Machine-Learning: Issues, Definitions, and Algorithms

Machine-Learning Issues

There are problems such as the interpretation of handwriting, the recognition of images and the identification of trends that are difficult and sometimes impossible to solve by writing a traditional computer program in which we instruct the computer step-by-step how to solve the problem. The reason is that either it would be too complicated to embed in the program all of the conditions that must be considered in arriving at a solution or we simply do not know how to solve the problem.

The approach that machine-learning relies on to solve such problems is very different from the traditional computer programming approach. We devise an algorithm that the computer can use to look at thousands of examples and learn to recognise the correct solution to the problem. In other words, the computer is learning by example very much like young children are taught to distinguish between objects such as rabbits, deer, dogs, cats, horses, cars, buses, bicycles, and so on.

Over the past three decades many such algorithms that are designed to allow computers to learn by example have been developed and tested. The fundmental assumtion is that it is possible to generalize with a reasonable degree of accuracy from the analysis of a large corpus of data. However, how can we be justified in generalizing from what we have observed to what we have not observed? Is there any way to learn something from the past that we can be confident will hold true for the future? This is a fundamental problem for machine-learning. Analysis of a corpus of data may result in the appearance of one or more patterns. These apparent patterns may be false due to the features selected for the analysis not being sufficiently decisive (e.g., there being no single or small set of predictive factors)[1], or the corpus of data that has been analyzed not being large enough, or one or more primary features having been considered of minor importance or altogether overlooked.

Generalization: Philosophers have debated the induction dilemma ever since it was first posed by Hume in 1739 without coming to a definitive conclusion. It essentially means that we can never be certain that some conclusion that has held for all instances until now will continue to hold for all instances in the future. At best we can possibly say that it has a very high probability of holding for all instances in the future. The questions then arise: How important is absolute certainty and how critical are any potential exceptions? The answer to each of these questions depends on the individual circumstances. When Google responds to a search query it looks in its massive logs for the same or similar search queries that have been entered by users in the past and assumes that the links that these users have clicked on in the results pages are equally applicable to this particular search query. The penalty for this assumption being incorrect is not severe. At most a second try by the user with a slightly different query and an acknowledgement that the Google search algorithms require further fine-tuning. On the other hand, in a medical diagnosis where the wellbeing or even the very life of the patient is at stake, a wrong assumption may have dire consequences. In this case, the circumstances are exacerbated by the fact that no two human beings are exactly alike and no set of symptoms are a complete match for two patients.

Seed Knowledge: The appropriate conclusion is that data alone are not sufficient for successful machine-learning. What is also needed is knowledge, so that the theoretically determined enormous number of possible cases is reduced to a much smaller set of relevant cases by incorporating context in the machine-learning algorithm. This of course also introduces a degree of bias into the machine-learning algorithm. Although we typically consider bias leading to preconceived notions as undesirable in our daily activities, such preconceived notions are a very necessary component of machine-learning. In the case of human learning both our genes and the external interventions of our environment, particularly during our early formative years, embed preconceived notions and beliefs in our brain. These embedded notions, beliefs and biases are an essential part of our learning process. Similarly, the objective in machine-learning is to include just sufficient knowledge in

[1] In other words, there is no basis to select one generalization over another.

our program so that the learner can continue to acquire knowledge at infinitum by reading and synthesizing data. This is a gamble because machine-learning will not always make the correct decision. In fact, it is likely to make many mistakes. However, we can deal with this inaccuracy by discarding the misses and building on the hits. The cumulative result is what is important to us and that has been shown to be generally reliable.

How much seed knowledge does machine-learning need to be cumulatively successful? The nature of the knowledge that we embed in the learning algorithm appears to be quite primitive, but it is in fact very powerful. It starts with the selection of the features of the data that we consider to be relevant. This is indicative of the gamble that we face. If we chose the wrong features then the learning algorithm is unlikely to produce any useful results, or worse, the results may be misleading. Then we apply initial weights to these features on the understanding that these weights will be progressively automatically adjusted by the algorithm as it reads and synthesizes more and more data. In this respect, mechanisms that allow the learning algorithm to change features and adjust weights as it gains more knowledge are of critical importance.

Conjunctive Concepts: In the treaties *Principia* Newton enunciated his well-known three laws of motion as well as four much lesser known rules of induction. The third of his induction rules states: *Whatever is true of everything we have seen is true of everything in the universe.* While this rule represents an enormous leap to generalization, it is also at the very heart of machine-learning. We start our automated inductive learning process by applying the most widely applicable rules and then proceed to reduce the scope of these rules when the data forces us to do so. For example, we initially assume that the rate of car accidents is mostly governed by the speed at which a car is traveling. When the application of this rule to the data produces contradictory results we are forced to reduce the scope of the rule by adding a second rule; namely, that the age of the driver is also a factor. If the conjunction of these two rules fails as well then we add a third rule, such as male drivers are more likely to have accidents, and so on. Until we have a set of *conjunctive concepts*[2] that correlate with the data and allow us to fairly accurately predict the rate of car accidents. The larger the set of conjunctions, the greater the number of possible combinations of features that have to be analyzed, requiring an ever increasing amount of computation. It now becomes clear that machine-learning requires huge volumes of data to reduce the risk of false results and an enormous amount of computation due to many plausible conjunctions.

