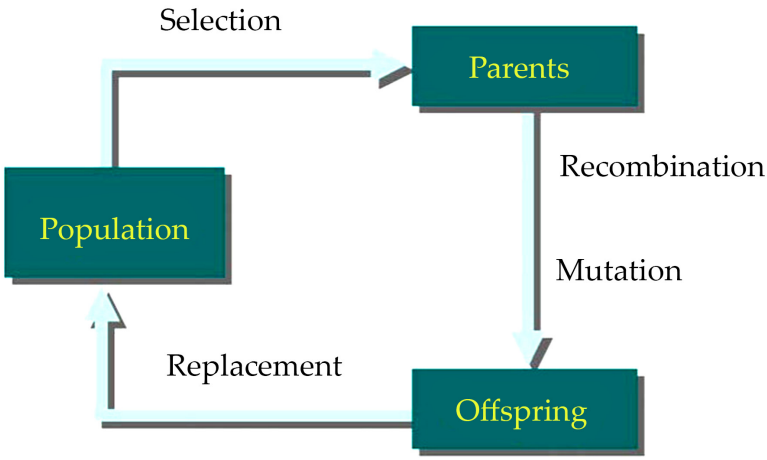


Figure 3: The Evolutionary Cycle.

Depending on the style of the EA, this may involve replacing the parent population in its entirety, or partial replacement is favoured by some researchers - perhaps replacing the poorest 10 % of the population by the best offspring, for example.

5.3.2 Neural Networks

Artificial Neural Networks (ANNs) are inspired by biological nervous systems, and emulate a simple "brain". They consist of large numbers of highly interconnected processing elements (neurons) working together and learning from experience. ANNs are specially configured for each application, and typical uses include pattern recognition and data classification. In a biological nervous systems, learning involves making adjustments to the synaptic connections between the neurons. In a similar way for ANNs, learning is accomplished through the adjustment of weights by application of some "learning rule" to the connections between the artificial neurons or nodes. Learning rules typically attempt to reinforce connections that contribute to a "correct output", and penalize connections that produce incorrect results. There are three main classes of ANN, distinguished by their different learning processes: 1) supervised learning, 2) unsupervised learning, and 3)

reinforcement learning. With supervised learning a training stage uses a set of test data and a teacher to score the performance of the ANN, then adjusts the connection weights in an effort to improve the performance to better match the actual output to the predicted output. The most widely known supervised learning ANNs are the backpropagation nets. ANNs that use unsupervised learning do not have a training stage, and these are frequently referred to as "self organizing networks".

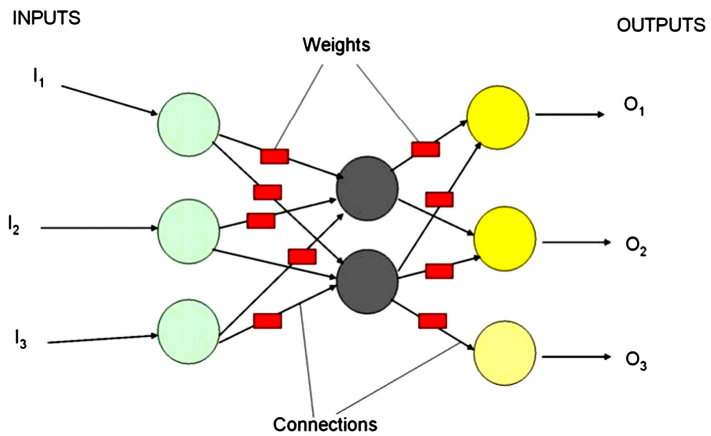


Figure 4: A Neural Network with One Hidden Layer.

Kohonen nets are the best known example of this type. In reinforcement learning data is not usually available. Instead the aim is to discover a policy for selecting actions that minimize some measure of long-term cost. A schematic neural network is illustrated in Figure 4.

5.3.3 Fuzzy Systems

Fuzzy logic was first proposed by Lotfi A. Zadeh of the University of California at Berkeley in a 1965. It is a modification of boolean (or crisp) logic which allows approximate and common sense reasoning in the absence of "true" or "false" certainty. In crisp logic, set membership is "all or nothing". In contrast, fuzzy logic allows partial membership of sets, known as fuzzy sets, and forms

the basis of fuzzy systems. Fuzzy Systems can deal with partial truth and incomplete data, and are capable of producing accurate models of how systems will behave in the real world, particulary when appropriate conventional system models are not available. Instead of supplying equations for a mathematical model, for example, a designer will need to produce appropriate fuzzy rules to describe the system he/she wishes to implement. The system operates when inputs are applied to the rules consisting of the current values of appropriate membership functions. Once activated, each rule will fire and produce an output, which will also be a partial truth value. In the final stage, the outputs from all the rules are combined, in some way, and converted into a single crisp output value. In summary, a fuzzy system consists of the following:

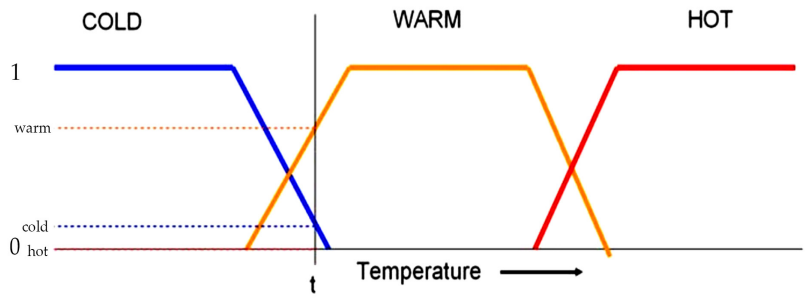


Figure 5: A Fuzzy Temperature Control System.

- a set of inputs
- a fuzzification system, for transforming the raw inputs into grades of memberships of fuzzy sets
- a set of fuzzy rules
- an inference system - to activate the rules and produce their outputs
- a defuzzification system - to produce one or more final crisp outputs

We will now look at a simplistic fuzzy system: a fuzzy controller for room temperature.

The fuzzy set membership diagram in Figure 4 characterizes three functions, identifiable as subranges of temperature: cold, warm and hot. Suppose we wish to keep a room at a comfortable temperature (warm) by building a control system to adjust a room heater. We can see in Figure 4 how each function maps the same temperature value to a truth value in the 0 to 1 range, so that any point on that scale has three "truth values", one for each of the three functions. It is these truth values that are used to determine how the room temperature should be controlled. The vertical line in the diagram represents a particular temperature, t. At this temperature it is easy to observe the degree of membership to "hot" (red) is zero, this temperature may be interpreted as "not hot". Membership of "warm" is about 0.7, and this may be described as "fairly warm". Similarly, examining membership of the "cold" function gives a value of about 0.15, which may describe it as "slightly cold". Adjectives such as "fairly" and "slightly", used to modify functions are referred to as "hedges", and can be a useful way to specify subregions of the functions to which they are applied.

To operate our fuzzy temperature control system, we require a number of fuzzy IF-THEN rules, in the form of "IF variable IS property THEN action". For example, an extremely simple temperature regulator that uses a heater might look like this:

1. IF temperature IS cold THEN turn heater to high
2. IF temperature IS warm THEN do nothing
3. IF temperature IS hot THEN turn off heater

Notice there is no "ELSE". All of the rules are evaluated, because the temperature will belong to all three sets (cold, warm and hot) at the same time, but to different degrees. At temperature t in Figure 4, for example, M(cold) = 0.15,M(warm) = 0.7 and M(hot) = 0.

Obviously, the greater the truth value of "cold", the higher the truth value of "turn the heater to high", although this does not necessarily mean that the output itself will be set to "high", since this is only one rule among many. In our example, the partial truth inputs for "cold", "warm" and "hot" will in turn produce

partial truth values for the outputs “turn the heater to high”, “do nothing” and “turn the heater off”. The simplest way to produce a single crisp instruction, is to select the output with the maximum value (which will probably map to “do nothing” in the case of our temperature t). A more sophisticated method involves finding the centroid of all the outputs. This methods locates the “centre of mass” of the combined membership function curves.

More complex rules can be built for fuzzy systems, using AND, OR, and NOT operators. These are the counterparts of the familiar crisp logic operators, and they are usually defined (respectively) as the minimum, maximum, and complement. So, for the fuzzy variables x and y:

NOT x = (1 - truth(x))

x AND y = minimum(truth(x), truth(y))

x OR y = maximum(truth(x), truth(y))

Clearly, the simple temperature controller described above is for illustration only, and practical fuzzy systems will typically be made up from many more rules - perhaps hundreds or even thousands. In these more sophisticated systems, it is likely that the fuzzy rule set will be less “flat”, and form more of a hierarchy, so that the outputs of some rules provide inputs to others. Systems with large rule sets will probably require more sophisticated inference systems to ensure the efficient processing of the rules, in a reasonable order.

To complete this section, it is worth mentioning a variation of fuzzy sets called rough sets. Rough Set Theory was introduced in the early 1980s. The basic idea is to take concepts and decision values, and create rules for upper and lower boundary approximations of the set. With these rules, a new object can easily be classified into one of the regions. Rough sets are especially helpful in dealing with vagueness and uncertainty in decision situations, and for estimating missing data. Uses include data mining, stock market prediction and financial data analysis, machine learning and pattern recognition.

5.3.4 Multi-Agent Systems

A multi-agent system (MAS) is a system composed of many interacting intelligent agents; each one is in itself simple and apparently acts only in its own interest, yet by collaborating and/or competing with each other agents, an MAS can be used to solve problems which would entirely defeat an individual agent or a monolithic system. MAS can exhibit self-organization and complex behaviour can emerge. Example applications include financial forecasting and online trading and disaster response.

The agents in a multi-agent system have several important characteristics:

- Autonomy: the agents are at least partially autonomous
- Local views: no agent has a full global view of the system
- Decentralization: there is no one controlling agent
- Typically multi-agent systems research refers to software agents. However, the agents in a multi-agent system could equally well involve robots, humans or human teams. A multi-agent system may contain combined human-software agent teams.

Generally, multi-agent systems are flexible and they are easily maintained or modified without the need for drastic rewriting or restructuring. MAS also tend to be robust and recover easily from a breakdown, due to built in duplication and redundancy of components.

5.4 COLLECTIVE INTELLIGENCE AND OTHER EXTENSIONS OF EVOLUTIONARY COMPUTATION

It's important to understand upfront that there are very many extensions of evolutionary algorithms. This text is intended to provide an introduction to the field of evolutionary computation that would be suitable as part of a broader introduction to computational intelligence within the framework of a one semester

college course. By consequence, we cannot cover everything here and do it in any level of depth that would be appropriate.

This chapter begins with a population-based optimization approach called particle swarm optimization (PSO), which models the flocking behavior we see in certain animals. We then focus on another population-based approach called differential evolution, which searches a landscape for optima by using different vectors between existing solutions. Another approach follows that is based on modeling how ants search and find food sources. It turns out that the strategy ants use can be applied for finding short paths through graphs and other engineering problems. Finally, we'll conclude with the application of evolutionary algorithms to multiple criteria optimization problems.

5.4.1 Particle Swarm Optimization

When considering how a flock of birds or a school of fish moves, it's easy to think that the actions of each individual are somehow coordinated with the actions of the others, and that's exactly the case. Research shows, for example, that starlings coordinate their movements based on seven neighboring birds. Thousands of starlings may form a flock, but the movements of the flock are based on the local interactions of overlapping groups of just seven birds.

It's interesting that modeling the way birds flock, or fish school, or things in general "swarm" can lead to an optimization algorithm. The organisms that we might model in these cases are likely optimizing something, such as group cohesiveness versus individual effort. That might not correlate to what we'd like to do in terms of using swarming as a method to find points of interest on an objective function, but we can tailor a swarming method toward that end.

Particle swarm optimization was introduced as such an approach. Suppose you have a collection of possible solutions to a problem, which we'll call particles. The particles reside in R^n and the goal

is to find the minimum, maximum, or some other point of an objective function f(x), where $x \circledcirc R^n$. Much like the solutions in the population of an evolutionary algorithm, these particles are located at various positions and can be evaluated in light of some objective criteria. Each particle is denoted by $x_i(t)$, where x_i is the ith particle located at x (a vector) at a particular time t.

Next comes the swarming part: How do the particles move? They each have a velocity vector, denoted by $v_i(t)$, where v_i is the velocity of the ith particle at a particular time t. If there were no swarming, each of the particles would move according to their unchanging velocity forever. The fact that the particles will swarm means that their velocities will change as a function of what is known about other particles in the collective, and also what a particle remembers about where it has been.

Each particle is given a memory. It remembers the location that has yielded the best result from the objective function. Each particle also has knowledge about the results of other particles in a neighborhood (akin to starlings being aware of other starlings), and each particle knows the location of the particle in its neighborhood that has the best result from the objective function. That information is used to change the velocities of the particles and thereby having them move to different locations, searching for a better location.

Each particle's new velocity is a function of

(i) its current velocity,

(ii) the vector that points to the particle's own best location, and

(iii) the vector that points to the best location of the particles in its neighborhood. The vectors that point to known best locations are weighted by random variables. With starlings, the neighborhood is believed to be seven birds, but in PSO the neighborhood can be small, like a particle and its two closest neighbors, or it can expand to be the entire collection of particles. Each particle then moves according to its new velocity.

A little pseudo code can help clarify the update procedure for each particle:

$$\mathbf{v}_i(t+1) = a \times \mathbf{v}_i(t) + b \times U_1 \times (\text{PersonalBest}_i - \mathbf{x}_i(t)) + c \times U_2 \times (\text{NeighborhoodBest}_i - \mathbf{x}_i(t))$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

where U_1 and U_2 are distributed U(0; 1), PersonalBest_i is the location where the ith particle found the best score in its memory, $\text{NeighborhoodBest}_i$ is the location where the best score was found in the ith particle's neighborhood of particles, and a, b, and c are scaling terms. Some basic settings for the scalars are a = 1, and b + c = 4. The trade-off of b and c essentially weights the relative importance of a particle's own experience for the importance of its neighborhood's experience.

One issue that comes with these update equations is that the particles can see their velocities increase in magnitude without bound. That's not particularly realistic for modeling real flocking and empirically it's not helpful when searching for optima in our typical objective functions. So, a limit is placed on the velocity in each dimension, v_{max}. Some experimentation may be required to have a good setting for this maximum value.

Another issue that arises is the choice of the number of particles. Many publications have used collections of 10–50 particles, but the appropriate size is problem dependent, as is the choice of how to construct a neighborhood for each particle.

Here's an example that uses 50 particles with a neighborhood of 3 particles and $|v| = 1$ applied to the function $f(x, y) = x^2 + y^2 - 20[\cos(\pi x) + \cos(\pi y) - 2]$, which is depicted in Figure 6. The global minimum value of the function is $f(x, y) = 0$ and the second-best minimum value is $f(x, y) = 4$.

In this example, all the particles were initialized uniformly at random between 10 and 10 in each dimension. After 500 iterations, the best score found was 0.0355. Figure 7 shows the best score of all the particles at each generation and the average of each particle's

historical best score. The graph shows continual improvement toward the global optimum.

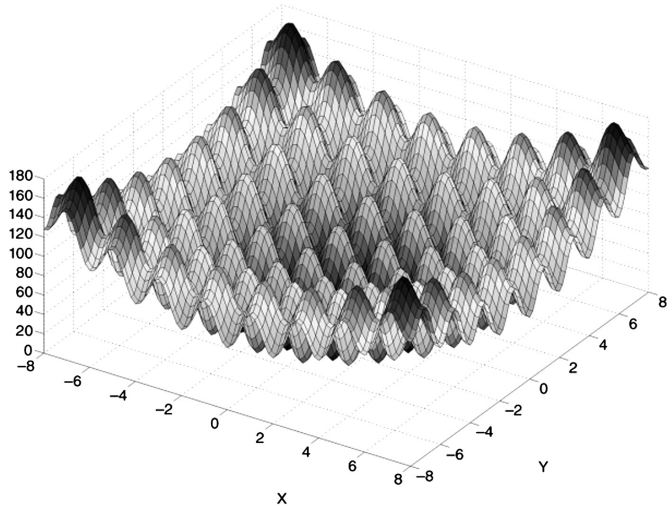


Figure 6: A surface with multiple local optima used for testing the particle swarm optimization algorithm.

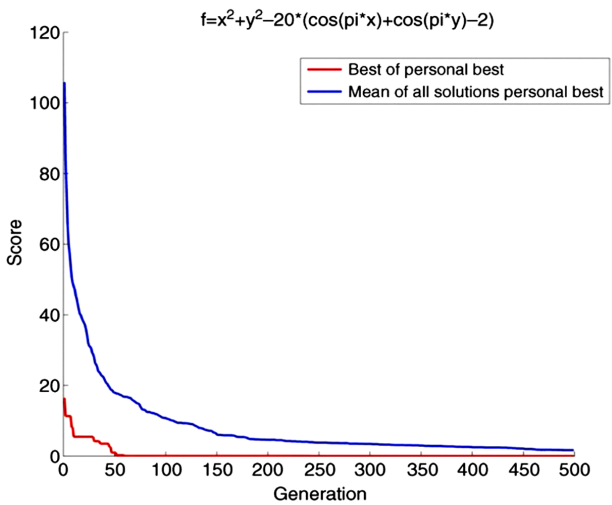


Figure 7: The rate of optimization for the best score ever found and the mean of all particles' best scores for the problem shown in Figure 6.

There are many areas for investigation in particle swarm optimization. For example, it may be that neighborhood size would

benefit by varying by particle, or varying by particle over time. The learning factors that represent the weights on the acceleration terms can also be subject to online learning per particle. The velocity update equation can be modified by applying a "constriction" factor to the overall velocity, which relieves the need for setting a value of v_{max}. Other extensions include applications in discrete space and dynamic settings.

5.4.2 Differential Evolution

Another population-based search algorithm that relies on updating the location of the individuals in the population is called differential evolution. It was introduced in Storn and Price [1996], Storn [1996], and other publications about the same time as particle swarm optimization and is tailored to searching in real-valued spaces for maxima or minima of functions. Each of the individuals in the population is subject to a form of mutation and recombination, as well as selection.

The key ingredient in differential evolution is that individuals move based on the differential vectors from the individual to other individuals in the population. The population size needs to be at least four individuals and as with all evolutionary or related methods, the population is initialized at random or based on some prior knowledge of where to search for a solution to the problem posed.

Each of the individuals in the population is subjected first to mutation, which is based on the individual's relationship to three other distinct individuals in the population, chosen at random each time. Let's call these three individuals x_1, x_2, and x_3, and let's call the individual that we are mutating x_0.

First we pick a random dimension of the problem, d, uniformly from 1 to n, where the problem has n dimensions. We'll remember that dimension. Then, for each dimension i = 1; ... ; n, we create a uniform random number $u_i \sim U(0, 1)$. If $u_i < p_c$, then the new value of the solution in the ith dimension is given by

$$x_{0i} = x_{1i} + a(x_{2i} - x_{3i})$$

where a is a scalar value between [0, 2] called a differential weight; otherwise, the new value of the solution in the ith dimension is retained from x_{0i}. The value p is a c "crossover probability," which really means that it is transplanting a value for that dimension from another random solution added with a weighted combination of a difference between two other solutions.

