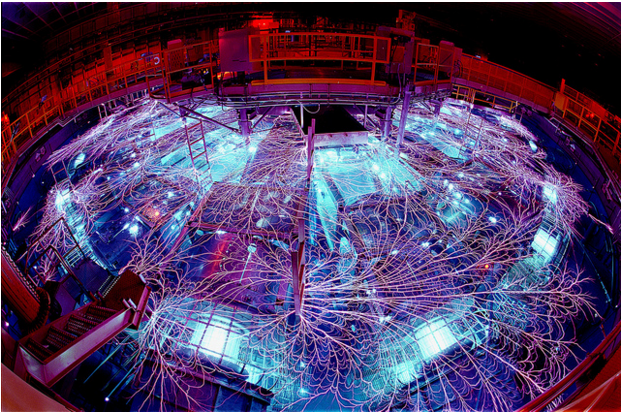
****­­

**Machine Learning and Neural Networks**

**And it’s Applications**

Made by: Ali Ibrahim

Supervised by: MR. Amjad Taha

Class: 11

# Introduction:

In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed"

Tom M. Mitchell provided a widely quoted, more formal definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". This definition is notable for its defining machine learning in fundamentally operational rather than cognitive terms, thus following Alan Turing's proposal in his paper "Computing Machinery and Intelligence" that the question "Can machines think?" And if they can how?

And what is a neural network? And what its application?

# Chapter (1):

## What is Machine Learning?

**Machine learning** is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data.[[1]](#footnote-1) Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions,[[2]](#footnote-2) rather than following strictly static program instructions.

Machine learning is closely related to computational statistics; a discipline that aims at the design of algorithm for implementing statistical methods on computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is infeasible. Example applications include spam filtering, optical character recognition (OCR),[[3]](#footnote-3) search engines and computer vision. Machine learning is sometimes conflated with data mining,[[4]](#footnote-4) although that focuses more on exploratory data analysis.[[5]](#footnote-5) Machine learning and pattern recognition "can be viewed as two facets of the same field."

It should be noted, however, that even when one has an apparently massive data set, the effective number of data points for certain cases of interest might be quite small. In fact, data across a variety of domains exhibits a property known as the **long tail**, which means that a few things (e.g., words) are very common, but most things are quite rare. For example, 20% of Google searches each day have never been seen before.

## Types of Learning:

There are three main types for learning:

### First (Supervised Learning):

**Supervised learning** is the machine learning task of inferring a function from labeled training data.[[6]](#footnote-6) The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.
2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.
5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation* set) of the training set, or via cross-validation.
6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

Algorithm:

In the **predictive** or **supervised learning** approach, the goal is to learn a mapping from inputs x to outputs y, given a labeledset of input-output pairs D = {(xi, yi)} Ni=1. Here D is called the **training set**, and N is the number of training examples.

In the simplest setting, each training input xi is a D-dimensional vector of numbers, representing, say, the height and weight of a person. These are called **features**, **attributes** or **covariates**. In general, however, xi could be a complex structured object, such as an image, a sentence, an email message, a time series, a molecular shape, a graph, etc.

Similarly the form of the output or **response variable** can in principle be anything, but most methods assume that yi is a **categorical** or **nominal** variable from some finite set, yi ∈ {1, . . . ,C} (such as male or female), or that yi is a real-valued scalar (such as income level). When yi is categorical, the problem is known as **classification** or **pattern recognition**, and when yi is real-valued, the problem is known as **regression**. Another variant, known as **ordinal regression**, occurs where label space Y has some natural ordering, such as grades A–F.

In supervised learning, the aim is to learn a mapping from the input to an output whose correct values are provided by a supervisor. In unsupervised learning, there is no such supervisor and we only have input data.

The aim is to find the regularities in the input. There is a structure to the input space such that certain patterns occur more often than others, and we want to see what generally happens and what does not. In statistics, this is called *density estimation*.[[7]](#footnote-7)

### Second (Unsupervised Learning):

We now consider **unsupervised learning**, where we are just given output data, without any inputs. The goal is to discover “interesting structure” in the data; this is sometimes called **knowledge discovery**. Unlike supervised learning, we are not told what the desired output is for each input. Instead, we will formalize our task as one of **density estimation**, that is, we want to build models of the form p (xi|θ). There are two differences from the supervised case.