Finally, for this solution, if the new version of x_0 is better than the original, it replaces the original in the next generation; otherwise the original x_0 is retained for the next generation. This process continues until a solution of sufficient quality is found, or a maximum number of generations is met.

Let's see how this approach works on two simple functions. The first is the usual 2 quadratic bowl defined by $f(x, y) = x^2 + y^2$ (Figure 8), and the second is the same one that we just saw in Figure 6, which has multiple local optima.

We'll use 50 solutions in the population in each case, with our scalar value a = 0:8 and p = 0:02, with all solutions initialized uniformly at random between 20 and 20.

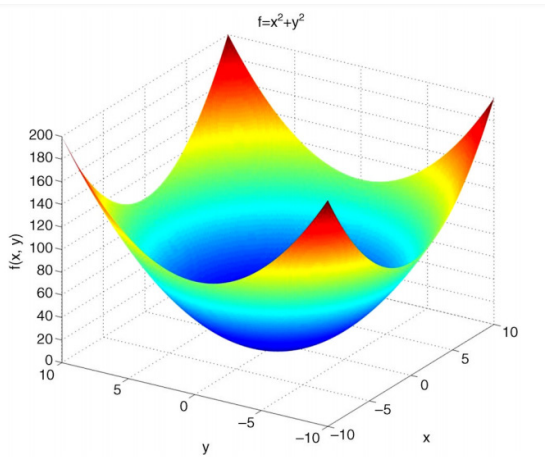


Figure 8: A simple quadratic function $f(x, y) = x^2 + y^2$ to be used in the differential evolution experiments.

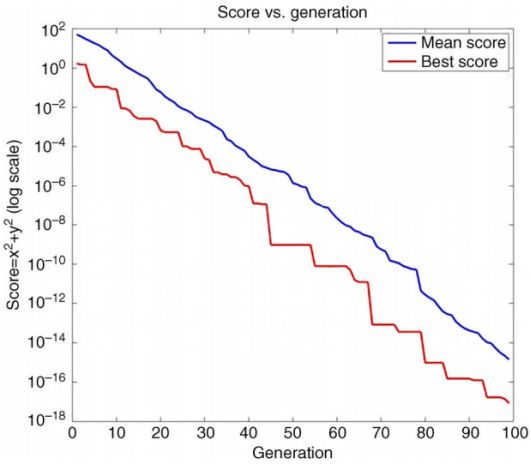


Figure 9: The average of all scores in the population (n = 50) and the best score in the population when applying differential evolution to the quadratic bowl. Note that the y-axis is on a log scale.

Figure 9 shows the average of all the scores in the population and the bests core as a function of the number of generations for the quadratic bowl and Figure 10 shows these data when applying differential evolution on the multimodal surface.

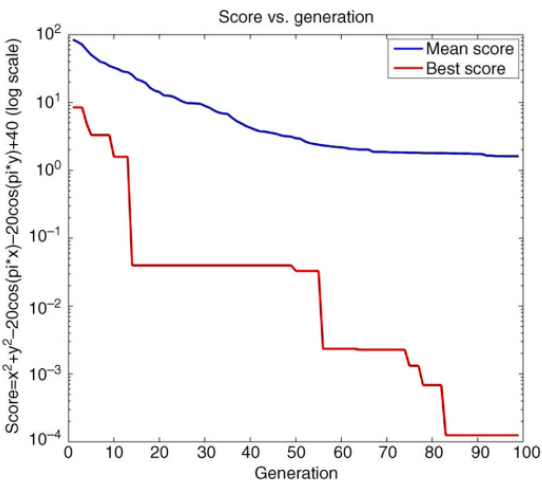


Figure 10: The average of all scores in the population (n = 50) and the best score in the population when applying differential evolution to the multli

modal function $f(x, y) = x^2 + y^2 - 20[\cos(\pi x) + \cos(\pi y) - 2]$. Note that the y-axis is on a log scale.

For the quadratic bowl, the best solution has less than 10^{-6} error in about 40 generations. For the multimodal function, the method is still able to find a solution that has less than 10^{-4} error in about 80 generations. A quick comparison with the results using PSO on the same multimodal function favors the use of differential evolution; however, it's always important to remember that this may not generalize to other functions, and may be sensitive even simply to the initialization.

5.4.3 Ant Colony Optimization

Another biologically inspired method for solving problems is called ant colony optimization (ACO), and it simulates how ants discover food sources and communicate their discoveries with other ants. ACO predates PSO and differential evolution and is used generally for combinatorial optimization problems.

Here's the thinking underlying the approach. Suppose you're an ant and you have to get from your colony's nest to a food source. The problem is that you don't know where the food source is. So you have to search for it. Once you find it, you have to get back home, so you have to search for that too.

It would be great if you could share what you learn as you search with other ants, so that they could search more efficiently. Nature provides a way to do that. It's called a pheromone trail, and ants leave pheromone trails as they move. The strength of the trail is dependent on the number of ants that traverse the trail and how recently the ants have traversed the trail.

More ants on a trail means more pheromone, which entices other ants to follow that trail rather than search somewhere else. The pheromone evaporates over time, so once food sources are exhausted, the ants don't continue to travel to an "empty refrigerator" forever.

Let's see how we can model these principles to address a classic 30-city traveling salesman problem. The ants will start at a particular city and they have to find a complete path that visits every other city once and only once and then returns home. The objective is to complete the path with the shortest possible distance. Let's say we have a set of 50 ants in our population.

It's time for the first ant to start its trek. The probability of it visiting any of the other available city is given by

$$p(i,j) = \frac{ph(i,j) \times cost(i,j)^{-1}}{\sum_{k \in All} ph(i,k) \times cost(i,k)^{-1}}$$

where p(i; j) is the probability of going from city i to city j, ph(i; j) is pheromone factor between city i and city j, and cost(i; j) is the distance between city i to city j. When starting, we set the value of ph(i; j) to 1.0 for all pairs of cities.

From the formula, the probability that the ant will go to city j is dependent in part on the pheromone factor. The higher the value of ph(i; j), the higher the likelihood of traveling to city j. Also, the probability is inversely related to the distance between the current city i and the new city j. Each ant wanders through the cities in traveling salesman problem according to the probability equation, which is computed for all available (unvisited) cities at each step through the path for each ant.

The pheromone factor can be updated in different ways. For our example, let's say we update the pheromone factor after each of the 50 ants has traversed a tour of the cities. The new pheromone strength is computed as

$$\text{new } ph(i,j) = \alpha ph(i,j) + \Delta ph(i,j)$$

where α is the evaporation rate and $\Delta ph(i; j)$ is the delta increment that comes from ants going from city i to city j. Here, let's say that the evaporation rate $\alpha = 0{:}9$ and

$$\Delta ph(i,j) = \sum_{k=1}^{50} \begin{cases} Q, & \text{if kth ant traveled between city i and city j} \\ 0, & \text{else} \end{cases}$$

and Q = 1/50 (i.e., the inverse of the number of ants).

After each set of 50 ants traverses the graph, the pheromone factors are updated, encouraging the ants to favor shorter paths that have been well traveled.

Here are some results from this approach on a 30-city traveling salesman problem. Figure 11 shows the best path found after 800 iterations of the 50 ants traversing the graph. The problem was created by distributing the cities uniformly at random in a 100 × 100 square. The final best path is 411.41 units long. Statistical mechanics can be used to estimate the optimum path length for n randomly distributed cities in an area A by the formula 0:749 ×(nA) $^{0:5}$.

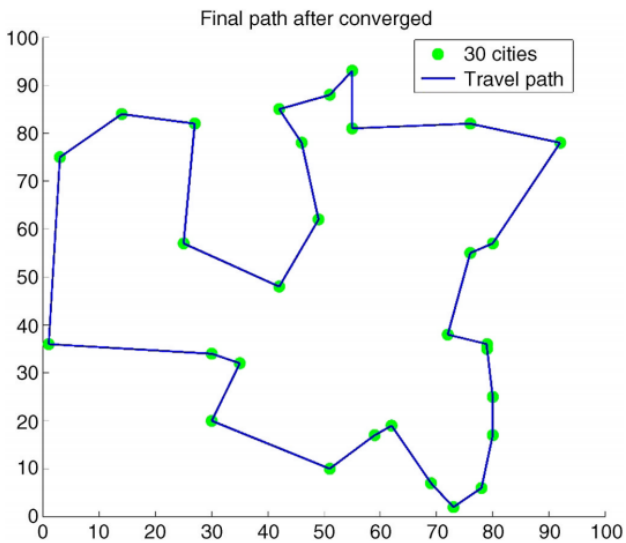


Figure 11: The best path found by a group of 50 ants iterated 800 times using the ant colony optimization routine on a randomly generated 30-city traveling salesman problem.

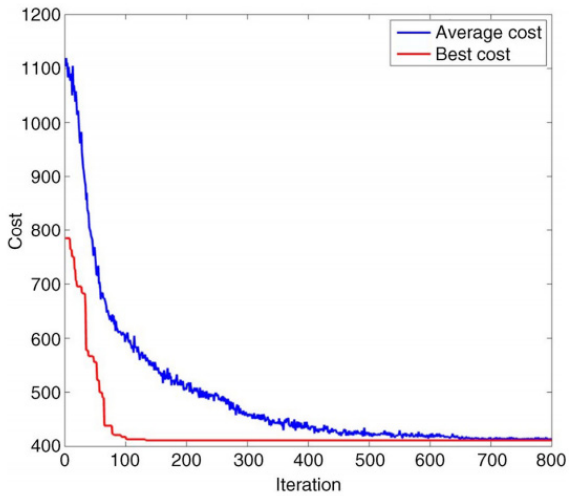


Figure 12: The average cost over each of the 50 ants at each iteration and cost of the best path found when iterating 800 times using the ant colony optimization routine on the traveling salesman problem shown in Figure 11.

In this example, the expected best solution is 410.24. So, the solution is close to the expected best tour length.

Figure 12 shows the rate of optimization of the best and average cost of tours found during the search. By about the 100th iteration, the best solution already has nearly the expected best score. By the 800th iteration, the average tour length across all 50 ants is also close to this score, indicating that the search has converged.

Ant colony optimization has been applied to a diverse set of engineering problems. For example, the method has been employed to construct neural network topologies, design type-2 fuzzy controllers, induce decision trees, perform fingerprint analysis, and many other application areas.

REFERENCES

1. Ali, Shawkat, Multidisciplinary Computational Intelligence Techniques: https://books.google.co.in/books?isbn=1466618310. - 2012
2. Andries P. Engelbrecht , Computational Intelligence - An Introduction Wiley , 2007.
3. Fei-Yue Wang (The University of Arizona, USA), Derong Liu (University of Illinois at Chicago, USA. Advances in Computational Intelligence: Theory and Applications2014.
4. Kevin E. Voges, Nigel Pope.(editors) ; Business Applications and Computational Intelligence, https://books.google.co.in/books?isbn=1591407044 Kevin E. Voges, Nigel Pope – 2006.
5. Ladislav Madarász, Jozef Živčák (editors) , Aspects of Computational Intelligence: Theory and Applications: . https://books.google.co.in/books?isbn=3642306683; 2012.
6. NP Belfiore, Applications of computational intelligence to mechanical engineering , ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7028702, 2014 .
7. Ondrej Linda, Applications of Computational Intelligence In Critical Infrastructures: Network Security, Robotics, And System Modeling Enhancements, July 2009, University of Idaho.
8. Rami Abielmona, Rafael Falcon, Nur Zincir-Heywood, Hussein A. Abbass., Recent Advances in Computational Intelligence in Defense and Security, Springer
9. Reusch, Bernd (Ed.), Computational Intelligence, Theory and Applications; International Conference 9th Fuzzy Days in Dortmund, Germany, Sept. 2006.
10. Shi, Yuhui , Recent Algorithms and Applications in Swarm Intelligence https://books.google.co.in/books?isbn=1466624809 - 2012
11. The Birth and historical Development of Computational Intelligence applications in archeology. Archeologia e Calcolatori, 20, 2009, 95-109.

12. Tomasz G. Smolinski, Mariofanna G. Milanova, Aboul-Ella Hassanien ; Vol. 1: Applications of Computational Intelligence in Biology: Current Trends and Open Problems, Vol. 2: Computational Intelligence in Biomedicine and Bioinformatics: Current Trends and ; Applicationwww.biology.emory.edu/research/Prinz/Tomasz/CIinBiology/CfC.html..
13. Yong-Hua Song, Johns, Allan, Aggarwal, Raj Computational Intelligence Applications to Power Systems , www.springer.com/gp/book/9780792340751.
14.

CHAPTER 6

ARTIFICIAL INTELLIGENCE

INTRODUCTION

Artificial intelligence (AI) is any task performed by program or machine, which otherwise human needs to apply intelligence to accomplish it. It is the science and engineering of making machines to demonstrate intelligence especially visual perception, speech recognition, decision-making, and translation between languages like human beings. AI is the simulation of human intelligence processes by machines, especially computer systems. This includes learning, reasoning, planning, self-correction, problem solving, knowledge representation, perception, motion, manipulation, and creativity. It is a science and a set of computational techniques that are inspired by the way in which human beings use their nervous system and their body to feel, learn, reason, and act. AI is related to machine learning and deep learning wherein machine learning makes use of algorithms to discover patterns and generate insights

from the data they are working on. Deep learning is a subset of machine learning, one that brings AI closer to the goal of enabling machines to think and work as human as possible.

AI is a debatable topic and is often represented in a negative way; some would call it a blessing in disguise for businesses, while for some it is a technology that endangers the mere existence of humankind as it is potentially capable of taking over and dominating human being, but in reality artificial intelligence has affected our lifestyle either directly or indirectly and shaping the future of tomorrow. AI has already become an intrinsic part of our daily life and has greatly impacted our lifestyle despite the imperative uses of digital assistants of mobile phones, driver-assistance systems, the bots, texts and speech translators, and systems that assist in recommending products and services and customized learning. Artificial intelligence (AI) is not a new technology. For decades, computer scientists have tried different approaches to reach the holy grail of computing: intelligent machines. While we are still far away from replicating the wonders of the human brain, AI applications have started to fill our daily lives and power our electronic devices, from smartphones to home alarm systems.

6.1 THE PATH TO MODERN AI

As humans, we've always tried to find ways to understand the world around us and bend nature to meet our goals. To do so, we have always relied on external tools that amplify our brain's capabilities.

The abacus was probably the first such tool, invented about 5,000 to 6,000 years ago to help people make calculations. Although it's still used in schools to help children visualize simple mathematical operations, it doesn't really save us from the labor of actually performing them. We had to wait until the 1960s for the first machines that could add and subtract numbers automatically. Computers have come a long way since then, but deep down their capability has still been pretty simple: executing calculations

exactly as some (expert) human has instructed them to do. There's little "intelligence" in them.

The two words *artificial* and *intelligence* were first put together on August 31, 1955, when professor John McCarthy from Dartmouth College, together with M.L Minsky from Harvard University, N. Rochester from IBM, and C. E. Shannon from Bell Telephone Laboratories, asked the Rockefeller Foundation to fund a summer of research on artificial intelligence. Their proposal stated the following:

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. . . . An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

The researchers knew that tackling intelligence as a whole was too tough of a challenge, both because of technical limitations *and* the inherent complexity of the task. Instead of solving the broad concept of intelligence, they decided to focus on subproblems, like language. Later, these applications would be called *narrow AI* . An artificial intelligence capable of matching or surpassing human capabilities would instead be called *general AI* . In other words:

- *General AI* (or *strong AI*) --An artificial intelligence program capable of tackling every kind of task it's presented. This is similar to an extremely resourceful human, and you can think of it as the robot from *The Terminator* (or, hopefully, a more peaceful version of it).
- *Narrow AI* --An artificial intelligence program capable of solving a single, well-defined task. It can be broad (recognizing objects from pictures) or extremely specific (predicting which customers who bought product A are more likely to purchase product B as well). This means

one task at a time, and not any other: an AI that recognizes cats in images can't translate English to Italian, and vice versa.

General AI is still far away: researchers still don't know when we'll finally get it. Some argue that we'll never get there. Even though general AI is still a distant, fuzzy dream, this is what many people have in mind when AI is mentioned in the news. If you were one of those people, and are now disappointed that general AI is not here yet, don't despair. Narrow AI applications are still capable of creating immense value. For example, AI that can detect lung cancer is a narrow application but nevertheless extremely useful.

The results of the Dartmouth research summer of 1956 were so interesting that they sparked a wave of excitement and hope among the participants. The enthusiasm of the scientists spread to the US government, which started heavily funding research on a specific application: English/Russian translation. Finding trustworthy Russian translators must not have been easy in the midst of the Cold War.

After the first few years of work, a government committee produced the infamous 1966 Automatic Language Processing Advisory Committee (ALPAC) report. The document featured the opinions of many researchers about the state of AI research. Most were not very positive:

Early machine translations of simple or selected text . . . were as deceptively encouraging as "machine translations" of general scientific text have been uniformly discouraging. . . . No one can guarantee, of course, that we will not suddenly or at least quickly attain machine translation, but we feel that this is very unlikely.. . . there is no immediate or predictable prospect of useful machine translation.

The ALPAC report marks the beginning of a period called the *first AI winter*: public funding for AI research stopped, excitement cooled, and researchers focused their work on other fields.

Interest in AI faded until the 1980s, when private companies such as IBM and Xerox started investing in a new AI spring. New hopes were fueled by a technology called *expert systems*: computer programs that encode the knowledge of a human expert in a certain field in the form of precise, *if-then* rules. An example will help you understand how expert systems were designed to work.

Suppose you want to build an AI system that can stand in for a gastroenterologist. This is how you do it with an expert system: you ask a doctor to describe with extreme precision how they make decisions about patients. You then ask a programmer to painstakingly transform the doctor's knowledge and diagnosis flow to if-then rules that can be understood and executed by a computer. An extremely simplified version would look something like this:

- If the patient has a stomachache and the body temperature is high, then the patient has the flu.
- If the patient has a stomachache and has eaten expired food, then the patient has food poisoning.

And so on. Once the doctor's knowledge is encoded into the software and a patient comes in, the software follows the same decision path as the doctor and (hopefully) comes up with the same diagnosis. This approach has several problems:

- *Poor adaptability* --The only way for the software to improve is to go back to the drawing board with a computer scientist and the expert (in this case, the doctor).
- *Extreme brittleness* --The system will fail in situations that weren't part of the original design. What if a patient has a stomachache but normal body temperature, and hasn't eaten spoiled food?
- *Tough to maintain* --The complexity of such a system is huge. When thousands of rules are put together, improving it or changing it is incredibly complicated, slow, and expensive. Have you ever worked with a huge Microsoft Excel sheet and struggled to find the root cause of a mistake? Imagine an Excel sheet 100 times bigger.

Expert systems were a commercial failure. By the end of the 1980s, many of the companies that were developing them went out of business, marking the beginning of the *second AI winter* . It wasn't until the early 2000s that the next generation of AI successes came along, fueled by an old idea that became new again: machine learning.

6.1.1 Challenges

Building trust: The AI is all about science, technology, and algorithms which mostly people are unaware of, which makes it difficult for them to trust it.

AI human interface: Being a new technology, there is a huge shortage of working manpower having data analytics and data science skills; those in turn can be deputed to get maximum output from artificial intelligence. As the advancement of AI rising, businesses lack a skilled professional who can match the requirement and work with this technology. Business owners need to train their professionals to be able to leverage the benefits of this technology.

Investment: AI is an expensive technology that not every business owner or manager can invest money into as large amount of computing power will be necessary and sometimes hardware acceleration with GPU, FPGA, or ASIC must be in place to run machine learning models effectively. Though adoptability of AI is surging high, it has not been integrated fully in business's value chain at the scale which it should have. Moreover, enterprises of those who have incorporated are still in nascent stage which have resulted in the slowdown in the lifting of the AI technology at scale and thus been deprived of cost benefit of scale. After decades of speculation and justifiable anxiety about the social implications of intensifying & potentially de-stabilizing AI technology for humankind and Black box problem, AI investors are bit skeptical from parking their money in potential startups.

Software malfunction: With machines and algorithms controlling AI, decision-making ability is automatically ceded to code-driven Black Box tools. Automation makes it difficult to identify the cause of mistakes and malfunctions. Moreover, due to the lack of ability of human beings to learn and understand how these tools work, they have little or no control over the system which is further complicated as automated systems become more prevalent and complex.

Non-invincible: (Can replace only certain tasks) Like any other technology, AI also has its own limitations; it simply cannot replace all tasks. However, it will result in emerging new job domain with different quality job profile.

High expectations: Research in artificial intelligence is conducted by large pool of technologist and scientists with varying objectives, motivation perspectives, and interests. Main focus of research is confined in understanding the underlying basis of cognition and intelligence with heavy emphasis on unraveling the mysteries of human intelligence and thought process. Not everyone understands the functioning of AI and might also have very high expectation of functioning.

Data security: Machine learning and decision-making capability of AI and AI application are based on huge volumes of classified data, often sensitive and personal in nature. This makes it vulnerable to serious issues like data breach and identity theft. Mostly, companies and government striving for profits and power, respectively, exploit the AI-based tools which are generally globally networked which make them difficult to regulate or rein in.

Algorithm bias: AI is all about data and algorithms. Accuracy of decision-making capability of AI is purely based on how accurately it has been trained and by using authentic and unbiased data. Unethical and unfair consequences are inherent in vital decision-making if data used for training is laced with racial, gender, communal, or ethnic biases. Such biases will probably be more

accentuated, as many AI systems will continue to be trained using bad data.

Data scarcity: Power and capabilities of AI and AI applications depend directly on the accuracy and relevancy of supervised and labeled datasets being used for training and learning. There is scarcity of quality-labeled data. Though efforts are underway by means of transfer learning, active learning, deep learning, and unsupervised learning, to devise methodologies to make AI models learn despite the scarcity of quality-labeled data, it will only aggravate the problem.

6.2 THE ENGINE OF THE AI REVOLUTION: MACHINE LEARNING

The first definition of *machine learning* dates back to 1959, from American AI pioneer Arthur Samuel:

Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.

The key elements here are *learning* and *without being explicitly programmed*. Let's focus on the latter first. Explicitly programming a computer means defining the rules and instructions it must follow to perform a specific task. This is what software engineers do when they write software that handles your everyday tasks like doing taxes or filling out spreadsheets.

People without programming experience often feel like software engineers are powerful creatures who can bend machines to their will. Unfortunately, things are not always that easy. Try to think about the various decisions you make as you perform some trivial actions: can you explain the process you follow to recognize your friends when you see them? All the split-second decisions you

make while driving? Can you list all the English grammar rules you apply as you talk? If you can't precisely explain how you do something, there's no chance that you can instruct a computer to do it.

Samuel proposed to replace "instructing computers" with "giving them the ability to learn." If you think about it, learning instead of following instructions is (coincidentally?) what human beings do all the time. Our mothers and fathers don't teach us their native tongue by giving us grammar books at the tender age of one. They just speak to us naturally, and we learn from their example, applying thousands of grammar rules without even knowing it. In fact, our brain is capable of automatically extracting rules way before it becomes capable of rationally understanding grammar at school! Even for us humans, it looks like learning rules from examples can be easier than being told about them.

In the same way we learn from experience, machine learning (ML) techniques allow computers to learn from data. Let's make it more concrete with a classic toy example: teaching a computer to tell dogs from cats in pictures. If you had to teach a kid to perform this task, you wouldn't pick up a veterinary book and start reading about the differences in ear shape or fur color. Instead, you'd probably just point them to a few pictures and let their brain do its magic.

An ML solution to the "dog or cat" problem is similar to our childhood learning experiences. We feed the computer thousands of images of cats and tell it "these are cats," and then thousands of images of dogs and tell it "these are dogs." Finally, we let it figure out the difference between the two pets automatically. We don't have to explain the key elements that distinguish dogs from cats. A good ML application learns to figure that out from the examples it receives. Figure 1 shows the difference between traditional programming and machine learning.

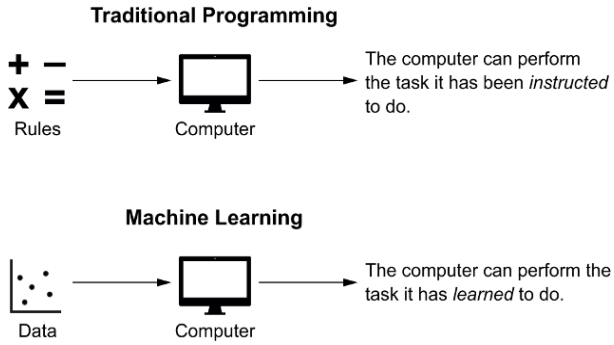


Figure 1: The difference between the traditional programming approach and machine learning: the first relies on precise rules and instructions, the latter on data and learning.

You may start to sense why ML couldn't possibly have blossomed before the 2000s. The main ingredient of this set of techniques is *data* , and the internet has made collecting data much easier. The other crucial ingredient for ML is *computing power* : learning from data doesn't happen for free, and computers need fast processors to perform this task. Thanks to cloud computing and increases in processing power, access to powerful computers has never been so easy and cheap.

To give you a sense of how much things have changed in just a few years, we asked Alex Waibel, one of the pioneers of AI in speech recognition and among the first hires of Facebook's AI team, how different it was to work on ML 20 years ago. The most powerful computer he could use in the early 2000s was as big as an apartment, cost a few million, and he needed to rent it to train his models. Today, he has much more computing power sitting on his desk for a few thousand dollars. Your phone is probably more powerful than what top researchers had available just 20 years ago.

Availability of data and cheap computing power created the perfect environment for machine learning to bloom. Indeed, many (most) of the coolest consumer-facing applications of what we call AI today rely heavily on ML: the Siri voice virtual assistant, Google Translate, self-driving cars, and many more.

Going back to the history of AI, it seems that ML is the engine that powered today's AI explosion, finally bringing some hope after the last AI winter of the 1980s. In fact, the success of modern AI has been so dependent on ML techniques that people are often confused about the difference between the two. What is artificial intelligence, then? Let's find out.

What is artificial intelligence, after all?

In our experience as technologists, consultants, and public speakers, we are constantly meeting people with different opinions about the definitions of *AI* , *data science* , and *ML* . Although many are quite opinionated, few can defend their position. Indeed, finding a universal definition of AI is not as trivial as it might look.

Going by its name, we might try to define *artificial intelligence* by finding the human traits that we associate with intelligence. Once we agree on what makes humans intelligent, we can say that any computer that does the same thing is AI. It makes sense, right? Although this is a common approach, it falls apart even with simple scenarios. For instance, a human who can divide 13.856 by 13 down to the tenth decimal number in a split second would definitely be called intelligent, yet its artificial counterpart is a $2 pocket calculator that nobody would dare call AI. At the same time, we would never call someone intelligent just because they're able to drive in heavy traffic, yet a self-driving car is generally considered one of the toughest forms of AI the tech industry is working on today. We shouldn't be surprised by how hard defining *intelligence* is; after all, philosophers and scientists have been debating about it for centuries.

Not only do we have different weights to measure human and machine intelligence, but we also seem to be changing our mind pretty fast about what is AI and what isn't. Let's take an example from Paul Graham, founder of Y Combinator, the most successful Silicon Valley startup accelerator, and arguably one of the most forward-looking people in tech. In 2002, Graham wrote an essay

proposing a new solution to detect spam emails. Back then, email was just getting off the ground, and spam (unwanted email) was one of the most serious threats to widespread use of the internet by nontechies. It seems hard to imagine now, but the best computer scientists were busy trying to write complex rules to let computers automatically sort through Viagra advertisements.

In his essay, Graham thought about a new ML-based approach that would learn to classify an email by processing thousands of "good" and spam emails. Paul's simple software learned to recognize spam better than the complex rules concocted by engineers. Fast-forward 20 years, and automatic spam detectors are such a boring technology that we would be laughed out of the room if we dared call it AI.

In fact, it seems like AI is about mastering tasks that our imagination suggests computers shouldn't be able to do. Once we get used to a technology in our daily life, we remove the AI badge of honor and start calling it just computer software. This is a well-studied phenomenon called the *AI effect* .

Because of the AI effect, the goalposts for what we call AI keep moving just as quickly as technology improves. The definition of AI we draw from these considerations is "a temporary label to a piece of software that does something cool and surprising, until we get used to it." We don't know about you, but that just doesn't feel like a satisfying definition.

We hope we have convinced you that it is extremely hard to find a definition that makes everyone happy and can be valid as technology evolves. With the AI effect in mind, we decided to avoid a narrow definition of AI that rewards "flashy" applications just to ditch them once the hype is gone. We embrace a broader definition that includes less flashy applications. This is our definition of AI:

Software that solves a problem without explicit human instruction.

As you can see, our definition focuses on the outcome of the technology rather than the specific techniques used to build it. The truth is, learning *is* an intelligent trait, and while ML is just a tool,

it is *the* tool behind 99% of the successful applications we happen to call AI today. This may change in the future, but we don't see any new approaches on the horizon that hold the same promise as ML.

We now have a clear view of what ML is, a working definition of modern AI, and some perspective about how these terms evolved. We are just missing the third buzzword you've probably heard about: data science.

Data science (*DS*) is a broad, multidisciplinary field that uses scientific methods and processes to analyze data and extract insights. ML techniques are some of the tools in the DS toolbox. In practice, when people refer to a *data science project* , they often mean something *static* : extracting insights from data and presenting them as a presentation or report. On the other hand, AI is more commonly used in the context of live software.

For instance, analyzing traffic data to design a new urban plan for a city to minimize congestion likely falls into the realm of data science. However, if you use the same data to control traffic in real time and direct cars through less-congested routes, most people would say the project is about AI. In the first case, the output of your project is a report, and in the second, it's "live" software that runs 24/7. Keep in mind that this division is mostly conventional: there really are no hard-and-fast rules about what's AI and what's data science. Table 1 summarizes the differences as we see them.

Table 1: The main differences between AI and data science

Artificial intelligence	Data science
Automates tasks or predicts future events based on data.	Produces insights based on data.
Is commonly used "live": it continuously elaborates new data and produces answers.	Is commonly "one-off": it produces some insights that inform decisions.
It commonly has the form of software.	It commonly has the form of a presentation or report.

Hopefully, these sections helped demystify some commonly misunderstood terms and created context for these technologies. Now you can start learning the core principles of AI, what you can potentially do with it, and how to bring this transformative technology into your organization. In the next section, we'll explain the steps of this journey and how this book guides you through them.

6.3 APPLICATION DOMAIN

Artificial neural networks allow modeling of nonlinear processes and become a useful tool for solving many problems such as classification, clustering, dimension reduction, regression, structured prediction, machine translation, anomaly detection, pattern recognition, decision-making, computer vision, visualization, and others. This wide range of abilities makes it possible to use artificial neural networks in many areas. Recent developments in AI techniques complimented by the availability of high computational capacity at increasingly accessible costs, wide availability of labeled data, and improvement in learning techniques result in exploring the wide application domain for AI.

AI improves lives of human beings by assisting in driving, taking personal care of aged /handicap people, executing arduous and dangerous tasks, assisting in making informed decisions, rationally managing huge amounts of data that would otherwise be difficult to interpret, assisting in translating, and communicating multilingually while not knowing the language of our interlocutors and many more.

Artificial intelligence is already everywhere and is widely used in ways that are quite obvious. Some of the areas currently on the priority list include but not limited to:

Collaborative systems: Research on collaborative systems investigates models and algorithms to support the development of autonomous systems that can collaborate with each other and with human beings.

Computer vision: Till the advent of computer vision, support-vector machines were considered the most used method for visual classification activities and were the most relevant form of machine perception. Further, deep learning has deep impact on computer vision which is complimented by the evolution and low-cost availability of large-scale computing and the availability of large amounts of data. Moreover, the fine-tuning of networks of neural network algorithms has allowed the AI to perform visual classification tasks better than human beings.

Crowd sourcing and human computation: It is focused on the creation of innovative ways to exploit human intelligence.

Deep learning (DL): The ability to learn convolutional neural networks has brought many benefits to the computer vision sector, with applications such as object recognition, video labeling, and other variants.

Internet of things (IoT): Artificial intelligence plays a growing role in IoT applications and deployments. The value of AI in this context is its ability to quickly wring insights from data. Moreover, machine learning brings the ability to automatically identify patterns and detect anomalies in the data that smart sensors and devices generate. Other AI technologies such as speech recognition and computer vision can help extract insight from data that used to require human review. AI plays a growing role in IoT applications and deployments and is making a big splash in the Internet of things.

Machine learning (ML): Many basic problems in machine learning (such as supervised and non-supervised learning) are well known. A central focus in current studies concerns the chance of increasing the ability of algorithms to work on extremely large datasets.

Natural language processing (NLP): It is a very dynamic sector in the area of machine perception which is majorly associated with automatic speech recognition. It is nothing but imparting the ability to understand human language as it is spoken to computer

program. Research in this area is basically focused on the ability to develop systems capable of interacting with people through dialog and not with simple standard reactions which find application in enterprise search which involves organized retrieval of structured and unstructured data within an organization.

Neuromorphic computing: Traditional computers use von Neumann's architecture model. With the success of the deep neural networks, alternative models are being developed, many of which are inspired by neural biological networks.

Reinforcement learning: Through rule extraction, pattern matching, and mining, machine learning become one of the important tools which is further complimented by motivational decision-making capability implemented via reinforcement learning. Advent of reinforcement learning sharpens the ability of AI to address the real-world dynamic problem of complex nature.

Robotics: Navigation of robots in static environments is widely addressed and resolved. Now studies are revolving around exploring their ability to interact with the surrounding reality in a predictable way in dynamic environment in real time.

This is just a partial list of exhaustive application domain area of artificial intelligence where it can be used extensively. One area which was explored during PhD thesis, illustrated next, is the design of fuzzy inference system (FIS)-based adaptive hardware task scheduler for multiprocessor systems.

6.4 OUR TEACHING METHOD

If you want to productively use AI in your work life, it's paramount that you understand its nuts and bolts first. We noticed that nontechnical people who approach AI without a solid understanding of its principles often end up dreaming about projects that are simply impossible to build, or miss low-hanging fruit that could be easily tackled. After the first part of the book,

you'll know all the AI principles you need to avoid these dead ends and get the best out of the technology.

When presenting cases, we followed a methodology inspired by the Harvard Business School case method: we'll first present the case in the most neutral way possible, and ask you open-ended questions at the end. Right after that, we include our thoughts about these questions and prompts for further discussion. We recommend you don't read these answers right away, but rather try thinking about how *you* would answer based on your knowledge and what you've read in the case, and only then read our take. Be aware that there's no unique solution to the questions we asked: if you found an interesting take on the cases that we didn't include in the answers, good job! This means you've learned what you needed and are able to extract insights on your own (so if that happens, we reached our goal with this book as well).

6.5 MACHINE LEARNING: SOFTWARE THAT LEARNS THROUGH TRAINING

What if the burden of finding solutions to complex problems can be transferred from the programmer to their program? This is the promise of modern AI.

Excitement regarding modern AI relates to a set of techniques called machine learning, where advances have been significant and rapid. Machine learning is a sub-set of AI (Fig. 2). All machine learning is AI, but not all AI is machine learning.

Machine learning shifts much of the burden of writing intelligent software from the programmer to their program, enabling more complex and subtle problems to be solved. Instead of codifying rules for programs to follow, programmers enable programs to learn. Machine learning is the "field of study that gives computers the ability to learn without being explicitly programmed" (Arthur Samuel).

Machine learning algorithms learn through training. In a simplified example, an algorithm is fed inputs – training data – whose outputs are usually known in advance ('supervised learning'). The algorithm processes the input data to produce a prediction or recommendation. The difference between the algorithm's output and the correct output is determined. If the algorithm's output is incorrect, the processing function in the algorithm changes to improve the accuracy of its predictions. Initially the results of a machine learning algorithm will be poor. However, as larger volumes of training data are provided, a program's predictions can become highly accurate (Fig. 3).

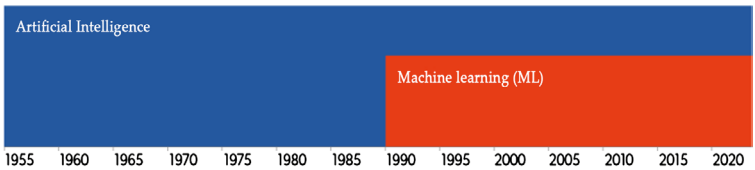


Figure 2: The Evolution of AI: machine learning

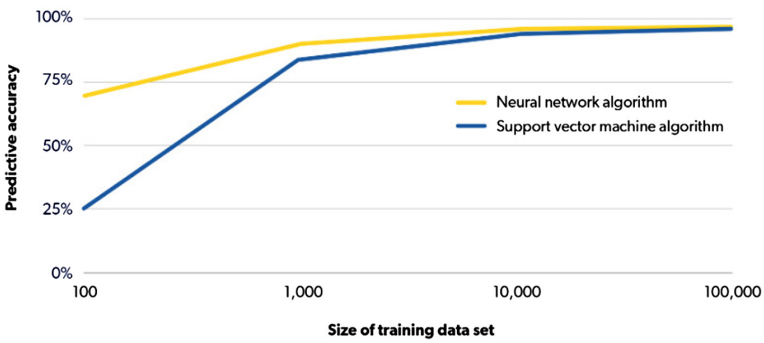


Figure 3: Large data sets enable effective machine learning

The defining characteristic of a machine learning algorithm, therefore, is that the quality of its predictions improves with experience. Typically, the more relevant data provided to a machine learning system, the more effective its predictions (up to a point).

By learning through practice, instead of following sets of rules, machine learning systems deliver better solutions than rules-based systems to numerous prediction and optimisation challenges.

There are many approaches to machine learning

There are more than 15 approaches to machine learning. Each uses a different form of algorithmic architecture to optimise predictions based on input data.

One, deep learning, is delivering breakthrough results in new domains. We explain deep learning below. Others receive less attention – but are widely used given their utility and applicability to a broad range of use cases. Popular machine learning algorithms beyond deep learning include:

- Random forests that create multitudes of decision trees to optimise predictions. Random forests are used by nearly half of data scientists (Kaggle).
- Bayesian networks that use probabilistic approaches to analyse variables and the relationships between them. One third of data scientists use Bayesian networks (Kaggle).
- Support vector machines that are fed categorised examples and create models to assign new inputs to one of the categories. A quarter of data scientists employ support vector machines (Kaggle).