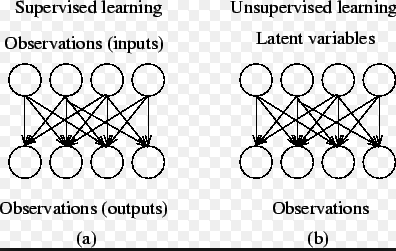
First, we have written p (xi|θ) instead of p (yi|xi, θ); that is, supervised learning is conditional density estimation, whereas unsupervised learning is unconditional density estimation. Second, xi is a vector of features, so we need to create multivariate probability models. By contrast, in supervised learning, yi is usually just a single variable that we are trying to predict. This means that for most supervised learning problems, we can use univariate probability models (with input-dependent parameters), which significantly simplifies the problem.

Unsupervised learning is arguably more typical of human and animal learning. It is also more widely applicable than supervised learning, since it does not require a human expert to manually label the data. Labeled data is not only expensive to acquire6, but it also contains relatively little information, certainly not enough to reliably estimate the parameters of complex models. Geo Hinton, who is a famous professor of ML at the University of Toronto, has said:

When we’re learning to see, nobody’s telling us what the right answers are — we just look. Every so often, your mother says “that’s a dog”, but that’s very little information.

You’d be lucky if you got a few bits of information — even one bit per second — that way. The brain’s visual system has 1014 neural connections. And you only live for 109 seconds. So it’s no use learning one bit per second. You need more like 105 bits per second. And there is only one place you can get that much information: from the input itself.

Here we are only given inputs, D = {xi} Ni=1, and the goal is to find “interesting patterns” in the data. This is sometimes called **knowledge discovery**. This is a much less well-defined problem, since we are not told what kinds of patterns to look for, and there is no obvious error metric to use (unlike supervised learning, where we can compare our prediction of y for a given x to the observed value).[[8]](#footnote-8)

(1)

### Third (Reinforcement learning):

Reinforcement learning is learning what to do--how to map situations to tell which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics--trial-and-error search and delayed reward--are the two most important distinguishing features of reinforcement learning.

Reinforcement learning is different from *supervised learning*, the kind of learning studied in most current research in machine learning, statistical pattern recognition, and artificial neural networks. Supervised learning is learning from examples provided by a knowledgeable external supervisor. This is an important kind of learning, but alone it is not adequate for learning from interaction. In interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. In uncharted territory-- where one would expect learning to be most beneficial--an agent must be able to learn from its own experience.

One of the challenges that arise in reinforcement learning and not in other kinds of learning is the trade-off between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward.

But to discover such actions, it has to try actions that it has not selected before. The agent has to *exploit* what it already knows in order to obtain reward, but it also has to *explore* in order to make better action selections in the future. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task. The agent must try a variety of actions *and* progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate its expected reward. The exploration-exploitation dilemma has been intensively studied by mathematicians for many decades. For now, we simply note that the entire issue of balancing exploration and exploitation does not even arise in supervised learning as it is usually defined.

One of the larger trends of which reinforcement learning is a part is that toward greater contact between artificial intelligence and other engineering disciplines. Not all that long ago, artificial intelligence was viewed as almost entirely separate from control theory and statistics. It had to do with logic and symbols, not numbers. Artificial intelligence was large LISP programs, not linear algebra, differential equations, or statistics. Over the last decades this view has gradually eroded.

Modern artificial intelligence researchers accept statistical and control algorithms, for example, as relevant competing methods or simply as tools of their trade. The previously ignored areas lying between artificial intelligence and conventional engineering are now among the most active, including new fields such as neural networks, intelligent control, and our topic, reinforcement learning. In reinforcement learning we extend ideas from optimal control theory and stochastic approximation to address the broader and more ambitious goals of artificial intelligence.

Beyond the agent and the environment, one can identify four main sub elements of a reinforcement learning system: a *policy*, a *reward function*, a *value function*, and, optionally, a *model* of the environment.