Each approach offers advantages and disadvantages. Frequently, combinations are used (an 'ensemble' method). In practice, developers frequently experiment to determine what is effective.

Machine learning can be applied to a wide variety of prediction and optimisation challenges. Examples include: assessing whether a credit card transaction is fraudulent; identifying products a person is likely to buy given their prior purchases; and predicting when an industrial asset is likely to experience mechanical failure.

Deep Learning: Offloading Feature Specification

Even with the power of general machine learning, it is difficult to develop programs that perform certain tasks well – such as understanding speech or recognising objects in images.

In these cases, programmers cannot specify the features in the input data to optimise. For example, it is difficult to write a program that identifies images of dogs. Dogs vary significantly in their visual appearance. These variations are too broad to be described by a set of rules that will consistently enable correct classification (Fig. 4). Even if an exhaustive set of rules could be created, the approach would not be scalable; a new set of rules would be required for every type of object we wished to classify.

Deep learning is delivering breakthrough results in these use cases. Deep learning is a sub-set of machine learning and one of many approaches to it (Fig. 5). All deep learning is machine learning, but not all machine learning is deep learning.

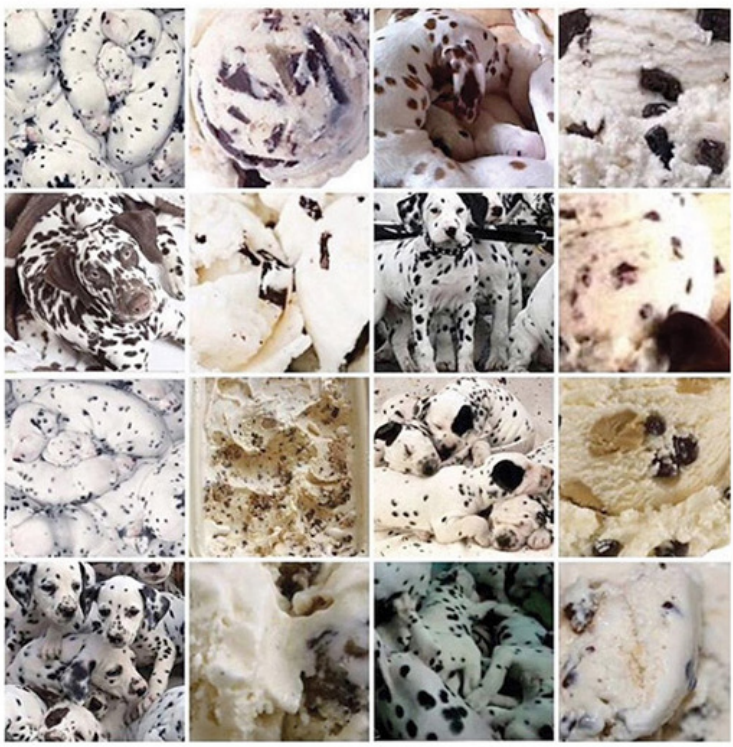


Figure 4: Identifying features can be difficult ('Dalmatians or ice cream?').

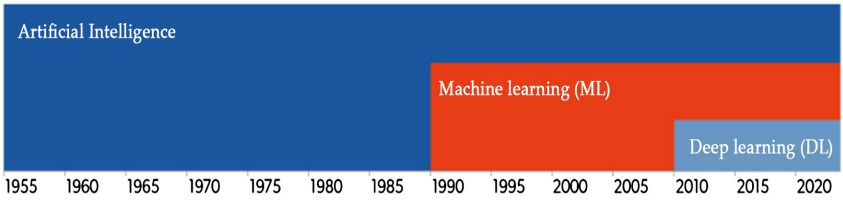


Figure 5: The Evolution of AI: deep learning.

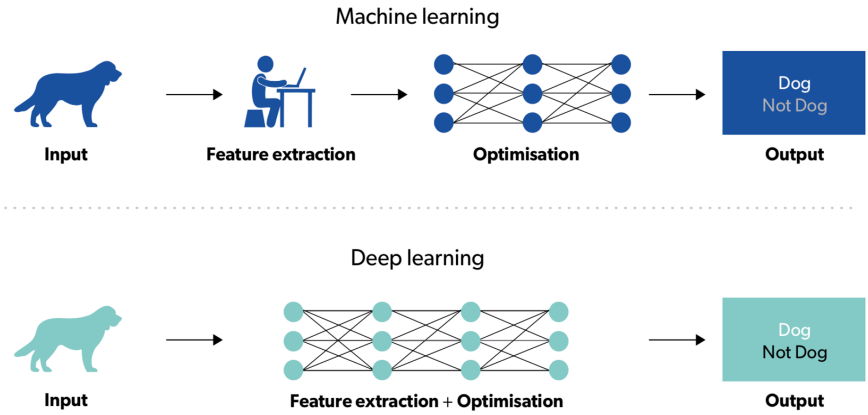


Figure 6: Deep learning offloads the burden of feature extraction from a programmer to their program

Deep learning is valuable because it transfers an additional burden – the process of feature extraction – from the programmer to their program (Fig. 6).

Humans learn to complete subtle tasks, such as recognising objects and understanding speech, not by following rules but through practice and feedback. As children, individuals experience the world (see a dog), make a prediction ('dog') and receive feedback. Humans learn through training.

Deep learning works by recreating the mechanism of the brain (Fig. 7) in software (Fig. 8). With deep learning we model the brain, not the world.

To undertake deep learning, developers create artificial neurons – software-based calculators that approximate, crudely, the function

of neurons in a brain. Artificial neurons are connected together to form a neural network. The network receives an input (such as a picture of a dog), extracts features and offers a determination. If the output of the neural network is incorrect, the connections between the neurons adjust to alter its future predictions. Initially the network's predictions will frequently be incorrect. However, as the network is fed many examples (potentially, millions) in a domain, the connections between neurons become finely tuned. When analysing new examples, the artificial neural network will then make consistently correct determinations.

Figure 7: A biological neural network

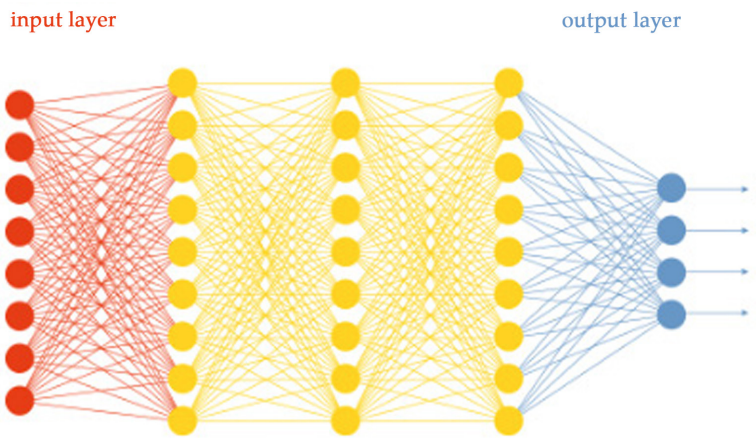


Figure 8: An artificial neural network

Deep learning has unlocked significant new capabilities, particularly in the domains of vision and language. Deep learning enables:

- autonomous vehicles to recognise entities and features in the world around them (Fig. 9);
- software to identify tumours in medical images;
- Apple and Google to offer voice recognition systems in their smartphones;
- voice-controlled devices, such as the Amazon Echo;
- real-time language translation (Fig. 10);
- sentiment analysis of text; and more.

Deep learning is not suited to every problem. Typically, deep learning requires large data sets for training. Training and operating a neural network also demand extensive processing power. Further, it can also be difficult to identify how a neural network developed a specific prediction – a challenge of 'explainability'.

However, by freeing programmers from the burden of feature extraction, deep learning has delivered effective prediction engines for a range of important use cases and is a powerful tool in the AI developer's arsenal.

Figure 9: Deep learning enables autonomous vehicles to identify objects around them

Figure 10: Google's Pixel Buds use deep learning to provide real-time language translation.

How does deep learning work?

Deep learning involves creating artificial neural networks – software-based calculators (artificial neurons) that are connected to one another.

An artificial neuron (Fig. 11) has one or more inputs. The neuron performs a mathematical function on its inputs to deliver an output. The output will depend on the weights given to each input, and the configuration of the input-output function in the neuron. The input-output function can vary. An artificial neuron may be a:

- **linear unit** (the output is proportional to the total weighted input);
- **threshold unit** (the output is set to one of two levels, depending on whether the total input is above a specified value);
- **sigmoid unit** (the output varies continuously, but not linearly as the input changes).

An artificial neural network (Fig. 12) is created when artificial neurons are connected to each other. The output of one neuron becomes an input for another.

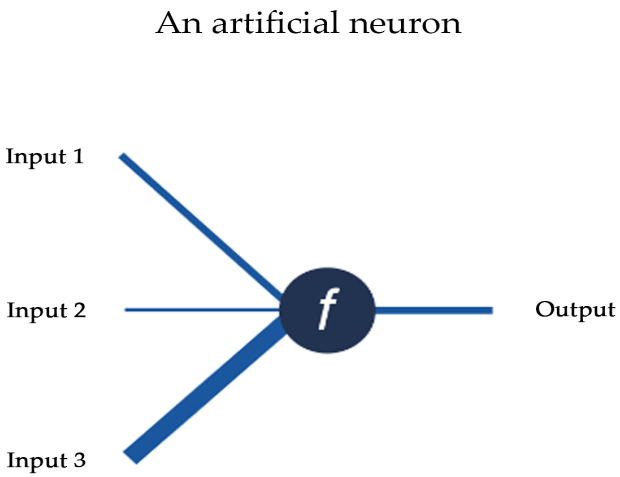


Figure 11: An artificial neuron

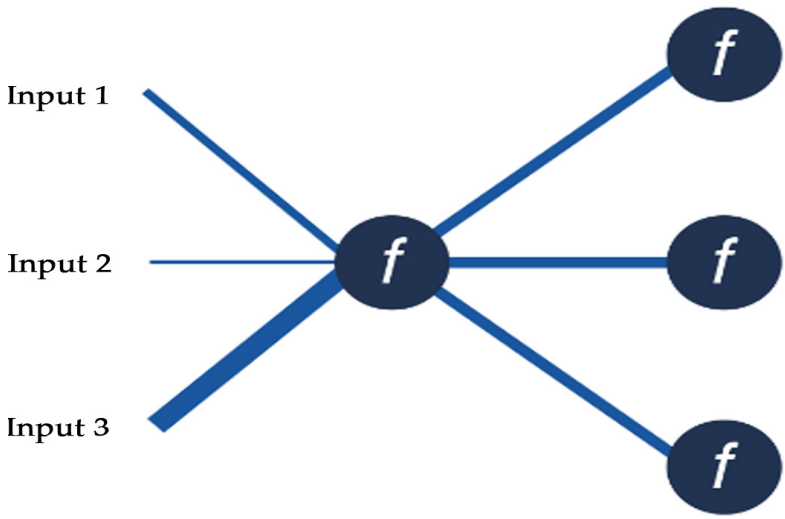


Figure 12: An artificial neural network

Neural networks are organized into multiple layers of neurons (Fig. 13) – hence 'deep' learning. An input layer receives information to be processed, such as a set of pictures. An output layer delivers

results. Between the input and output layers are layers referred to as 'hidden layers' where features are detected. Typically, the outputs of neurons on one level of a network all serve as inputs to each neuron in the next layer.

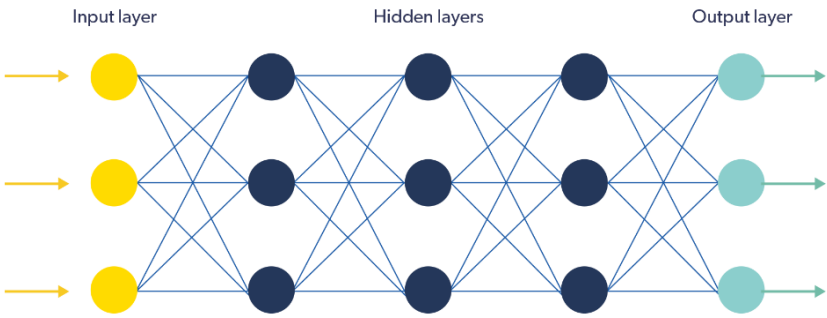


Figure 13: Deep learning: structuring an artificial neural network

Fig. 14 illustrates a neural network designed to recognize pictures of human faces. When pictures are fed into the neural network, the first hidden layers identify patterns of local contrast (low-level features such as edges). As images traverse the hidden layers, progressively higher-level features are identified. Based on its training, at its output layer the neural network will deliver a probability that the picture is of a human face.

Typically, neural networks are trained by exposing them to a large number of labelled examples. As errors are detected, the weightings of the connections between neurons adjust to offer improved results. When the optimization process has been repeated extensively, the system is deployed to assess unlabelled images.

The structure and operation of the neural network below is simple (and simplified), but structures vary and most are more complex.

It takes considerable skill to design and improve a neural network. AI professionals undertake multiple steps including: structuring the network for a particular application; providing suitable

training data; adjusting the structure of the network according to progress; and combining multiple approaches to optimize results.

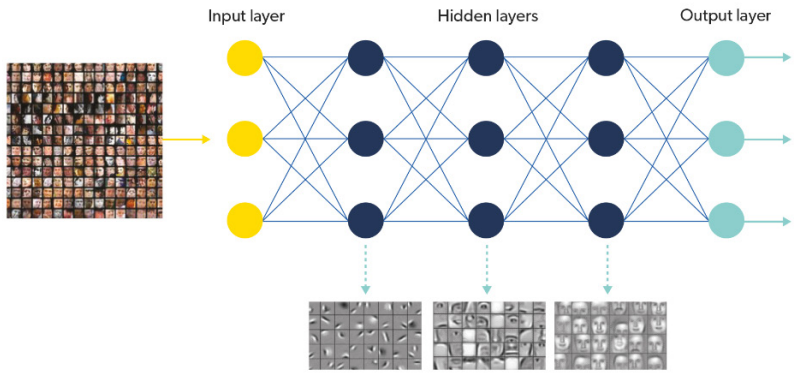


Figure 14: Deep learning: the process of feature extraction

6.6 IMPORTANCE OF AI

AI is important because, for the first time, traditionally human capabilities can be undertaken in software inexpensively and at scale. AI can be applied to every sector to enable new possibilities and efficiencies.

6.6.1 AI tackles profound technical challenges

AI is significant because it successfully tackles a profound set of technical challenges. Increasingly, human capabilities – understanding, reasoning, planning, communication and perception – can be undertaken by software, at scale and at low cost. General analytical tasks, including finding patterns in data, that have been performed by software for many years can also be performed more effectively using AI.

Together, these capabilities create new opportunities in most business processes and consumer applications.

AI is enabling new possibilities			
Knowledge	Medical diagnosis	Drug creation	Media recommendation
	Financial trading	Information synthesis	Consumer targeting
Reasoning	Legal analysis	Asset management	Application processing
	Games	Autonomous weapons	Compliance
Planning	Logistics	Fleet management	Navigation
	Network optimisation	Predictive maintenance	Demand forecasting
Communication	Voice control	Intelligent agents	Customer support
	Real-time transcription	Real-time translation	Client service
Perception	Autonomous vehicles	Medical imaging	Authentication
	Augmented reality	Surveillance	Industrial analysis

6.6.2 The applications of AI in industry are numerous and tangible

AI is not a solution seeking a problem; it is a tangible set of capabilities unlocking revenue growth and cost savings. The capabilities of AI – its power to incorporate broader data sets into analyses, identify concepts and patterns in data better than rules-based systems, and enable human-to-machine conversation – have applications in all sectors and numerous business processes. In approximately 60% of occupations, at least 30% of constituent activities are technically automatable by adapting currently proven AI technologies (McKinsey Global Institute). As such, AI is a key 'enabling technology'.

6.6.3 Data-centric sectors will see the greatest impact

AI is being deployed in all sectors and to a wide variety of business processes. However, AI will have more numerous applications and greater impact in certain sectors.

AI's impact will be greatest in sectors in which a large proportion of time is spent collecting or synthesising data, or undertaking predictable physical work. In several sectors (Fig. 15), professionals spend one third or more of their time on the above (McKinsey, Julius Baer).

These sectors include:

- Finance and insurance (50% of time)
- Retail, transport and trade (40% of time)
- Professional services (37% of time)
- Manufacturing (33% of time)
- Healthcare (33% of time)

Applications will be more limited in sectors in which data synthesis and processing activities are limited, or in which the majority of people's time is spent managing others or undertaking unpredictable physical work. Occupations such as management and teaching will be more resilient to AI in the medium term.

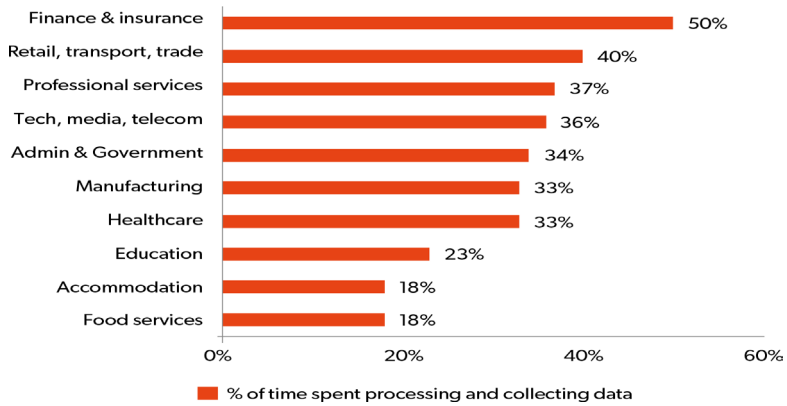


Figure 15: In several sectors, professionals spend a third or more of their time collecting or synthesising data

6.6.4 Asset management

AI's ability to extract content from unstructured data using natural language processing, find subtle patterns in disparate data sets, and enable machine-to-human communication via chatbots, has multiple applications in asset management. Core use cases include investment strategy, portfolio construction, risk management and client service.

By augmenting or automating many of an asset manager's tasks, AI enables asset managers to deliver to the mass affluent a degree of personalisation and service quality reserved for high net worth clients. Additionally, AI can decrease the cost of portfolio construction while improving quality – the era of the 'robo-advisor'.

Investment strategy: AI can improve a firm's investment strategy by synthesising its research and data, and incorporating broader data sets including unstructured information. Superior pattern recognition can then deliver better multi-objective optimisation. AI can balance a diverse range of inter-connected objectives (including fund deployment, risk and profitability) to enhance returns more effectively than rules-based systems.

Portfolio construction: AI tools can augment, and increasingly automate, an asset manager's process of portfolio construction. AI – 'robo-advisors' – can analyse a client's goals, and within a firm's investment rules develop personalised, optimised portfolios at low cost and high speed.

Risk management: AI can improve risk management by incorporating broader data sets and improving analytical processing. 90% of data generated today is unstructured information, stored outside traditional databases (International Data Group). Natural language processing enables additional data sets to be incorporated into firms' analyses. Other AI techniques, including deep learning, then enable patterns in data to be identified with greater granularity and confidence. Together,

these capabilities enable risks to be identified and quantified more effectively.

Client service: Chatbot interfaces are being applied within and beyond asset management firms. Deployed in clientfacing channels, natural language systems enable client enrolment, support and self-service. Embedded in internal tools, chatbots let account managers query client details and understand developments relevant to a client's portfolio in seconds instead of minutes. Fewer account managers can then provide a higher quality service to a greater number of clients.