A *policy* defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. It corresponds to what in psychology would be called a set of stimulus-response rules or associations.

In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior. In general, policies may be stochastic.

A *reward function* defines the goal in a reinforcement learning problem.

Whereas a reward function indicates what is good in an immediate sense, a *value function* specifies what is good in the long run.

The fourth and final element of some reinforcement learning systems is a *model* of the environment.

This is something that mimics the behavior of the environment. For example, given a state and action, the model might predict the resultant next state and next reward. Models are used for *planning*, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced. The incorporation of models and planning into reinforcement learning systems is a relatively new development. Early reinforcement learning systems were explicitly trial-and-error learners; what they did was viewed as almost the *opposite* of planning.

Nevertheless, it gradually became clear that reinforcement learning methods are closely related to dynamic programming methods, which do use models, and that they in turn are closely related to state-space planning methods.[[9]](#footnote-9)

# Chapter (2):

## What Are Neural Networks?

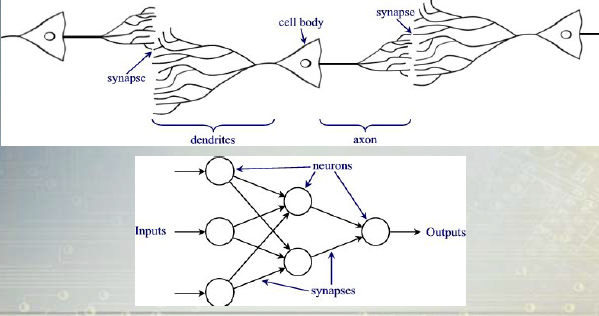
A biological neuron consists of dendrite, a cell body, and an axon. The connections between the dendrite and the axons of other neurons are called synapses. Electric pulses coming from other.

Neurons are translated into chemical information at each synapse. The cell body inputs these pieces of information and fires an electric pulse if the sum of the inputs exceeds a certain threshold. The network consisting of these neurons is a NN, the most essential part of our brain activity.

The main function of the biological neuron is to output pulses according to the sum of multiple signals from other neurons with the characteristics of a pseudo-step function. The second function of the neuron is to change the transmission rate at the synapses to optimize the whole network. An artificial neuron model simulates multiple inputs and one output, the switching function of input–output relation, and the adaptive synaptic

The first neuron model proposed in 1943 used a step function for the switching function [(McCulloch&Pitts, 1943)]. However, the perceptron [(Rosenblatt, 1958)] that is a NN consisting of this type of neuron has limited capability, because of the constraints of binary on/off signals. Today, several continuous functions, such as sigmoidal or radial functions, are used as a neuron characteristic functions, which results higher performance of NNs.

Several learning algorithms that change the synaptic weights have been proposed. The combination of the artificial NNs and the learning algorithms have been applied to several engineering purposes.

(2)

## Components of neural networks:

A technical neural network consists of simple processing units, the neurons, and directed, weighted connections between those neurons. Here, the strength of a connection (or the connecting weight) between two neurons i and j is refereed to us wi, j. A neural network is a sorted triple (N, V, w) with two sets N, V and a function w, where N is the set of neurons and V a set C:\Users\Ali\Desktop\Capture.PNG whose elements are called connections between neuron i and neuron j. The function w: V 🡪 R defines the weights, where w ((i,j)) the weight of the connection between neuron i and neuron j is shortened wi,j. Depending on the point of view it is either undefined or 0 for connections that don’t exist in the network.

So the weight can be implemented in a square weight matrix W or, optionally, in a weight vector W with the row number of the matrix indicating where the connection begins, and the column number of the matrix indicating, which neuron is the target. Indeed, in this case numeric 0 marks a non-existing connection. This matrix representation is also called Hinton diagram.[[10]](#footnote-10)

How Do Neural Networks Work?  
●The output of a neuron is a function of the weighted sum of the inputs plus a bias.

●The function of the entire neural network is simply the computation of the outputs of all the neurons.  
►An entirely deterministic calculation.