6.6.5 Healthcare

In the next decade, AI can unlock a paradigm shift in healthcare to improve patient care and process efficiency. Automated diagnosis was an early use case for rudimentary AI in the 1980s. 'Expert systems' mimicked human approaches to diagnosis, applying rules-based inferences to bodies of knowledge. Modern AI, particularly deep learning, is more effective and applicable to a wider range of processes. Key use cases include diagnosis, drug discovery and patient monitoring.

Diagnosis: Deep learning systems can replace complex, human-coded sets of probabilistic rules and identify subtle correlations between vast, multi-variate data sets to deliver scalable, automated diagnosis. While systems are nascent, accuracy is improving rapidly. Separately, computer vision solutions powered by deep learning are transforming diagnostic imaging. While human radiologists require extensive expertise and years of training to identify abnormalities in magnetic resonance images and ultrasounds, deep learning systems trained on large data sets deliver impressive results. Diagnostic imaging, powered by deep learning, now offers human-level accuracy and high speed in select contexts.

Drug discovery: Today's drug discovery process is lengthy, averaging 12 years to market (California Biomedical Research

Association). Expense and uncertainty are also prohibitive; drug development costs an average of $359m and just 2% of US preclinical drugs are approved for human use (California Biomedical Research Association). AI is being applied to multiple stages of the drug development process to accelerate time to market and reduce uncertainty. AI is being applied to synthesize information and offer hypotheses from the 10,000 research papers published daily, predict how compounds will behave from an earlier stage of the testing process, and identify patients for clinical trials.

6.6.6 Insurance

While the fundamentals of insurance – customer prospecting, risk assessment, claims processing and fraud detection – have remained unchanged, modern AI can improve every stage in the insurance process to deliver efficiency savings and improved customer experience. By identifying patterns in data better than rules-based systems, AI can improve and accelerate decision-making and claims processing, reduce fraud and automate a large proportion of customer service enquiries.

Risk assessment: AI can gather information from broader data sets, including web and social media profiles, to compile richer customer information and inform risk assessment. AI can then assess the risk of individual policies more accurately than rules based systems, by detecting non-linear patterns in multi-variety data sets and making more accurate projections.

Claims processing: AI can reduce time-to-quote, time-to-claim and claims processing costs for consumers and insurers. To accelerate claims processing, AI systems can automatically extract and classify structured and unstructured information from insurance policies and claim forms. By analyzing images of damaged assets, computer vision systems can automatically classify claims. Through improved pattern recognition applied to prior cases, AI can also predict settlement costs. Algorithms using deep learning are effective for image analysis, while Bayesian (probability-based) AI is useful for predicting settlement costs.

Fraud detection: Insurance fraud costs UK insurers £1.3bn annually and adds £50 to the average policyholder's annual bill (Association of British Insurers). UK insurers invest over £200m annually to tackle the challenge (Association of British Insurers). Fraud detection algorithms enhanced with AI can identify fraudulent transactions, while reducing false positives, more effectively than traditional approaches.

Customer service: Chatbot interfaces integrated with insurers' databases can use natural language processing to offer 24/7 product information and answers to policyholders' enquiries in a scalable, inexpensive and personalized channel.

6.6.7 Law and Compliance

AI's abilities to process language in documents, synthesize knowledge and automate reasoning have broad application in the legal services and compliance sector. With junior lawyers spending a high proportion of their time accessing and collating information, scope for augmentation and automation is considerable. Key AI use cases include identifying relevant case law, processing documents for discovery and due diligence, and informing litigation strategy. In October 2018, Harvard Law School Library advanced its 'Caselaw Access Project' by releasing over 40 million pages of digitized legal information, including every reported state and federal legal case in the United States from the 1600s to the summer of 2017 – providing extensive further data to train AI systems.

Regarding compliance, costs have grown significantly since 2008 – particularly for financial services firms. With rules-based software poorly suited to catching infractions, banks have invested in additional compliance personnel. Citi, while reducing its global headcount 32% between 2008 and 2016, doubled its regulatory and compliance staff to 29,000 – over 13% of it workforce (Citi). AI's ability to learn patterns of behavior over time, and highlight unusual activity in real-time, offers greater scalability at lower cost.

Case law, discovery and due diligence: Natural language processing AI can identify, classify and utilise content from databases and unstructured documents at scale and speed, saving legal firms time and cost for routine document review. Use cases include sourcing and ranking relevant case law and identifying key documents in due diligence and discovery processes. With a merger and acquisition data room containing an average of 34,000 pages for review (Luminance), AI can increase business velocity and reduce costs.

Litigation strategy: AI can analyses past judgments at greater speed, granularity and subtlety than has been possible. By anticipating the probability of different outcomes, lawyers can inform and enhance their strategic decision-making. In high volume areas, such as personal injury, software can help a firm decide whether to accept a case. In high value areas, including corporate litigation, software can suggest the probability of a particular outcome based on juries' prior behavior and opposing lawyers' tendency to settle or proceed to trial.

Compliance: Preventing accidental or deliberate breaches of policy, from the theft of sensitive data to accidentally misaddressing an email containing a customer database, is challenging for rules-based systems. By learning the habits of users over time, AI systems can flag potential compliance breaches in real-time, before they occur, with sufficient accuracy to enable broad deployment.

6.6.8 Manufacturing

AI can significantly improve manufacturers' efficiency and profitability. Overall Equipment Effectiveness (OEE), a measure of manufacturers' productivity relative to potential, varies widely by industry, from 75%-91% (LNS Research). The performance of companies within the same industry also varies widely, offering scope for competitive advantage. AI can boost OEE and profitability by predicting equipment failure (to reduce unplanned downtime), improving assets' operational efficiency, and reducing utility supply costs.

Predictive maintenance: The failure of production assets is costly; one hour of unplanned downtime on an automotive assembly line can cost a manufacturer £1.5m (MMC Ventures). AI can identify subtle patterns in data from vibration, temperature, pressure and other sensors to identify leading indicators of equipment failure. By predicting more accurately which components are likely to fail, and when, parts can be proactively replaced to prevent failures and save money.

Asset performance: AI can improve the operation of high value assets, including gas and wind turbines, to optimize yield. Rules-based programs deliver limited results when applied to complex tasks, such as tuning fuel valves on a gas turbine to optimize combustion while reducing wear and emissions. Applying neural networks to optimize the turbine inputs can improve results by 20% or more.

Utility optimization: Optimizing the purchase and consumption of utilities, such as power and water, according to real-time demands on a factory floor is too challenging and variable to manage using rules-based software. AI enables companies to anticipate, and align, utility consumption with process requirements in real time, lowering utility consumption by 5% or more.

6.6.9 Retail

E-commerce, now 17% of UK retail sales and growing (eMarketer), has transformed the quantity, breadth and granularity of data available to retailers. Retailers that turn data into insight can increase competitive advantage by engaging, monetizing and retaining customers more effectively. Every stage of a retailer's customer journey – from lead generation and content selection to price optimization and churn prediction – can be improved by AI algorithms that ingest richer data sets and identify patterns in them better than rules-based systems. By enabling analytics at the 'per-customer' level, AI is introducing the era of retail personalization. Leaders enjoy competitive advantage; 75% of Netflix users select

films recommended to them by the Company's AI algorithm (Netflix).

Customer segmentation: Limitations in available data, and the linear analysis of information, inhibit the ability of traditional customer segmentation software to identify desirable customer attributes. Deep learning algorithms enable natural language processing, which enables retailers to access additional data sets including social media data. Deep learning algorithms also offer more granular analysis than rules-based systems, to optimize segmentation, channel selection and messaging.

Content personalization: Most content presented to online shoppers is irrelevant or poorly suited to users' preferences, reducing conversion to an average of 1.0% on smartphones and 2.8% on desktops (Adobe). As with customer segmentation, AI offers additional unstructured data sets for analysis, and improved multivariate analysis to identify more subtle correlations than rules-based systems can detect. When Netflix recommends content to a user, in addition to analyzing a user's actions, ratings and searches, the Company's AI algorithm considers social media data and meta-data from third parties. The Company is now analyzing images from content, including color palette and scenery, for deeper personalization.

Price optimization: A 1% change in price provides, on average, a 10% change in profitability (Blue Yonder). The smaller a company's margins, the greater the impact. Willingness to pay is a key determinant for price. AI enables price optimization that is more sophisticated than traditional 'cost plus', 'relative-to-competitors' or 'odd pricing' (£0.99) models. By identifying correlations within and between data sets, AI can better optimize for factors including price elasticity, revenue, profit, product availability and phases in a product's lifecycle (introduction or end-of-life).

Churn prediction: Traditional programs struggle to incorporate new sources of information, maximize the value from multi-variety data sets or offer granular recommendations. AI-powered churn prediction can identify leading indicators of churn more

effectively, and improve remediation by predicting more accurately the format and content of successful interventions.

6.6.10 Transport

The transport sector will be transformed by AI. Breakthroughs in computer vision are enabling the age of autonomous vehicles – self-driving cars, buses and trucks. The implications, from shifts in sector value chains to new business models, will be profound. In addition to enabling autonomy, AI can be applied to the many prediction and optimisation challenges – from congestion modelling to fleet management – at the core of today's logistics networks.

Autonomous vehicles: AI computer vision systems enable vehicles to sense and identify the physical features and dynamics of their environment, from road lanes to pedestrians and traffic lights, with a high degree of accuracy. Combined with AI data processing and planning algorithms, AI is enabling the age of autonomous transport. Cars, buses and trucks will be able to operate and guide themselves, without human involvement. SAE International, a US-based global professional association and standards body, has identified five degrees of vehicle autonomy, from Level 0 (no automation) to Level 5 (full automation; no requirement for human control).

Select companies, including Google, intend to release vehicles offering Level 5 automation. Challenged by the autonomous vehicle programmes of Google, Uber and Tesla, incumbent manufacturers are accelerating their own initiatives by increasing investment and making acquisitions. Ford intends to deliver high-volume availability of at least a Level 4 autonomous vehicle by 2021. In October 2018, in the UK, private hire firm Addison Lee announced its intention to deploy self-driving cars in London by 2021, by partnering with UK autonomy company Oxbotica.

Infrastructure and system optimization: AI's abilities to detect patterns and optimise complex data are being applied to traffic,

congestion and infrastructure challenges in transport systems. Predicting traffic flows, or modelling the deterioration of transport infrastructure, are difficult because inputs are complex (combining traffic, construction and environmental data) and because the relationships between inputs and outputs are non-linear (Transportation Research Circular). In these contexts, machine learning and deep learning systems are well suited to deliver better results than rules-based systems.

Fleet management: Transportation fleets are pervasive, from the logistics networks that underpin the economy to taxi fleets and food delivery services that provide point-topoint convenience. AI can optimise pick-ups, route planning and delivery scheduling to maximise asset utilisation, while considering economic, social and environmental impacts.

Control applications: Machine learning systems are well suited to the numerous prediction and optimisation challenges presented by air traffic control, vehicle traffic signalling, and train control

6.6.11 Utilities

Information processing will become critical to utility companies, and their business models, as the utility sector undergoes a greater change in the next 25 years than it has during the 150. 'Prosumers' – consumers who also own capacity for energy production – will require integration into the energy market. By processing data more intelligently, AI will be a significant value driver in this transition. AI use cases for utility companies are varied, from demand optimization and security to customer experience.

The foundations for AI adoption in the utilities sector are robust. 67% of utility companies – a higher proportion than in any other sector – use 'internet of things' (IoT) technologies such as sensors (Gartner). Further, compared with peers in other sectors, utility CIOs have a stronger focus on cost reduction, managing geographically dispersed assets and security.

Supply management: AI algorithms can predict changes in supply, including those caused by the intermittency of renewable resources, more effectively than rules-based systems – enabling smaller reserves and greater cost savings. AI solutions can also optimise supply networks, which are becoming increasingly complex as consumers deploy sources of renewable energy that contribute energy back to the National Grid.

Demand optimization: By identifying detailed patterns in consumer behavior, AI algorithms can move consumption of energy from periods of peak use and high prices to times of lower demand and cost.

Security: Rules-based systems struggle to deliver system security given the continually evolving nature of security threats. By identifying abnormal patterns in network behavior, deep learning systems can identify breaches in network security that elude traditional programs.

Customer experience: Chatbots, which offer natural language conversations powered by deep learning algorithms, offer consumers self-service account administration, product information and customer service.

REFERENCES

1. Asada, M.; Hosoda, K.; Kuniyoshi, Y.; Ishiguro, H.; Inui, T.; Yoshikawa, Y.; Ogino, M.; Yoshida, C. (2009). "Cognitive developmental robotics: a survey". IEEE Transactions on Autonomous Mental Development. 1 (1): 12–34. doi:10.1109/tamd.2009.2021702. S2CID 10168773.
2. Bach, Joscha (2008). "Seven Principles of Synthetic Intelligence". In Wang, Pei; Goertzel, Ben; Franklin, Stan (eds.). Artificial General Intelligence, 2008: Proceedings of the First AGI Conference. IOS Press. pp. 63–74. ISBN 978-1-58603-833-5. Archived from the original on 8 July 2016. Retrieved 16 February 2016.
3. Brooks, R. A. (1991). "How to build complete creatures rather than isolated cognitive simulators". In VanLehn, K. (ed.). Architectures for Intelligence. Hillsdale, NJ: Lawrence Erlbaum Associates. pp. 225–239. CiteSeerX 10.1.1.52.9510.
4. Brooks, Rodney (1990). "Elephants Don't Play Chess" (PDF). Robotics and Autonomous Systems. 6 (1–2): 3–15. CiteSeerX 10.1.1.588.7539. doi:10.1016/S0921-8890(05)80025-9. Archived (PDF) from the original on 9 August 2007.
5. Buchanan, Bruce G. (2005). "A (Very) Brief History of Artificial Intelligence" (PDF). AI Magazine: 53–60. Archived from the original (PDF) on 26 September 2007.
6. Butler, Samuel (13 June 1863). "Darwin among the Machines". Letters to the Editor. The Press. Christchurch, New Zealand. Archived from the original on 19 September 2008. Retrieved 16 October 2014 – via Victoria University of Wellington.
7. Clark, Jack (1 July 2015a). "Musk-Backed Group Probes Risks Behind Artificial Intelligence". Bloomberg.com. Archived from the original on 30 October 2015. Retrieved 30 October 2015.

CHAPTER 7

INFORMATION SECURITY AND PRIVACY (ISA)

INTRODUCTION

Data breaches are on the rise, making information security and privacy top priorities for business and IT leaders. Today's threats are more potent than ever, with employees cited as the primary risk. This trend, coupled with the expansion of data privacy laws around the world, has led to the growing realization that Enterprise Information Management solutions are must-have tools for data protection and regulatory compliance.

Information security is the practice of defending information – in all forms - from unauthorized access, use, examination, disclosure, modification, copying, moving, or destruction. There are numerous global and industry standards and regulations mandating information security practices for organizations.

Information privacy, or data privacy is the relationship between the collection and dissemination of data and the public expectation of privacy. The safeguarding of personal data is the objective i.e. data about individuals such as contact information, health, financial, and family information; these individuals could be your employees, your customers and other stakeholders. There are various legal, regulatory, political, and technological issues surrounding the issue of data privacy.

7.1 INFORMATION SECURITY

Information security (infosec) is a set of strategies for managing the processes, tools and policies necessary to prevent, detect, document and counter threats to digital and non-digital information. Infosec responsibilities include establishing a set of business processes that will protect information assets regardless of how the information is formatted or whether it is in transit, is being processed or is at rest in storage.

Many large enterprises employ a dedicated security group to implement and maintain the organization's infosec program. Typically, this group is led by a chief information security officer. The security group is generally responsible for conducting risk management, a process through which vulnerabilities and threats to information assets are continuously assessed, and the appropriate protective controls are decided on and applied. The value of an organization lies within its information -- its security is critical for business operations, as well as retaining credibility and earning the trust of clients.

Information Security is not only about securing information from unauthorized access. Information Security is basically the practice of preventing unauthorized access, use, disclosure, disruption, modification, inspection, recording or destruction of information. Information can be physical or electronic one. Information can be anything like Your details or we can say your profile on social media, your data in mobile phone, your biometrics etc. Thus Information Security spans so many research areas like Cryptography, Mobile Computing, Cyber Forensics, Online Social Media etc.

During First World War, Multi-tier Classification System was developed keeping in mind sensitivity of information. With the beginning of Second World War formal alignment of Classification System was done. Alan Turing was the one who successfully decrypted Enigma Machine which was used by Germans to encrypt warfare data.

Information Security programs are build around 3 objectives, commonly known as CIA – Confidentiality, Integrity, Availability.

- **Confidentiality –** means information is not disclosed to unauthorized individuals, entities and process. For example if we say I have a password for my Gmail account but someone saw while I was doing a login into Gmail account. In that case my password has been compromised and Confidentiality has been breached.

- **Integrity** – means maintaining accuracy and completeness of data. This means data cannot be edited in an unauthorized way. For example if an employee leaves an organisation then in that case data for that employee in all departments like accounts, should be updated to reflect status to JOB LEFT so that data is complete and accurate and in addition to this only authorized person should be allowed to edit employee data.
- **Availability** – means information must be available when needed. For example if one needs to access information of a particular employee to check whether employee has outstanded the number of leaves, in that case it requires collaboration from different organizational teams like network operations, development operations, incident response and policy/change management. Denial of service attack is one of the factor that can hamper the availability of information.

Apart from this there is one more principle that governs information security programs. This is Non repudiation.

- **Non repudiation** – means one party cannot deny receiving a message or a transaction nor can the other party deny sending a message or a transaction. For example in cryptography it is sufficient to show that message matches the digital signature signed with sender's private key and that sender could have a sent a message and nobody else could have altered it in transit. Data Integrity and Authenticity are pre-requisites for Non repudiation.
- **Authenticity** – means verifying that users are who they say they are and that each input arriving at destination is from a trusted source.This principle if followed guarantees the valid and genuine message received from a trusted source through a valid transmission. For example if take above example sender sends the message along with digital signature which was generated using the hash value of message and private key. Now at the

receiver side this digital signature is decrypted using the public key generating a hash value and message is again hashed to generate the hash value. If the 2 value matches then it is known as valid transmission with the authentic or we say genuine message received at the recepient side

- **Accountability –** means that it should be possible to trace actions of an entity uniquely to that entity. For example as we discussed in Integrity section not every employee should be allowed to do changes in other employee's data. For this there is a separate department in an organization that is responsible for making such changes and when they receive request for a change then that letter must be signed by higher authority for example Director of college and person that is allotted that change will be able to do change after verifying his bio metrics, thus timestamp with the user (doing changes) details get recorded. Thus we can say if a change goes like this then it will be possible to trace the actions uniquely to an entity.

At the core of Information Security is Information Assurance, which means the act of maintaining CIA of information, ensuring that information is not compromised in any way when critical issues arise. These issues are not limited to natural disasters, computer/server malfunctions etc.

Thus, the field of information security has grown and evolved significantly in recent years. It offers many areas for specialization, including securing networks and allied infrastructure, securing applications and databases, security testing, information systems auditing, business continuity planning etc.

7.2 THE SECURITY PROBLEM IN COMPUTING

Computer security refers to the security of computing devices such as computers and smartphones, as well as computer networks such as private and public networks, and the Internet. The field has growing importance due to the increasing reliance on computer

systems in most societies. It concerns the protection of hardware, software, data, people, and also the procedures by which systems are accessed. The means of computer security include the physical security of systems and security of information held on them.

- To prevent theft of or damage to the hardware
- To prevent theft of or damage to the information
- To prevent disruption of service

Computer security is security applied to computing devices such as computers and smartphones, as well as computer networks such as private and public networks, including the whole Internet. The field covers all the processes and mechanisms by which digital equipment, information and services are protected from unintended or unauthorized access, change or destruction, and are of growing importance in line with the increasing reliance on computer systems of most societies worldwide. It includes physical security to prevent theft of equipment, and information security to protect the data on that equipment. It is sometimes referred to as "cyber security" or "IT security", though these terms generally do not refer to physical security (locks and such).

Some important terms used in computer security are:

- ***Vulnerability***: Vulnerability is a weakness which allows an attacker to reduce a system's information assurance. Vulnerability is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw. To exploit vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness. In this frame, vulnerability is also known as the attack surface.
- Vulnerability management is the cyclical practice of identifying, classifying, remediating, and mitigating vulnerabilities. This practice generally refers to software vulnerabilities in computing systems.
- ***Backdoors***: A backdoor in a computer system, is a method of bypassing normal authentication, securing remote

access to a computer, obtaining access to plaintext, and so on, while attempting to remain undetected.

- The backdoor may take the form of an installed program (e.g., Back Orifice), or could be a modification to an existing program or hardware device. It may also fake information about disk and memory usage.
- ***Denial-of-service attack***: Unlike other exploits, denials of service attacks are not used to gain unauthorized access or control of a system. They are instead designed to render it unusable. Attackers can deny service to individual victims, such as by deliberately entering a wrong password enough consecutive times to cause the victim account to be locked, or they may overload the capabilities of a machine or network and block all users at once. These types of attack are, in practice, very hard to prevent, because the behavior of whole networks needs to be analyzed, not only the behavior of small pieces of code. Distributed denial of service (DDoS) attacks are common, where a large number of compromised hosts (commonly referred to as "zombie computers", used as part of a botnet with, for example; a worm, Trojan horse, or backdoor exploit to control them) are used to flood a target system with network requests, thus attempting to render it unusable through resource exhaustion.
- ***Direct-access attacks***: An unauthorized user gaining physical access to a computer (or part thereof) can perform many functions, install different types of devices to compromise security, including operating system modifications, software worms, key loggers, and covert listening devices. The attacker can also easily download large quantities of data onto backup media, for instance CD-R/DVD-R, tape; or portable devices such as key drives, digital cameras or digital audio players. Another common technique is to boot an operating system contained on a CD-ROM or other bootable media and read the data from the hard drive(s) this way. The only way to defeat this is to encrypt the storage media

and store the key separate from the system. Direct-access attacks are the only type of threat to Standalone computers (never connect to internet), in most cases.

- ***Eavesdropping:*** Eavesdropping is the act of surreptitiously listening to a private conversation, typically between hosts on a network. For instance, programs such as Carnivore and NarusInsight have been used by the FBI and NSA to eavesdrop on the systems of internet service providers.
- ***Spoofing:*** Spoofing of user identity describes a situation in which one person or program successfully masquerades as another by falsifying data and thereby gaining an illegitimate advantage.
- ***Tampering:*** Tampering describes an intentional modification of products in a way that would make them harmful to the consumer.
- ***Repudiation:*** Repudiation describes a situation where the authenticity of a signature is being challenged
- ***Information Disclosure:*** Information Disclosure (Privacy breach or Data leak) describes a situation where information, thought as secure, is released in an untrusted environment.
- ***Elevation of Privilege:*** Elevation of Privilege describes a situation where a person or a program want to gain elevated privileges or access to resources that are normally restricted to him/it.
- ***Exploits:*** An exploit is a piece of software, a chunk of data, or sequence of commands that takes advantage of a software "bug" or "glitch" in order to cause unintended or unanticipated behavior to occur on computer software, hardware, or something electronic (usually computerized). This frequently includes such things as gaining control of a computer system or allowing privilege escalation or a denial of service attack. The term "exploit" generally refers to small programs designed to take advantage of a software flaw that has

been discovered, either remote or local. The code from the exploit program is frequently reused in Trojan horses and computer viruses.

- ***Indirect Attacks:*** An indirect attack is an attack launched by a third-party computer. By using someone else's computer to launch an attack, it becomes far more difficult to track down the actual attacker. There have also been cases where attackers took advantage of public anonymizing systems, such as the tor onion router system.
- ***Computer Crime:*** Computer crime refers to any crime that involves a computer and a network.

7.3 COMPUTER CRIMINALS

In television and film westerns, the bad guys always wore shabby clothes, looked mean and sinister, and lived in gangs somewhere out of town. By contrast, the sheriff dressed well, stood proud and tall, was known and respected by everyone in town, and struck fear in the hearts of most criminals.

To be sure, some computer criminals are mean and sinister types. But many more wear business suits, have university degrees, and appear to be pillars of their communities. Some are high school or university students. Others are middle-aged business executives. Some are mentally deranged, overtly hostile, or extremely committed to a cause, and they attack computers as a symbol. Others are ordinary people tempted by personal profit, revenge, challenge, advancement, or job security. No single profile captures the characteristics of a "typical" computer criminal, and many who fit the profile are not criminals at all.

Whatever their characteristics and motivations, computer criminals have access to enormous amounts of hardware, software, and data; they have the potential to cripple much of effective business and government throughout the world. In a sense, then, the purpose

of computer security is to prevent these criminals from doing damage.

For the purposes of studying computer security, we say computer crime is any crime involving a computer or aided by the use of one. Although this definition is admittedly broad, it allows us to consider ways to protect ourselves, our businesses, and our communities against those who use computers maliciously.

The U.S. Federal Bureau of Investigation regularly reports uniform crime statistics. The data do not separate computer crime from crime of other sorts. Moreover, many companies do not report computer crime at all, perhaps because they fear damage to their reputation, they are ashamed to have allowed their systems to be compromised, or they have agreed not to prosecute if the criminal will "go away." These conditions make it difficult for us to estimate the economic losses we suffer as a result of computer crime; our dollar estimates are really only vague suspicions. Still, the estimates, ranging from $300 million to $500 billion per year, tell us that it is important for us to pay attention to computer crime and to try to prevent it or at least to moderate its effects.

One approach to prevention or moderation is to understand who commits these crimes and why. Many studies have attempted to determine the characteristics of computer criminals. By studying those who have already used computers to commit crimes, we may be able in the future to spot likely criminals and prevent the crimes from occurring. In this section, we examine some of these characteristics.

7.3.1 Amateurs

Amateurs have committed most of the computer crimes reported to date. Most embezzlers are not career criminals but rather are normal people who observe a weakness in a security system that allows them to access cash or other valuables. In the same sense, most computer criminals are ordinary computer professionals or

users who, while doing their jobs, discover they have access to something valuable.

When no one objects, the amateur may start using the computer at work to write letters, maintain soccer league team standings, or do accounting. This apparently innocent time-stealing may expand until the employee is pursuing a business in accounting, stock portfolio management, or desktop publishing on the side, using the employer's computing facilities. Alternatively, amateurs may become disgruntled over some negative work situation (such as a reprimand or denial of promotion) and vow to "get even" with management by wreaking havoc on a computing installation.

7.3.2 Crackers or Malicious Hackers

System crackers often high school or university students, attempt to access computing facilities for which they have not been authorized. Cracking a computer's defenses is seen as the ultimate victimless crime. The perception is that nobody is hurt or even endangered by a little stolen machine time. Crackers enjoy the simple challenge of trying to log in, just to see whether it can be done. Most crackers can do their harm without confronting anybody, not even making a sound. In the absence of explicit warnings not to trespass in a system, crackers infer that access is permitted. An underground network of hackers helps pass along secrets of success; as with a jigsaw puzzle, a few isolated pieces joined together may produce a large effect. Others attack for curiosity, personal gain, or self-satisfaction. And still others enjoy causing chaos, loss, or harm. There is no common profile or motivation for these attackers.

7.3.3 Career Criminals

By contrast, the career computer criminal understands the targets of computer crime. Criminals seldom change fields from arson, murder, or auto theft to computing; more often, criminals begin as

computer professionals who engage in computer crime, finding the prospects and payoff good. There is some evidence that organized crime and international groups are engaging in computer crime. Recently, electronic spies and information brokers have begun to recognize that trading in companies' or individuals' secrets can be lucrative.

Recent attacks have shown that organized crime and professional criminals have discovered just how lucrative computer crime can be. Mike Danseglio, a security project manager with Microsoft, said, "In 2006, the attackers want to pay the rent. They don't want to write a worm that destroys your hardware. They want to assimilate your computers and use them to make money". Mikko Hyppönen, Chief Research Officer with the Finnish security company f-Secure, agrees that today's attacks often come from Russia, Asia, and Brazil and the motive is now profit, not fame. Ken Dunham, Director of the Rapid Response Team for Verisign says he is "convinced that groups of well-organized mobsters have taken control of a global billion-dollar crime network powered by skillful hackers".

Snow observes that a hacker wants a score, bragging rights. Organized crime wants a resource; they want to stay and extract profit from the system over time. These different objectives lead to different approaches: The hacker can use a quick-and-dirty attack, whereas the professional attacker wants a neat, robust, and undetected method.

Some companies are reticent to prosecute computer criminals. In fact, after having discovered a computer crime, the companies are often thankful if the criminal quietly resigns. In other cases, the company is (understandably) more concerned about protecting its assets and so it closes down an attacked system rather than gathering evidence that could lead to identification and conviction of the criminal. The criminal is then free to continue the same illegal pattern with another company.

7.3.4 Terrorists

The link between computers and terrorism is quite evident. We see terrorists using computers in three ways:

- ***Targets of Attack***: denial-of-service attacks and web site defacements are popular for any political organization because they attract attention to the cause and bring undesired negative attention to the target of the attack.
- ***Propaganda Vehicles***: web sites, web logs, and e-mail lists are effective, fast, and inexpensive ways to get a message to many people.
- ***Methods of Attack***: to launch offensive attacks requires use of computers.

We cannot accurately measure the amount of computer-based terrorism because our definitions and measurement tools are rather weak. Still, there is evidence that all three of these activities are increasing.

7.3.5 Top 10 Cyber Crime Prevention Tips

- ***Use Strong Passwords***: Use different user ID / password combinations for different accounts and avoid writing them down. Make the passwords more complicated by combining letters, numbers, special characters (minimum 10 characters in total) and change them on a regular basis.
- ***Secure Your Computer***:
  - *Activate your firewall*: Firewalls are the first line of cyber defense; they block connections to unknown or bogus sites and will keep out some types of viruses and hackers.
  - *Use anti-virus/malware software*: Prevent viruses from infecting your computer by installing and regularly updating anti-virus software.

  - *Block spyware attacks*: Prevent spyware from infiltrating your computer by installing and updating anti-spyware software.
- ***Be Social-Media Savvy***: Make sure your social networking profiles (e.g. Facebook, Twitter, Youtube, MSN, etc.) are set to private. Check your security settings. Be careful what information you post online. Once it is on the Internet, it is there forever!
- ***Secure your Mobile Devices***: Be aware that your mobile device is vulnerable to viruses and hackers. Download applications from trusted sources.
- ***Install the latest operating system updates***: Keep your applications and operating system (e.g. Windows, Mac, Linux) current with the latest system updates. Turn on automatic updates to prevent potential attacks on older software.
- ***Protect your Data***: Use encryption for your most sensitive files such as tax returns or financial records, make regular back-ups of all your important data, and store it in another location.
- ***Secure your wireless network***: Wi-Fi (wireless) networks at home are vulnerable to intrusion if they are not properly secured. Review and modify default settings. Public Wi-Fi, a.k.a. "Hot Spots", are also vulnerable. Avoid conducting financial or corporate transactions on these networks.
- ***Protect your e-identity***: Be cautious when giving out personal information such as your name, address, phone number or financial information on the Internet. Make sure that websites are secure (e.g. when making online purchases) or that you've enabled privacy settings (e.g. when accessing/using social networking sites).
- ***Avoid being scammed***: Always think before you click on a link or file of unknown origin. Don't feel pressured by any emails. Check the source of the message. When in doubt, verify the source. Never reply to emails that ask

you to verify your information or confirm your user ID or password.

- ***Call the right person for help***: Don't panic! If you are a victim, if you encounter illegal Internet content (e.g. child exploitation) or if you suspect a computer crime, identity theft or a commercial scam, report this to your local police. If you need help with maintenance or software installation on your computer, consult with your service provider or a certified computer technician.

7.4 METHODS OF DEFENSE

Computer crime is certain to continue. The goal of computer security is to institute controls that preserve secrecy, integrity, and availability. Sometimes these controls can prevent or mitigate attacks; other, less powerful methods can only inform us that security has been compromised, by detecting a breach as it happens or after it occurs.

Harm occurs when a threat is realized against a vulnerability. To protect against harm, then, we can neutralize the threat, close the vulnerability, or both. The possibility for harm to occur is called **risk**. We can deal with harm in several ways. We can seek to

- *prevent it*, by blocking the attack or closing the vulnerability
- *deter it*, by making the attack harder but not impossible
- *deflect it*, by making another target more attractive (or this one less so)
- *detect it*, either as it happens or sometime after the fact
- *recover* from its effects

Of course, more than one of these can be done at once. So, for example, we might try to prevent intrusions. But in case we do not prevent them all, we might install a detection device to warn of an imminent attack. And we should have in place incident

response procedures to help in the recovery in case an intrusion does succeed.

7.4.1 Controls

To consider the controls or countermeasures that attempt to prevent exploiting a computing system's vulnerabilities, we begin by thinking about traditional ways to enhance physical security. In the Middle Ages, castles and fortresses were built to protect the people and valuable property inside. The fortress might have had one or more security characteristics, including

- a strong gate or door, to repel invaders
- heavy walls to withstand objects thrown or projected against them
- a surrounding moat, to control access
- arrow slits, to let archers shoot at approaching enemies
- crenellations to allow inhabitants to lean out from the roof and pour hot or vile liquids on attackers
- a drawbridge to limit access to authorized people
- gatekeepers to verify that only authorized people and goods could enter

Similarly, today we use a multipronged approach to protect our homes and offices. We may combine strong locks on the doors with a burglar alarm, reinforced windows, and even a nosy neighbor to keep an eye on our valuables. In each case, we select one or more ways to deter an intruder or attacker, and we base our selection not only on the value of what we protect but also on the effort we think an attacker or intruder will expend to get inside.

Computer security has the same characteristics. We have many controls at our disposal. Some are easier than others to use or implement. Some are cheaper than others to use or implement. And some are more difficult than others for intruders to override. Figure 1 illustrates how we use a combination of controls to secure our valuable resources. We use one or more controls, according to

what we are protecting, how the cost of protection compares with the risk of loss, and how hard we think intruders will work to get what they want.

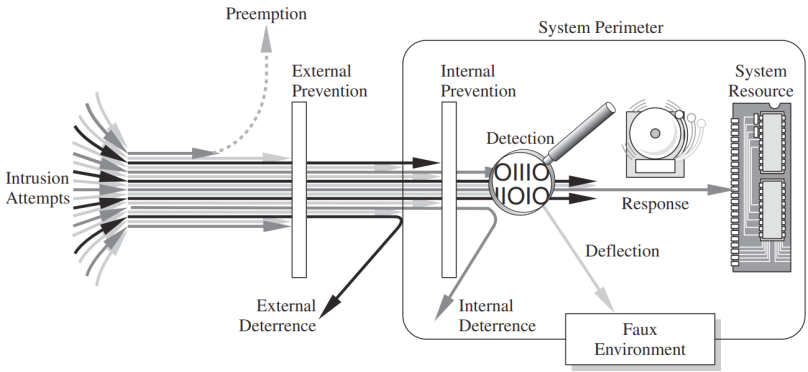


Figure 1. Multiple Controls.

7.4.2 Encryption

Encryption is the formal name for the scrambling process. We take data in their normal, unscrambled state, called cleartext, and transform them so that they are unintelligible to the outside observer; the transformed data are called enciphered text or ciphertext. Using encryption, security professionals can virtually nullify the value of an interception and the possibility of effective modification or fabrication.

Encryption clearly addresses the need for confidentiality of data. Additionally, it can be used to ensure integrity; data that cannot be read generally cannot easily be changed in a meaningful manner. Furthermore, encryption is the basis of protocols that enable us to provide security while accomplishing an important system or network task. A protocol is an agreed-on sequence of actions that leads to a desired result. For example, some operating system protocols ensure availability of resources as different tasks and users request them. Thus, encryption can also be thought of as

supporting availability. That is, encryption is at the heart of methods for ensuring all aspects of computer security.

Although encryption is an important tool in any computer security tool kit, we should not overrate its importance. Encryption does not solve all computer security problems, and other tools must complement its use. Furthermore, if encryption is not used properly, it may have no effect on security or could even degrade the performance of the entire system. Weak encryption can actually be worse than no encryption at all, because it gives users an unwarranted sense of protection. Therefore, we must understand those situations in which encryption is most useful as well as ways to use it effectively.

7.4.3 Software Controls

If encryption is the primary way of protecting valuables, programs themselves are the second facet of computer security. Programs must be secure enough to prevent outside attack. They must also be developed and maintained so that we can be confident of the programs' dependability.

Program controls include the following:

- *internal program controls:* parts of the program that enforce security restrictions, such as access limitations in a database management program
- *operating system and network system controls:* limitations enforced by the operating system or network to protect each user from all other users
- *independent control programs:* application programs, such as password checkers, intrusion detection utilities, or virus scanners, that protect against certain types of vulnerabilities
- *development controls:* quality standards under which a program is designed, coded, tested, and maintained to prevent software faults from becoming exploitable vulnerabilities

We can implement software controls by using tools and techniques such as hardware components, encryption, or information gathering. Software controls frequently affect users directly, such as when the user is interrupted and asked for a password before being given access to a program or data. For this reason, we often think of software controls when we think of how systems have been made secure in the past. Because they influence the way users interact with a computing system, software controls must be carefully designed. Ease of use and potency are often competing goals in the design of a collection of software controls.

7.4.4 Hardware Controls

Numerous hardware devices have been created to assist in providing computer security. These devices include a variety of means, such as

- hardware or smart card implementations of encryption
- locks or cables limiting access or deterring theft
- devices to verify users' identities
- firewalls
- intrusion detection systems
- circuit boards that control access to storage media

7.4.5 Policies and Procedures

Sometimes, we can rely on agreed-on procedures or policies among users rather than enforcing security through hardware or software means. In fact, some of the simplest controls, such as frequent changes of passwords, can be achieved at essentially no cost but with tremendous effect. Training and administration follow immediately after establishment of policies, to reinforce the importance of security *policy* and to ensure their proper use.

We must not forget the value of community standards and expectations when we consider how to enforce security. There are

many acts that most thoughtful people would consider harmful, and we can leverage this commonality of belief in our policies. For this reason, legal and ethical controls are an important part of computer security. However, the law is slow to evolve, and the technology involving computers has emerged relatively suddenly. Although legal protection is necessary and desirable, it may not be as dependable in this area as it would be when applied to more well-understood and long-standing crimes.