●Applied to the weighted sum of the inputs of a neuron to produce the output.  
●Majority of NN’s use sigmoid functions  
►Smooth, continuous, and monotonically increasing (derivative is always positive).  
►Bounded range - but never reaches max or min.   
■Consider “ON” to be slightly less than the max and “OFF” to  
be slightly greater than the min.

●The most common sigmoid function used is the logistic function:   
►f(x) = 1 / (1 + e-x).  
►The calculation of derivatives are important for neural networks and the logistic function has a very nice derivative.  
■ F’(x) = f(x) (1 - f(x)).  
●Other sigmoid functions also used.  
►Hyperbolic tangent.  
►Arctangent.  
●The exact nature of the function has little effect on the abilities of the neural network.

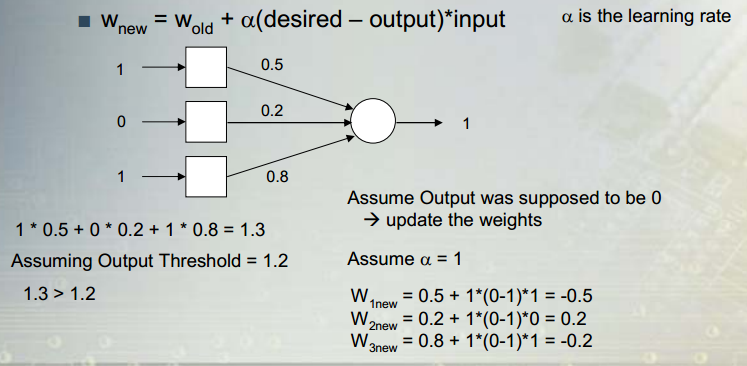
**Where Do The Weights Come From?**  
●The weights in a neural network are the most important factor in determining its function.  
●Training is the act of presenting the network with some sample data and modifying the weights to better approximate the desired function.  
●There are two main types of training:

►Supervised Training.

►Unsupervised Training.

**Perceptron:**   
●First neural network with the ability to learn.  
●Made up of only input neurons and output neurons.  
●Input neurons typically have two states: ON and OFF.  
●Output neurons use a simple threshold activation function.  
●In basic form, can only solve linear problems.  
►Limited applications.

**How Do Perceptrons Learn?**  
●Uses supervised training.  
●If the output is not correct, the weights are.[[11]](#footnote-11)  
Adjusted according to the formula:

****

## Benefits:

**1.** *Nonlinearity.* An artificial neuron can be linear or nonlinear. A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear. Moreover, the nonlinearity is of a special kind in the sense that it is *distributed* throughout the network.

Nonlinearity is a highly important property, particularly if the underlying physical mechanism responsible for generation of the input signal (e.g., speech signal) is inherently nonlinear.

**2.** *Input–Output Mapping.* A popular paradigm of learning, called *learning with a teacher*, or *supervised learning*, involves modification of the synaptic weights of a neuralnetwork by applying a set of labeled *training examples*, or *task examples.* Each exampleconsists of a unique *input signal* and a corresponding *desired (target) response.* The networkis presented with an example picked at random from the set, and the synaptic weights(free parameters) of the network are modified to minimize the difference between thedesired response and the actual response of the network produced by the input signal inaccordance with an appropriate statistical criterion. The training of the network is repeatedfor many examples in the set, until the network reaches a steady state where thereare no further significant changes in the synaptic weights. The previously applied trainingexamples may be reapplied during the training session, but in a different order. Thusthe network learns from the examples by constructing an *input–output mapping* for theproblem at hand. Such an approach brings to mind the study of *nonparametric statistical inference*, which is a branch of statistics dealing with model-free estimation, or, from a biologicalviewpoint, *tabula rasa* learning (Geman et al., 1992); the term “nonparametric”is used here to signify the fact that no prior assumptions are made on a statistical modelfor the input data. Consider, for example, a *pattern classification* task, where the requirementis to assign an input signal representing a physical object or event to one ofseveral prespecified categories (classes). In a nonparametric approach to this problem,the requirement is to “estimate” arbitrary decision boundaries in the input signal spacefor the pattern-classification task using a set of examples, and to do so *without* invokinga probabilistic distribution model. A similar point of view is implicit in the supervisedlearning paradigm, which suggests a close analogy between the input–output mapping performedby a neural network and nonparametric statistical inference.