Society in general and the computing community in particular have not adopted formal standards of ethical behavior. Some organizations have devised codes of ethics for computer professionals. However, before codes of ethics can become widely accepted and effective, the computing community and the general public must discuss and make clear what kinds of behavior are inappropriate and why.

7.4.6 Physical Controls

Some of the easiest, most effective, and least expensive controls are *physical controls*. Physical controls include locks on doors, guards at entry points, backup copies of important software and data, and physical site planning that reduces the risk of natural disasters. Often the simple physical controls are overlooked while we seek more sophisticated approaches.

7.4.7 Effectiveness of Controls

Merely having controls does no good unless they are used properly. Let us consider several aspects that can enhance the effectiveness of controls.

Awareness of Problem

People using controls must be convinced of the need for security. That is, people will willingly cooperate with security requirements

only if they understand why security is appropriate in a given situation. However, many users are unaware of the need for security, especially in situations in which a group has recently undertaken a computing task that was previously performed with lax or no apparent security.

Likelihood of Use

Of course, no control is effective unless it is used. The lock on a computer room door does no good if people block the door open. Some computer systems are seriously uncontrolled.

- **Principle of Effectiveness:** Controls must be used—and used properly—to be effective. They must be efficient, easy to use, and appropriate.

This principle implies that computer security controls must be efficient enough, in terms of time, memory space, human activity, or other resources used, that using the control does not seriously affect the task being protected. Controls should be selective so that they do not exclude legitimate accesses.

Overlapping Controls

As we have seen with fortress or home security, several different controls may apply to address a single vulnerability. For example, we may choose to implement security for a microcomputer application by using a combination of controls on program access to the data, on physical access to the microcomputer and storage media, and even by file locking to control access to the processing programs.

Periodic Review

Few controls are permanently effective. Just when the security specialist finds a way to secure assets against certain kinds of attacks, the opposition doubles its efforts in an attempt to defeat

the security mechanisms. Thus, judging the effectiveness of a control is an ongoing task.

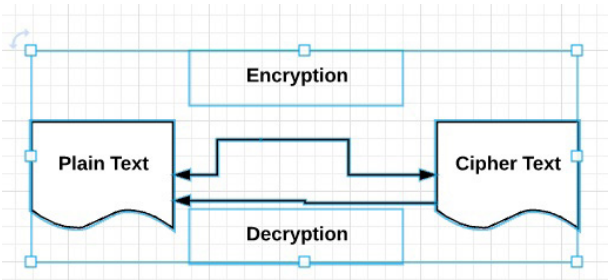
Seldom, if ever, are controls perfectly effective. Controls fail, controls are incomplete, or people circumvent or misuse controls, for example. For that reason, we use overlapping controls, sometimes called a layered defense, in the expectation that one control will compensate for a failure of another. In some cases, controls do nicely complement each other. But two controls are not always better than one and, in some cases, two can even be worse than one. This brings us to another security principle.

- **Principle of Weakest Link:** Security can be no stronger than its weakest link. Whether it is the power supply that powers the firewall or the operating system under the security application or the human who plans, implements, and administers controls, a failure of any control can lead to a security failure.

7.5 ELEMENTARY CRYPTOGRAPHY

Encryption is the process of encoding a message so that its meaning is not obvious; decryption is the reverse process, transforming an encrypted message back into its normal, original form. Alternatively, the terms encode and decode or encipher and decipher are used instead of encrypt and decrypt. That is, we say that we encode, encrypt, or encipher the original message to hide its meaning. Then, we decode, decrypt, or decipher it to reveal the original message. A system for encryption and decryption is called a cryptosystem.

The original form of a message is known as plaintext, and the encrypted form is called cipher text. For convenience, we denote a plaintext message P as a sequence of individual characters P = <p1, p2, …, pn>. Similarly, cipher text is written as C = <c1, c2, …,cm>.



For instance, the plaintext message "I want cookies" can be denoted as the message string <I, ,w,a,n,t,c,o,o,k,i,e,s>. It can be transformed into cipher text<c1, c2, …,c14>, and the encryption algorithm tells us how the transformation is done.

We use this formal notation to describe the transformations between plaintext and cipher text. For example:

we write C = E (P) and P = D(C), where C represents the cipher text, E is the encryption rule, P is the plaintext, and D is the decryption rule.

P = D (E(P)).

In other words, we want to be able to convert the message to protect it from an intruder, but we also want to be able to get the original message back so that the receiver can read it properly.

The cryptosystem involves a set of rules for how to encrypt the plaintext and how to decrypt the cipher text. The encryption and decryption rules, called algorithms, often use a device called a key, denoted by K, so that the resulting cipher text depends on the original plaintext message, the algorithm, and the key value. We write this dependence as C =E (K, P). Essentially, E is a set of encryption algorithms, and the key K selects one specific algorithm from the set.

There are many types of encryption. Two simple forms of encryption: substitutions in which one letter is exchanged for another and transpositions, in which the order of the letters is rearranged.

- ***Cryptanalyst***: cryptanalyst is a person who studies encryption and encrypted message and tries to find the hidden meanings (to break an encryption).
- ***Confusion***: it is a technique for ensuring that ciphertext has no clue about the original message.
- ***Diffusion***: it increases the redundancy of the plaintext by spreading it across rows and columns.

7.5.1 Substitutions Cipher

It basically consists of substituting every plaintext character for a different cipher text character.

It is of two types

1. Mono alphabetic substitution cipher
2. Poly alphabetic substitution cipher

Mono Alphabetic Substitution Cipher

Relationship between cipher text symbol and plain text symbol is 1:1.

- ***Additive cipher***: Key value is added to plain text and numeric value of key ranges from 0 – 25.

Example

Plain text(P)- H E L LO (H=7,E=4,L=11,L=11,O=14)
Key (K)=15
Cipher text (C)= 7+15,4+15,11+15,11+15,14+15
= 22,19, 26,26,(29%26)=3
= W T A AD

- ***Affine cipher***: It is the combination of additive and multiplicative cipher

C= (P+K) mod 26
P= (C-K) mod 26

Let K1 and K2 are two keys

$C = [(P \times K1) + K2] \bmod 26$

$P = [(C-K2) \times K1^{-1}] \bmod 26$

Polyalphabetic Substitution Cipher

In polyalphabetic cipher each occurrence of a character may have different substitution. The relationship between characters in plain text and cipher text is 1 to many.

- Auto key cipher
- Playfair cipher
- Vigegeire cipher
- Hill cipher

Auto Key Cipher

- In this cipher, key is stream of subkeys in which subkey is used to encrypt the corresponding character in the plain text.
- Here 1st subkey is predefined and 2nd subkey is the value of the 1st character of the plain text 3rd subkey is the value of the 2nd plain text and so on.

Example: A T T A C K
0 19 19 0 2 10
Key=12
12 0 19 19 0 2

Cipher text(C)= (12,19,38 19,2 12)%26 → M T M T C M

Playfair Cipher

In playfair cipher the secret key is made of 25 characters arranged in 5x5 matrix

Rules:

- If 2 letters in a plaintext are located in the same row of the secret key then the corresponding encrypted character for each letter is next letter to the right.

- If 2 letters in a pair are in same column then the corresponding encrypted character is next below in the same column.
- If 2 letters are neither in same row or in same column then encrypted character is in its own row but in the same column as the other character.

Example:

$$K = \begin{vmatrix} L & G & D & B & A \\ Q & M & H & E & C \\ U & R & N & I/J & F \\ X & V & S & O & K \\ Z & Y & W & T & P \end{vmatrix}$$

Plain text= HELLO

It is then made as pair.

HE L X L O

H ⟶ E L ⟶ Q L ⟶ B

E ⟶ C X ⟶ Z O ⟶ X

Vigener Cipher

The key stream is the repetition of the initial secret key stream of length m. (1<=m<=26)

Example:

Plaintext- A B C D E F G H
Ks= 0, 5, 8

A	B	C	D	E	F	G	H	(B=1 =>1+5=6=>G)
0	5	8	0	5	8	0	5	
0	**6**	**10**	**3**	**9**	**13**	**6**	**12**	
A	**G**	**K**	**D**	**J**	**N**	**G**	**M**	<= ciphertext

7.5.2 Transposition Cipher

A transposition cipher is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed.

The goal of substitution is confusion; the transposition method is an attempt to make it difficult i.e diffusion.

- Keyless transposition cipher

There are two methods for permutation of characters

- Text is written into a table column by column and transmitted row by row

Example: plaintext- meet me at the park

m e m a t e a k
e t e t h p r

ciphertext- mematteaketethpr

- Text is written into the table row by row and then transmitted column by column.

Example:

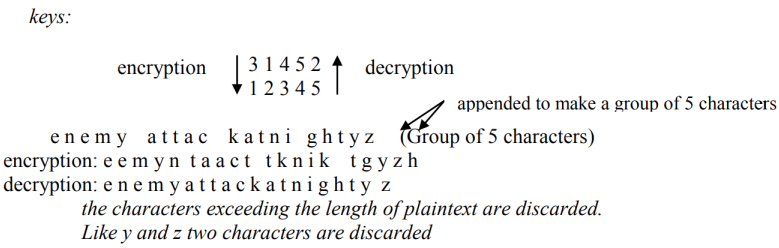
m e e t
m e a t
t h e p
a r k

ciphertext- mmtaeehreaekttp

- Keyed transposition cipher

Plaintext is divided into groups and permutes the character in each group.

Example: plaintext- "enemy attack at night"



- Combining the two approaches:

Encryption and decryption is done in three steps.

- Text is written into a table row by row.
- Permutation is done by reordering the column.
- New table is read column by column

7.6 MAKING "GOOD" ENCRYPTION ALGORITHMS

So far, the encryption algorithms we have seen are trivial, intended primarily to demonstrate the concepts of substitution and permutation. At the same time, we have examined several approaches cryptanalysts use to attack encryption algorithms. Now we examine algorithms that are widely used in the commercial world.

For each type of encryption we considered, has the advantages and disadvantages. But there is a broader question: What does it mean for a cipher to be "good"? The meaning of good depends on the intended use of the cipher. A cipher to be used by military personnel in the field has different requirements from one to be used in a secure installation with substantial computer support. In this section, we look more closely at the different characteristics of ciphers.

7.6.1 Shannon's Characteristics of "Good" Ciphers

In 1949, Claude Shannon proposed several characteristics that identify a good cipher.

- The amount of secrecy needed should determine the amount of labor appropriate for the encryption and decryption.
- The set of keys and the enciphering algorithm should be free from complexity.
- This principle implies that we should restrict neither the choice of keys nor the types of plaintext on which the algorithm can work. For instance, an algorithm that works only on plaintext having an equal number of A's and E's is useless. Similarly, it would be difficult to select keys such that the sum of the values of the letters of the key is a prime number.
- Restrictions such as these make the use of the encipherment prohibitively complex. If the process is too complex, it will not be used. Furthermore, the key must be transmitted, stored, and remembered, so it must be short.
- The implementation of the process should be as simple as possible.
- Principle 3 was formulated with hand implementation in mind: A complicated algorithm is prone to error or likely to be forgotten. With the development and popularity of digital computers, algorithms far too complex for hand implementation became feasible. Still, the issue of complexity is important. People will avoid an encryption algorithm whose implementation process severely hinders message transmission, thereby undermining security. And a complex algorithm is more likely to be programmed incorrectly.
- Errors in ciphering should not propagate and cause corruption of further information in the message.

Principle 4 acknowledges that humans make errors in their use of enciphering algorithms. One error early in the process should not throw off the entire remaining ciphertext. For example, dropping one letter in a columnar transposition throws off the entire remaining encipherment. Unless the receiver can guess where the letter was dropped, the remainder of the message will be unintelligible. By contrast, reading the wrong row or column for a polyalphabetic substitution affects only one character and remaining characters are unaffected.

- The size of the enciphered text should be no larger than the text of the original message.

The idea behind principle 5 is that a ciphertext that expands dramatically in size cannotpossibly carry more information than the plaintext, yet it gives the cryptanalyst more datafrom which to infer a pattern. Furthermore, a longer ciphertext implies more space for storage and more time to communicate.

7.6.2 Properties of "Trustworthy" Encryption Systems

Commercial users have several requirements that must be satisfied when they select an encryption algorithm. Thus, when we say that encryption is "commercial grade," or "trustworthy," we mean that it meets these constraints:

- It is based on sound mathematics. Good cryptographic algorithms are not just invented; they are derived from solid principles.
- It has been analyzed by competent experts and found to be sound. Even the best cryptographic experts can think of only so many possible attacks, and the developers may become too convinced of the strength of their own algorithm. Thus, a review by critical outside experts is essential.
- It has stood the atest of time.a As a new algorithm gains popularity, people continue to review both its mathematical foundations and the way it builds on those foundations. Although a long period of successful use

and analysis is not a guarantee of a good algorithm, the flaws in many algorithms are discovered relatively soon after their release.

We can divide all the cryptography algorithms (ciphers) into two groups: symmetric key cryptography algorithms and asymmetric cryptography algorithms. Figure 2 shows the taxonomy

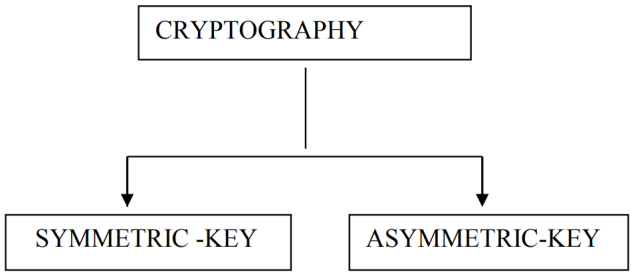


Figure 2. Categories of Cryptography.

7.6.3 Symmetric·Key Cryptography

In symmetric-key cryptography, the same key is used by both parties. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data.

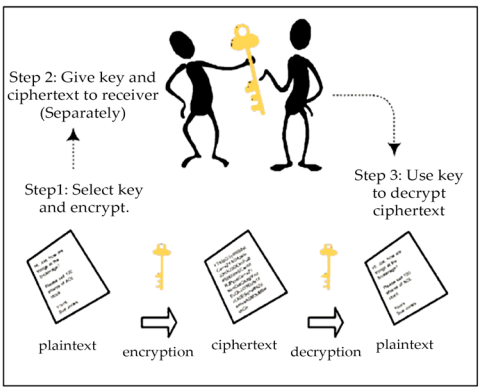


Figure 3. Symmetric-key Cryptography.

7.6.4 Asymmetric-Key Cryptography

In asymmetric or public-key cryptography, there are two keys: a private key and a public key. The private key is kept by the receiver. The public key is announced to the public.

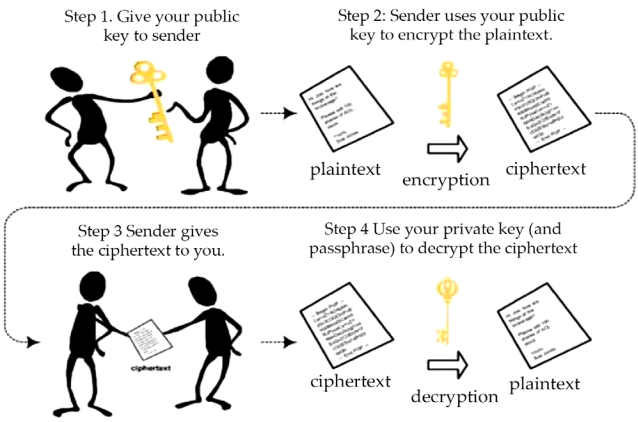


Figure 4. Asymmetric-key Cryptography.

7.7 PRIVACY-PRESERVING REPUTATION MANAGEMENT IN FULLY DECENTRALIZED SYSTEMS: CHALLENGES AND OPPORTUNITIES

Reputation is one of crucial personal information strongly attached to each person, so it affects directly to its owner in whatever way it is used. Any form of reputation violation can carry a serious consequence to its owner. Therefore, reputation needs to be managed in a secure way. Moreover, reputation management is a substantial process as it plays a key role in building up a certain trust level, among just-met users. Thus just-met users can base on reputation scores to make a decision on starting up their communication. In addition, reputation management reduces risks of leaking user privacy and losing data security. So far, several works on managing reputation in online social network have been studied, mostly using the Internet for data transmission.

Thus far, practical systems have mainly evolved around logically centralized models for the management of identity, data, wealth and computing. Centralization requires trust in a service provider, but the continuously reported incidents of surveillance and privacy breaches have shown that this trust can be broken. In addition to that, centralization creates single points of failure that once they become out of service, no one can benefit from them anymore.

Centralization also requires huge investments (e.g., in building data centers), that once established by one party create obstacles of entry to competitors, creating as such a service market dominated by monopoles (e.g., Facebook, Google, etc.). These issues have been calling into question the centralized model for computing, and researchers have reacted by giving a considerable focus on the development of decentralized peer-to-peer (P2P) computing. In P2P computing, services are supposed to be provided by the collaboration of various free peers (i.e., nodes), that put their computing power into the network and contribute in turning the service's protocol. Although such a setting seems to be providing more scalability in terms of both performance and service provision, more privacy, and a more flexible business model within which market entry is more affordable, it does also present tremendous challenges. One of these are related to the security of the P2P network, in terms of the honesty of the participating nodes. One of the main techniques around this problem is reputation management.

Reputation management is the process by which entities in the system are assigned reputation scores that increase or decrease based on their observed behavior. Reputation management is a required in both centralized and decentralized systems, but with mostly different logical needs. In fact, in centralized systems, reputation management is a process carried out by an already trusted service provider, and reputation scores are assigned to end-users based on their centrally observed behavior when using the centrally provided service. That is, the central provider has the full view of usage behaviors of all users, and can impose a corresponding technique to compute user's reputation. The

trusted central provider makes it straightforward for all other users to trust in the corresponding assigned reputation scores to themselves and to the other users.

However, in decentralized system, reputation management becomes a required process not only for end-users but also for the nodes making part of the underlying service provision P2P network. Moreover, in decentralized environments, there is no central trusted authority that observes all behaviors and that is trusted to certify both the mechanism adopted and the reputation scores assigned. These differences, makes it technically challenging to devise a reputation management system in a decentralized setting. In this survey position paper, we focus on the main challenge that the insurance of the security of the decentralized reputation system, in the sense that the assigned scores can be certified, and users or peers cannot tamper with scores, including their own.

7.7.1 Decentralized Protocols for Reputation Management

Reputation score can be obtained by doing evaluation by the surrounding users. For example, trading and sharing systems such as e-commerce, goods selling, hotel booking, ridesharing, etc. where agents do not know each other's, it seems hard for the involver to make the right choice. They, therefore, should rely on the feedback about services or goods they intent to choose. Provider and customer can rate each other after trading or sharing. Peers can learn about the reputation of the others to make a good decision. Reputation allows agents to build trust and to make agents accountable for their behavior.

However, for reputation's importance, attackers pay much attention to find a way abusing the other reputation for their further target. Authors mentioned several rational attacks which can be made when the agents are benefit by deviating from them, including self-promotion, whitewashing, slandering, and denial of service. Hence, they need a tool protect their privacy.