**3.** *Adaptivity.* Neural networks have a built-in capability to *adapt* their synaptic weights to changes in the surrounding environment. In particular, a neural network trained to operate in a specific environment can be easily *retrained* to deal with minor changes in the operating environmental conditions. Moreover, when it is operating in a

*nonstationary* environment (i.e., one where statistics change with time), a neural network may be designed to change its synaptic weights in real time. The natural architecture of a neural network for pattern classification, signal processing, and control applications, coupled with the adaptive capability of the network, makes it a useful tool in adaptive pattern classification, adaptive signal processing, and adaptive control. As a general rule, it may be said that the more adaptive we make a system, all the time ensuring that the system remains stable, the more robust its performance will likely be when the system is required to operate in a nonstationary environment. It should be emphasized, however, that adaptivity does not always lead to robustness; indeed, it may do the very opposite. For example, an adaptive system with short-time constants may change rapidly and therefore tend to respond to spurious disturbances, causing a drastic degradation in system performance. To realize the full benefits of adaptivity, the principal time constants of the system should be long enough for the system to ignore spurious disturbances, and yet short enough to respond to meaningful changes in the environment; the problem described here is referred to as the *stability–plasticity dilemma*

(Grossberg, 1988).

**4.** *Evidential Response.* In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to *select*, but also about the *confidence* in the decision made. This latter information may be used to reject ambiguous patterns, should they arise, and thereby improve the classification performance of the network.

**5.** *Contextual Information.* Knowledge is represented by the very structure and activation state of a neural network. Every neuron in the network is potentially affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally by a neural network.

**6.** *Fault Tolerance.* A neural network, implemented in hardware form, has the potential to be inherently *fault tolerant*, or capable of robust computation, in the sense that its performance degrades gracefully under adverse operating conditions.

For example, if a neuron or its connecting links are damaged, recall of a stored pattern is impaired in quality. However, due to the distributed nature of information stored in the network, the damage has to be extensive before the overall response of the network is degraded seriously. Thus, in principle, a neural network exhibits graceful degradation in performance rather than catastrophic failure. There is some empirical evidence for robust computation, but usually it is uncontrolled. In order to be assured that the neural network is, in fact, fault tolerant, it may be necessary to take corrective measures in designing the algorithm used to train the network (Kerlirzin and Vallet, 1993).

**7.** *VLSI Implementability.* The massively parallel nature of a neural network makes it potentially fast for the computation of certain tasks. This same feature makes a neural network well suited for implementation using *very-large-scale-integrated* (VLSI) technology.

One particular beneficial virtue of VLSI is that it provides a means of capturing truly complex behavior in a highly hierarchical fashion (Mead, 1989).

**8.** *Uniformity of Analysis and Design.* Basically, neural networks enjoy universality as information processors. We say this in the sense that the same notation is used in all domains involving the application of neural networks. This feature manifests itself in different ways:

• Neurons, in one form or another, represent an ingredient *common* to all neural networks.

• This commonality makes it possible to *share* theories and learning algorithms in different applications of neural networks.

• Modular networks can be built through a *seamless integration of modules.[[12]](#footnote-12)*

# Chapter (3):

## Information retrieval:

Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on metadata or on full-text (or other content-based) indexing.

Automated information retrieval systems are used to reduce what has been called "information overload”. Many universities and public libraries use IR systems to provide access to books, journals and other documents.

An information retrieval process begins when a user enters a query into the system. Queries are formal statements of information needs, for example search strings in web search engines. In information retrieval a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy.

An object is an entity that is represented by information in a database. User queries are matched against the database information. Depending on the application the data objects may be, for example, text documents, images, audio, mind maps or videos. Often the documents themselves are not kept or stored directly in the IR system, but are instead represented in the system by document surrogates or metadata.

Most IR systems compute a numeric score on how well each object in the database matches the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

A search engine maintains the following processes in near real time:

1. Web Crawling.
2. Indexing.
3. Searching.

Web search engines get their information by web crawling from site to site. The "spider" checks for the standard filename robots.txt , addressed to it, before sending certain information back to be indexed  depending on many factors, such as the titles, page content, headings, as evidenced by the standard HTML markup of the informational content, or its metadata in HTML meta tags).

Indexing means associating words and other definable tokens found on web pages to their domain names and HTML-based fields. The associations are made in a public database, made available for web search queries. A query from a user can be a single word. The index helps find information relating to the query as quickly as possible.

Some of the techniques for indexing, and caching are trade secrets, whereas web crawling is a straightforward process of visiting all sites on a systematic basis.

Between visits by the *spider*, the cached version of page (some or all the content needed to render it) stored in the search engine working memory is quickly sent to an inquirer. If a visit is overdue, the search engine can just act as a web proxy instead. In this case the page may differ from the search terms indexed. The cached page holds the appearance of the version whose words were indexed, so a cached version of a page can be useful to the web site when the actual page has been lost, but this problem is also considered a mild form of linkrot.

Typically when a user enters a query into a search engine it is a few keywords. The index already has the names of the sites containing the keywords, and these are instantly obtained from the index. The real processing load is in generating the web pages that are the search results list: Every page in the entire list must be weighted according to information in the indexes. Then the top search result item requires the lookup, reconstruction, and markup of the *snippets* showing the context of the keywords matched. These are only part of the processing each search results web page requires, and further pages (next to the top) require more of this post processing.

Beyond simple keyword lookups, search engines offer their own GUI- or command-driven operators and search parameters to refine the search results. These provide the necessary controls for the user engaged in the feedback loop users create by *filtering* and *weighting* while refining the search results, given the initial pages of the first search results. For example from 2007 the Google.com search engine has allowed one to *filter* by date by clicking "Show search tools" in the leftmost column of the initial search results page, and then selecting the desired date range.It’s also possible to *weight* by date because each page has a modification time. Most search engines support the use of the Boolean operators AND, OR and NOT to help end users refine the search query. Boolean operators are for literal searches that allow the user to refine and extend the terms of the search. The engine looks for the words or phrases exactly as entered. Some search engines provide an advanced feature called proximity search, which allows users to define the distance between keywords. There is also concept-based searching where the research involves using statistical analysis on pages containing the words or phrases you search for. As well, natural language queries allow the user to type a question in the same form one would ask it to a human. A site like this would be ask.com.

The usefulness of a search engine depends on the relevance of the *result set* it gives back. While there may be millions of web pages that include a particular word or phrase, some pages may be more relevant, popular, or authoritative than others. Most search engines employ methods to rank the results to provide the "best" results first. How a search engine decides which pages are the best matches, and what order the results should be shown in, varies widely from one engine to another. The methods also change over time as Internet usage changes and new techniques evolve. There are two main types of search engine that have evolved: one is a system of predefined and hierarchically ordered keywords that humans have programmed extensively. The other is a system that generates an "inverted index" by analyzing texts it locates. This first form relies much more heavily on the computer itself to do the bulk of the work.

Most Web search engines are commercial ventures supported by advertising revenue and thus some of them allow advertisers to have their listing ranked higher in search results for a fee. Search engines that do not accept money for their search results make money by running search related ads alongside the regular search engine results. The search engines make money every time someone clicks on one of these ads.

Google uses a very sophisticated and meticulous algorithm to get a page with text which has inbound links, internal links, appears in the title, is at the beginning of first page header. Google pretty well manages to differentiate between spams and real documents. In crawling they look at both the inbound and outbound link quality. Google is heavily based towards informational websites and web pages. Google first develops a trust among sites before ranking them. New pages on a new site do not develop trust easily. Google’s indexing system needs to handle hundreds of gigabytes of data and handle queries at a rate of hundreds to thousands per second. The search results may contain many relevant documents but the users are interested in only looking at a few first 10 results. While calculating the page rank algorithm Google takes care of the following: 1) Anchor Text: Anchor text is the text of links. In Google the page containing the link and the page to which the link points is both taken into consideration. 2) Meta Data Information: This is the information about the document such as the a. Reputation of the source b. Update Frequency c. Quality d. Popularity or Usage e. Citations.