Technically, whenever user privacy is in need, anonymity techniques are considered as the first solution to be used. It can cover the real user ID by a coding layer obfuscating the observance of attackers. Authors built up pseudonym based schemes which require a Trusted Third Party (TTP). They therefore cannot be considered truly decentralized. Additionally, decentralized anonymity-preserving reputation system prevents Sybil attacks by charging a fee. Accredited signers are required to make resource heavy calculations for each rating of each Service Provider. Decentralized anonymous marketplace called Beaver that is resistant against Sybil attacks on vendor reputation, while preserving user anonymity. By employing some cryptographic primitives such as non-interactive zero-knowledge proofs and linkable ring signatures, Beaver attain high levels of usability and practicality, and strong anonymity for buyer. However, it only offers limited protection for vendors and reveals many other weaknesses such as the problems of unclaimed reviewing fees, collusion attacks, advanced searching, etc. Decentralized anonymity-preserving reputation system. In this system, the feedback will be sent by the client after the transaction a changable timeout. Merkle trees and signed blocks of data are used to minimize the workload on the trackers and to fairly distribute the record maintenance tasks to the service providers. It is proved to be efficient, anonymity-preserving, decentralized, and robust against various known attacks against reputation systems, such as ballot-stuffing and Sybil attacks.

Indeed, anonymity techniques have first been designed for static collections of data; however, with the emergence of situations and scenarios requiring the anonymization of dynamic and real time data (such as online data-streams, geolocalization information, etc.), these techniques have been revised to fit such scenarios. Therefore, a desired protocol should be satisfied the criteria of anonymity, low overhead, and proper management of new agents.

There are also some approaches on data hiding or cryptography. Pavlov et al. proposed one of the first privacy preserving reputation systems. They protect confidentiality of feedback

by hiding the secure sum and verifiable secret sharing of the submitted ratings. In the meanwhile, approaching another methods based on cryptographic algorithms, Dolev et al. can only make a conditionally secure system using homomorphic and commutative cryptography. However, it cannot be resistant against large numbers of colluding malicious adversaries. Some systems based on additive homomorphic cryptography and Zero-Knowledge proofs where privacy of a given user can be preserved even in the presence of a large majority of malicious users. However, these protocols cannot hide the list of users who participated in the rating.

Moreover, we can find some other techniques can be used for reputation management. Bethencourt et al.' protocol used signatures on published data. However, attacker may take advantage of his old good reputation without being affected by any new dissatisfaction that his recent activity because the feedback is monotonic.

For applications, another well-known protocol for this problem is EigenTrust which meet most of the requirements. This co-utile protocol can be applied to a variety of scenarios and heterogeneous reputation needs. Some extended version of it tried to provide a more secure distributed calculation that cancels benefits of deviating from the protocol. Sanchez et al. focused on privacy concerns and the lack of trust of ridesharing. They proposed a fully decentralized P2P ridesharing management network that brings trust among peers. This is based on the co-utility idea, like invisible hand theory, which makes rational (even selfish) user collaborate and follows the proposed protocols in a self-enforcing way because of their own benefit. This benefit also directs them out of attack these system.

7.7.2 Challenges

Mobile user's reputation in the decentralized environment gets more delicate and easier to be defrauded and violated, since

there is no any central system managing users as well as doing authentication and authorization, like in the centralized network. The situation gets harsher when a peer enters a new geographical area, and interacts with strangers nearby. Trusting wrong user can give the malicious user a chance of misusing user's reputation. This then carries a serious consequence to the reputation owner as the reputation affects directly to the others' trust in them. In addition, restrictions of decentralized mobile environments, such as weak power, weak computing performance, lack of physical resources, etc. also contribute to make the environment. Moreover, reputation computation has its own higher security constraints. Furthermore there are still the other edges essentially taken into consideration as trustworthiness estimation, identity validation, owner authentication, as well as privacy preservation problems. In this section, each of challenges is discussed as follows.

Identity Validation

Identity validation is one of major concerns relating to reputation management, particularly in online social network (OSN), since an accurate identity validation can support the process of reputation use or user's credentiality evaluation to be performed in a more trustworthy manner. Well validating identity intensifies the security of reputation computing too. Based on such benefits for reputation management, identity has been a very interesting problem taking much effort of researchers. Some recent works investigated to validate user identity in OSN. In some other works, authors focused on another side of identity validation as detecting the fake user's profile on OSN, while some others studied how to enhance the environment safety for OSN users.

The authors proposed an identity validating model, by learning the correlations between attributes of users' profiles so the process can achieve the identity perspective. The model also manipulate the profile's trustworthiness validation. For example, attributes (Job, Education) could be used for inferring a trustworthy profile on OSN. The authors exploited a supervised and feedback-based approach by which a group of trusted users provided feedback

on the correlations between attribute values. Then the learned correlations are used for evaluating the trustworthiness of new target profiles. The approach can be considered as a solution for quantifying trustworthiness of an OSN user's profile and aiming to give a proof for the reputation evaluation process. However, when this solution is applied to the mobile decentralized environment, still many points need to be revised, such as autonomous trustworthiness quantification and authentication.

Privacy Challenges in Autonomous Identity Validation

In decentralization model, identity validation is not performed by any central node. It should be done independently at each peer in the network. The workload from the server now is divided to peers. Peers need to authenticate, authorize, data synchronization, and data coordination between them with their communication partners. The interaction only between peers can carry various risks as one peer itself cannot learn the necessary information about its partners in the network. Therefore, the identity validation in the decentralization model make more challenges to peers than the centralized model. Any information revealed not to the trustworthy users can be abused for further hazardous goals. Privacy can be breached as a result of personal data inference from provided information, or leaked while personal information is transmitted over the underlying network.

Although anonymization techniques have largely used for privacy preservation across a diversity of scenarios and application domains, they do not ensure complete guarantees on privacy. Moreover, these techniques have been proved to be vulnerable to some analysis attacks especially when attackers can access to some additional information that is not then saved into the targeted dataset. In the other hand, these reputation data often include the users' private data such as their name, location, habits, etc. Although there are some syntactic anonymization techniques such as k-anonymity, l-diversity, t-closeness, β-likeness, etc. for identity disclosure and personal information inference, this is still a problem where a centralized reputation management can enable

the threat of compilation, aggregation, exploitation on these data, or even provide the bias matches. This central point of failure can also become a target for external attacks.

Newer research on privacy suggested the concept of differential privacy. Actually, differential privacy can be achieved by adding noise to database to obfuscate the real content. While some researchers believe that differential privacy is the answer to the new requirements on data privacy, others still find it immature to completely replace the well-established syntactic alternatives. We believe that the two approaches are equally important in the sense that each of them answers or fits different requirements and different scenarios.

7.7.3 Opportunities and Possible Research Paths

As presented in previous section, the mentioned challenges open further requirements for researchers on the reputation management. In this section, we discuss about abilities of reputation management autonomy and of reputation privacy preservation, as follows:

- ***Autonomous reputation management***. Each user in the peer-to-peer network needs to manage their own reputation score in an autonomous way. It means that the involved users have to store their reputation score locally, and have to validate the other partners' reputation by themselves without the support of any the third party or server, before deciding to make any trade with the others. Hence, there is a need of an effective protocol for users in the decentralized systems to store and use the reputation scores. Additionally, there is also a need of evaluating correctly a user based on its overhanded reputation score. Moreover, rating the partner's reputation after performing any trade is requested to be honest and correct as well.
- ***Reputation Privacy Preservation***. So as to avoid an incorrect identity validation which may cascade the user

reputation degradation, anonymization and pseudonym techniques are first thought about as effective solutions for this problem. However, they still have drawbacks which may cause another analysis attacks. Hence, to improve the reputation management, cryptographic algorithms, e.g., hash function, homomorphic encryption, etc., are considered as the powerful means to strengthen the ability of protecting user personal information against the stealing behaviors of malicious insiders or outsiders, and expected to be strongly against any malicious behavior. Anonymity techniques and cryptographic algorithms are always considered as two competitive solutions, as anonymity costs less time and space than cryptographic algorithms. While, encryption takes more time and memory of the system. However, it is not the truth like that. Combinations of the two can carry the wonderful result in protecting the privacy against adversaries. However, up to now there have not existed any effective solution for all scenarios, particularly, most of systems have very strict constraints in performance and time.

REFERENCES

1. Anceaume, E., Guette, G., Lajoie Mazenc, P., Prigent, N., Tong, V.V.T.: A Privacy Preserving Distributed Reputation Mechanism (2012). https://hal.archivesouvertes. fr/hal-00763212
2. Anderson, J. M. (2003). "Why we need a new definition of information security". Computers & Security. 22 (4): 308–313. doi:10.1016/S0167-4048(03)00407-3.
3. Androulaki, E., Choi, S.G., Bellovin, S.M., Malkin, T.: Reputation systems for anonymous networks. In: Proceedings of the 8th International Symposium on Privacy Enhancing Technologies, PETS. Springer, Heidelberg (2008)
4. Bahri, L., Carminati, B., Ferrari, E.: Community-based identity validation on online social networks. In: IEEE ICDCS, pp. 21–30 (2014)
5. Bazin, R., et al.: A decentralized anonymity-preserving reputation system with constant-time score retrieval. IACR, p. 146 (2016)
6. Cao, J., Carminati, B., Ferrari, E., Tan, K.L.: CASTLE: a δ-constrained scheme for k_s-anonymizing data streams. In: IEEE 24th International Conference on Data Engineering (ICDE 2008), pp. 1376–1378 (2008)
7. Cao, J., Karras, P.: Publishing microdata with a robust privacy guarantee. Proc. VLDB Endowment 5(11), 1388–1399 (2012)
8. Cao, Q., Sirivianos, M., Yang, X., Pregueiro, T.: Aiding the detection of fake accounts in large scale social online services. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, p. 15 (2012)
9. Chairunnanda, P., Pham, N., Hengartner, U.: Privacy: gone with the typing! identifying web users by their typing patterns. In: IEEE PASSAT (2011)
10. Cherdantseva Y. and Hilton J.: "Information Security and Information Assurance. The Discussion about the Meaning, Scope and Goals". In: Organizational, Legal, and Technological

Dimensions of Information System Administrator. Almeida F., Portela, I. (eds.). IGI Global Publishing. (2013)

11. Clifton, C., Tassa, T.: On syntactic anonymity and differential privacy. In: ICDE Workshops, pp. 88–93 (2013)
12. Dhillon, Gurpreet (2007). Principles of Information Systems Security: text and cases. NY: John Wiley & Sons. ISBN 978-0-471-45056-6.
13. Dimitriou, T., Michalas, A.: Multi-party trust computation in decentralized environments in the presence of malicious adversaries. Ad-Hoc Netw. 15, 53–66 (2014)
14. Dolev, S., Gilboa, N., Kopeetsky, M.: Efficient private multiparty computations of trust in the presence of curious and malicious users. J. Trust Manage. 1, 1–8 (2014)
15. Domingo-Ferrer, J., Farràs, O., Martínez, S., Sánchez, D., Soria-Comas, J.: Selfenforcing protocols via co-utile reputation management. Inf. Sci. 367(C), 159–175 (2016). https://doi.org/10.1016/j.ins.2016.05.050
16. Domingo-Ferrer, J., Sanchez, D., Soria-Comas, J.: Co-utility: self-enforcing collaborative protocols with mutual help. Prog. Artif. Intell. 5(2), 105–110 (2016)
17. Domingo-Ferrer, J., Soria-Comas, J., Ciobotaru, O.: Co-utility: self-enforcing protocols without coordination mechanisms. In: Proceedings of the 5th International Conference on Industrial Engineering and Operations Management-IEOM, pp. 1–7 (2015)
18. Furuhata, F., Dessouky, M., Ordoez, F., Brunet, M.E., Wang, X., Koening, S.: Ridesharing: the state-of-the-art and future directions. Transp. Res. Part B 57, 28–46 (2013)
19. Goga, O., Lei, H., Parthasarathi, S.H.K., Friedland, G., Sommer, R., Teixeira, R.: Exploiting innocuous activity for correlating users across sites. In: International World Wide Web Conferences Steering Committee, WWW (2013)
20. Guo, K., Zhang, Q.: Fast clustering-based anonymization approaches with time constraints for data streams. Knowl. Based Syst. 46, 95–108 (2013)

21. Hasan, O., Brunie, L., Bertino, E., Shang, N.: A decentralized privacy preserving reputation protocol for the malicious adversarial model. IEEE Trans. Inf. Forensics Secur. 8(6), 949–962 (2013)
22. Hasan, O., Brunie, L., Bertino, E.: Preserving privacy of feedback providers in decentralized reputation systems. Comput. Secur. 31(7), 816–826 (2012)
23. He, B., Chen, C., Su, Y., Sun, H.: A defence scheme against Identity Theft Attack based on multiple social networks. Expert Systems with Application. Elsevier (2014)
24. Hoffman, K., Zage, D., Nita-Rotaru, C.: A survey of attack and defense techniques for reputation systems. ACM Comput. 42(1), 1 (2009)
25. Jin, L., Takabi, H., Joshi, J.B.D.: Towards active detection of identity clone attacks on online social networks. In: ACM CODASPY (2011)
26. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th International Conference on World Wide Web, pp. 640–651 (2003)
27. Krutz, Ronald L.; Russell Dean Vines (2003). The CISSP Prep Guide (Gold ed.). Indianapolis, IN: Wiley. ISBN 978-0-471-26802-4.
28. Lajoie-Mazenc, P., Anceaume, E., Guette, G., Sirvent, T., Tong, V.V.T.: Efficient distributed privacy-preserving reputation mechanism handling non-monotonic ratings (2015). https://hal.archives-ouvertes.fr/hal-01104837
29. Layton, Timothy P. (2007). Information Security: Design, Implementation, Measurement, and Compliance. Boca Raton, FL: Auerbach publications. ISBN 978-0-8493-7087-8.
30. Li, H.P., Hu, H., Xu, J.: Nearby friend alert: location anonymity in mobile geosocial networks. IEEE Pervasive Comput. 12(4), 62–70 (2013)
31. Li, N., Li, T., Venkatasubramanian, S.: t-Closeness: privacy beyond k-Anonymity and l-Diversity. ICDE 7, 106–115 (2007)

32. llen, Julia H. (2001). The CERT Guide to System and Network Security Practices. Boston, MA: Addison-Wesley. ISBN 978-0-201-73723-3.
33. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data (TKDD), 1, 1–3 (2007)
34. Mascetti, S., Freni, D., Bettini, C., Wang, X.S., Jajodia, S.: Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. VLDB J. Int. J. Very Large Data Bases 20(4), 541–566 (2011)
35. McNab, Chris (2004). Network Security Assessment. Sebastopol, CA: O'Reilly. ISBN 978-0-596-00611-2.
36. O'Hagan, A.: No Place to Hide: Edward Snowden, the NSA and the Surveillance State by Glenn Greenwald. London Review of Books. Nicholas Spice. vol. 36, no. 18, pp. 11–12 (2014)
37. Pavlov, E., Rosenschein, J.S., Topol, Z.: Supporting privacy in decentralized additive reputation systems. In: Trust Management, Lecture Notes in Computer Science, vol. 2995, pp. 108–119. Springer, Heidelberg (2004)
38. Peltier, Thomas R. (2001). Information Security Risk Analysis. Boca Raton, FL: Auerbach publications. ISBN 978-0-8493-0880-2.
39. Peltier, Thomas R. (2002). Information Security Policies, Procedures, and Standards: guidelines for effective information security management. Boca Raton, FL: Auerbach publications. ISBN 978-0-8493-1137-6.
40. Petrlic, R., Lutters, S., Sorge, C.: Privacy-preserving reputation management. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, New York, USA, pp. 1712–1718 (2014)
41. Pipkin, D. (2000). Information security: Protecting the global enterprise. New York: Hewlett-Packard Company.
42. Raval, S.: Decentralized Applications. O'Reilly Media (2016)

43. Roffo, G., Segalin, C., Vinciarelli, A., Murino, V., Cristani, M.: Reading between the turns: statistical modeling for identity recognition and verification in chats. IEEE AVSS, pp. 99–104 (2013)
44. Samonas, S.; Coss, D. (2014). "The CIA Strikes Back: Redefining Confidentiality, Integrity and Availability in Security". Journal of Information System Security. 10 (3): 21–45. Archived from the original on 2018-09-22. Retrieved 2018-01-25.
45. Sanchez, D., et al.: Co-utile P2P ridesharing via decentralization and reputation management. Transp. Res. Part C Emerg. Technol. 73, 147–166 (2016)
46. Schaub, A., Bazin, R., Hasan, O., Brunie, L.: A trustless privacy-preserving reputation system. IFIP SEC - Privacy (2016)
47. Sirivianos, M., Kim, K., Gan, J.W., Yang, X.: Assessing the veracity of identity assertions via osns. IEEE COMSNETS (2012)
48. Soska, K., et al.: Beaver: a decentralized anonymous marketplace with secure reputation. IACR Cryptography ePrint Archive, 464 (2016)
49. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertainty Fuzziness Knowl. Based Syst. 10(5), 557–570 (2002)
50. Venter, H. S.; Eloff, J. H. P. (2003). "A taxonomy for information security technologies". Computers & Security. 22 (4): 299–307. doi:10.1016/S0167-4048(03)00406-1.
51. White, Gregory (2003). All-in-one Security+ Certification Exam Guide. Emeryville, CA: McGraw-Hill/Osborne. ISBN 978-0-07-222633-1.
52. Yu, H., Gibbons, P.B., Kaminsky, M., Xiao, F.: Sybillimit: Anear-optimal social network defense against sybil attacks. IEEE Security and Privacy (2008)

INDEX

Intelligent Informatics

Data is all around us. The Internet of Things (IoT) and sensors have the ability to harness large volumes of data, while artificial intelligence (AI) can learn patterns in the data to automate tasks for a variety of business benefits. Artificial Intelligence enhances the speed, precision and effectiveness of human efforts. In financial institutions, AI techniques can be used to identify which transactions are likely to be fraudulent, adopt fast and accurate credit scoring, as well as automate manually intense data management tasks. Artificial intelligence is going to change every industry, but we have to understand its limits. The principle limitation of AI is that it learns from the data. There is no other way in which knowledge can be incorporated. That means any inaccuracies in the data will be reflected in the results. And any additional layers of prediction or analysis have to be added separately. Artificial intelligence (AI) has become a technological reality for businesses and organizations across industries. Even if its benefits may not be always easy to quantify, AI has proven itself capable of improving process efficiency, reducing human errors and labor, and extracting insights from big data.
Intelligence and security informatics (ISI) is an emerging field of study aimed at developing advanced information technologies, systems, algorithms, and databases for national - and homeland-security-related applications, through an integrated technological, organizational, and policy-based approach. This book summarizes the broad application and policy context for this emerging field. On a smaller scale, AI has made it into many people's lives in one form or another, even if its presence is not always recognized. To better understand the state of AI, let's take a closer look at the role AI plays in business and in many people's lives even as we speak. It will cover fundamental principles and algorithms of intelligent informatics. It also deals with understand the data collection, data integration, data cleansing, data representation, model construction, model selection, model averaging, model evaluation, and model interpretation components in intelligent informatics.

Jiquan Chen holds PhD in in Information and Communication Engineering. He is a Professor and the Head of the Department of Information Technology. His research interesets are biometrics, biomedical signals and images, data mining, classification systems, signal and image processing, machine learning, and environmental intelligence.

ISBN 978-1-98467-790-7

BIBLIOTEX
Digital Library