**Procedure Involved**

1) The crawler is used to download the web pages. Usually 3 distributed crawlers are used. The lists of URL’s to be fetched is sent to the crawler by a URL server.

2) The URL’s fetched are then sent to the store server which compresses the documents and stores them in the repository, every web page now is given a unique ID called the DocID.

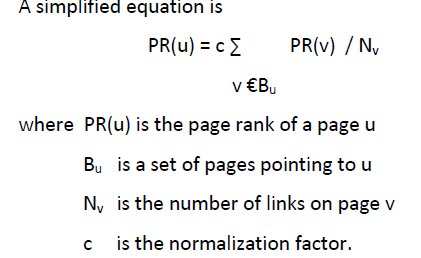
3) The Indexer and the Sorter are used together to perform indexing of web pages.

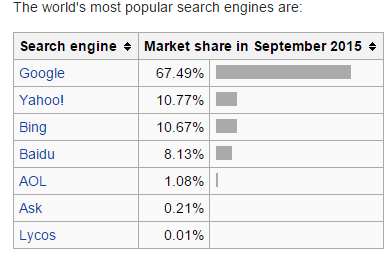
4) The Indexer reads from the repository, uncompressed the documents and parses them. It also converts the set of word occurrences called hits. It also records the word’s position, font size, capitalization etc and distributes these words into barrels and also creates a forward index. The other function of the indexer is to find out the no of links in every page and store it into the anchor file. The anchor file is used to record the no of inbound, outbound links and the text of the link.

5) The URL resolver takes the help of anchor file and converts relative URLs to absolute URLs or in other words the DocIds. The anchor text is used to provide information to forward index associated with the DocId and a database of links are created.

6) The sorter takes information from the barrels which are sorted by the DocId and resorts them by wordId to create the inverted index. After each document is parsed, it is encoded into a number of barrels. Every word is converted into a wordId by using an in memory hash table – Lexicon. Once the words are converted into wordIds, their occurrences in the current document are translated into hit lists and are written into forward barrels. The sorter takes each of the barrels and sorts it by WordId to produce an inverted barrel. This happens one barrel at a time and it can be done in parallel for a number of barrels at the same time to save time.  
**How to Rank**

The hit lists include position, font, capitalization, information. Also the anchor list I information and the page rank of the document is there. Page rank of a page is a sum of all the values of the links that point to it.[[13]](#footnote-13)



(3)

# Conclusion:

After reading the seminar we are sure that machines can think by lots of ways and there are three main types for leaning them:

**Supervised learning** is the machine learning task of inferring a function from labeled training data.

In **Unsupervised learning** unlike supervised learning, we are not told what the desired output is for each input.

Trial-and-error search and delayed reward--are the two most important distinguishing features of reinforcement learning.

Neural networks are similar to biological neuron networks and it has a lot of benefits like:

*Nonlinearity*, *Input–Output Mapping*, *Adaptivity*, *Evidential Response, Contextual Information.*

A technical neural network consists of simple processing units, the neurons, and directed, weighted connections between those neurons.

We can use machine learning and neural networks in many applications like: Information Retrieval which is used in web engines.

● a mobile robot decides whether it should enter a new room in search of more trash to collect or start trying to find its way back to its battery recharging station. It makes its decision based on how quickly and easily it has been able to find the recharger in the past.

Finally, this field can be developed and can be used in many other fields.

# Suggestions:

I propose to contact with SCS to teach us more about machine learning and its practical applications.

And I recommend to use it in Syria because it has a lot of positives although it requires hard work and time to perfect it.

# The Index:

Contents

[Introduction: 2](#_Toc437872968)

[Chapter (1): 3](#_Toc437872969)

[What is Machine Learning? 3](#_Toc437872970)

[Types of Learning: 4](#_Toc437872971)

[First (Supervised Learning): 4](#_Toc437872972)

[Second (Unsupervised Learning): 6](#_Toc437872973)

[Third (Reinforcement learning): 7](#_Toc437872974)

[Chapter (2): 9](#_Toc437872975)

[What Are Neural Networks? 9](#_Toc437872976)

[Components of neural networks: 10](#_Toc437872977)

[How Do Neural Networks Work? 11](#_Toc437872978)

[Benefits: 12](#_Toc437872979)

[Chapter (3): 15](#_Toc437872980)

[Information retrieval: 15](#_Toc437872981)

[Conclusion: 21](#_Toc437872982)

[Suggestions: 21](#_Toc437872983)

[The Index: 22](#_Toc437872984)

[Photo Index: 22](#_Toc437872985)

[References: 23](#_Toc437872986)

# Photo Index:

|  |  |  |
| --- | --- | --- |
| Number | Explanation | Page |
| 1 | Supervised Learning and Unsupervised Learning | 7 |
| 2 | Biological and Neural networks | 10 |
| 3 | The world’s most popular search engines | 20 |

# References:

., R. K. F. P. ((1998)). "Glossary of terms".

Alpaydin, E. Introduction to Machine Learning Second Edition. Massachusetts, The MIT Press.

Barto, R. S. S. a. A. G. Reinforcement Learning: An Introduction, The MIT Press.

Bishop, C. M. ((2006)). Pattern Recognition and Machine Learning.

Cannons, V. C. a. K. An Introduction to Neural Networks. Winnipeg, Manitoba, Canada.

Friedman, J. H. ((1998)). ""Data Mining and Statistics: What's the connection?"." 3–9.

Haykin, S. Neural Networks and Learning Machines Third Edition. Hamilton, Ontario, Canada.

Kriesel, D. A Brief Introduction to Neural Networks.

Mannila, H. ((1996)). Data mining: machine learning, statistics, and databases.

Mehryar Mohri, A. R., Ameet Talwalkar ((2012)). Foundations of Machine Learning, The MIT Press.

Murphy, K. P. Machine Learning A Probabilistic Perspective, The MIT Press.

Verma, A. (April 2013). "SEARCH ENGINE ALGORITHMS." International Journal of Advanced Research in Engineering and Applied Sciences.

Wernick, Y., Brankov, Yourganov and Strother (July 2010). Machine Learning in Medical Imaging.

1. ., R. K. F. P. ((1998)). "Glossary of terms".

   [↑](#footnote-ref-1)
2. Bishop, C. M. ((2006)). Pattern Recognition and Machine Learning.

   [↑](#footnote-ref-2)
3. Wernick, Y., Brankov, Yourganov and Strother (July 2010). Machine Learning in Medical Imaging.

   [↑](#footnote-ref-3)
4. Mannila, H. ((1996)). Data mining: machine learning, statistics, and databases.

   [↑](#footnote-ref-4)
5. Friedman, J. H. ((1998)). ""Data Mining and Statistics: What's the connection?"." 3–9.

   [↑](#footnote-ref-5)
6. Mehryar Mohri, A. R., Ameet Talwalkar ((2012)). Foundations of Machine Learning, The MIT Press.

   [↑](#footnote-ref-6)
7. Murphy, K. P. Machine Learning A Probabilistic Perspective, The MIT Press.

   [↑](#footnote-ref-7)
8. Alpaydin, E. Introduction to Machine Learning Second Edition. Massachusetts, The MIT Press.

   [↑](#footnote-ref-8)
9. Barto, R. S. S. a. A. G. Reinforcement Learning: An Introduction, The MIT Press.

   [↑](#footnote-ref-9)
10. Kriesel, D. A Brief Introduction to Neural Networks.

    [↑](#footnote-ref-10)
11. Cannons, V. C. a. K. An Introduction to Neural Networks. Winnipeg, Manitoba, Canada.

    [↑](#footnote-ref-11)
12. Haykin, S. Neural Networks and Learning Machines Third Edition. Hamilton, Ontario, Canada.

    [↑](#footnote-ref-12)
13. Verma, A. (April 2013). "SEARCH ENGINE ALGORITHMS." International Journal of Advanced Research in Engineering and Applied Sciences.

    [↑](#footnote-ref-13)