Rules: The problem with the conjunctive concepts approach is that it allows for neither exceptions nor alternative ways of achieving a goal. For example, birds fly unless they are ostriches, kiwis, penguins, or are housed in a cage. A chess game can be won in many different ways. However, the conjunctive concepts methodology can be used to generate a set of rules. After the system learns a rule all positive examples that it accounts for are removed from the data. Then the next rule is applied to the remaining data and all cases that it can account for are discarded, and so on until there are no remaining cases. For example, applying this to the previously discussed corpus of traffic accident data, we might apply the following set of rules: you are more likely to have a car accident *if you are under 25 years of age; if you have been drinking alcohol; if you are using your mobile phone; if you are driving in peak-hour traffic*. While rules are more powerful than conjunctive concepts, they are also more dangerous. For example, if a rule set were to cover the exact positive cases in a corpus of data, then it will predict that every new case that is not exactly like at least one of the positive cases is negative. In other words, the ability of computers to remember every detail of every instance is not especially useful for machine-learning.

Overfitting: Whenever machine-learning finds a pattern in the data that is not actually true in the context of our real world then we refer to that as *overfitting* the data. Unfortunately, learning algorithms are particularly vulnerable to finding patterns in data that do not exist in the real world. The reason is that the computer has the ability to process vast amounts of data and therefore an unlimited capacity to identify potential patterns. For example, *The Bible Code* published in 1989 became a bestseller due to its claim that the Bible contains potential

[2] Dictionary definitions are *conjunctive concepts*. For example, a chair has a seat, a back, and one or more legs. If any of these features are missing it is no longer a chair.

predictions of future events when one constructs new words by skipping letters at regular intervals. It turns out that there are so many ways of doing this that one is bound to construct something useful as long as the text is long enough. The claim was easily countered by demonstrating the same phenomenon in transcripts of U.S. Supreme Court rulings.

Overfitting is greatly exacerbated by noise in data, which equates to errors and random occurrences in machine-learning that can lead to false hypotheses. However, even without considering noise in data, for a learning algorithm based on conjunctive principles the number of possible instances of a conjunctive concept is an exponential function of the number of features (attributes). In the case of Boolean attributes, each new attribute doubles the number of possible instances by virtue of its required yes/no value. As an analogy, if we take the 64 squares on a chessboard and place two grains of wheat on the first square, four grains on the second square, and so on doubling the number of grains for each square, we end up with 2^{64} or over one trillion tons of wheat on the 64^{th} square alone[3].

Verification: It can be argued that machine-learning is a trade-off between the number of hypotheses considered and the amount of data available to test each hypothesis. While more data can exponentially reduce the number of positive hypotheses and increase the probability that a hypothesis is correct, there is no absolute certainty that the hypothesis will hold for all new data.

The question then arises: When can we believe that a hypothesis that appears to have been validated by a learning algorithm is in fact correct? The core response is that it has to be verified with new data. It is not sufficient for a new theory to explain past data. The theory must also hold true for new data. For example, Einstein's general relativity theory was only finally accepted after Eddington confirmed with experimental evidence its prediction that the sun bends the light from distant stars. To accomplish this verification in machine-learning it is common practice to divide the available data arbitrarily into two sections; - namely training data and validation data. The training data is used to create the hypothesis and the validation data is used to test it against new data. Testing the learning algorithm on new data is of critical importance in machine-learning. However, even then we cannot be sure that the possibility of overfitting has been eliminated. For example, in an early military application, a relatively simple learning algorithm was able to detect with almost 100% accuracy in both the training and testing data the presence of tanks in photographic images. It was later discovered that all of the images containing tanks were lighter than the other images and that was all that the algorithm had learned to base its recognition of tanks on.

Definitions

The following terms describe notions, concepts, reasoning mechanisms, and caveats that are either used by or relevant to most machine-learning algorithms.

abduction Abductive reasoning starts with an observation or set of observations and then seeks to find the simplest and most likely explanation.

classification The principal goal of *classification* is to predict the target class. If the trained model is intended to predict one of only two possible target classes then it is referred to as binary classification (e.g., predicting whether a student will pass or fail a course based on the student's profile). Multi-classification is the term used when the prediction involves more than two target classes (e.g., considering all subject details of a student to predict in which subject the student will achieve the highest score.

deduction Deductive reasoning progresses from one or more premises to a logically certain

[3] 2^{64} is equal to 1.8447×10^{19}. If there are 7,000 grains in one pound then there are 15,680,000 grains in one ton (i.e., 2,240 lb) and 1.176×10^{12} tons in 2^{64} grains of wheat. The weight of wheat required to fill the entire chess board is therefore far beyond the world's annual wheat production of approximately 65 million tons.

conclusion. Therefore, deductive reasoning moves from generalized principles that are accepted as being true to a specific conclusion.

empiricism Empiricists believe that all reasoning is fallible and that true knowledge must come from observation and experimentation. Typically, journalists, doctors, and scientists are empiricists. Empiricists prefer to try things to see how they turn out; - in computer science hackers and machine learners are empiricists. Aristotle was an early empiricist followed by Locke, Berkeley, and Hume in more recent times.

induction Inductive reasoning infers from a particular case to the general case. Therefore, inductive reasoning moves from specific instances to a generalized conclusion.

inverse deduction Instead of the classical model of starting with a premise and looking for the conclusions, *inverse deduction* starts with a set of premises and conclusions and works backward to fill in the gaps. Therefore, *inverse deduction* is a combination of inductive generalizations obtained by means of the comparative method (or by statistical method) and deduction from more ultimate laws. It is a way to arrive at reality through experiment, observation and conclusion.

overfitting Imagining or assuming patterns that are not really there. For example, the statistical correlation between the electricity consumption in one UK city with the suicide rate in another UK city that held true for several years, but was in fact a mere coincident. Machine-learning is particularly vulnerable to *overfitting* since the large amount of data involved provides an almost unlimited capacity to identify potential patterns.

rationalism Rationalists believe that the senses can be deceiving and that logical reasoning is the only sure path to knowledge. Typically, lawyers and mathematicians are rationalists. Rationalists prefer to plan everything in advance before making the first move; - in computer science theorists and knowledge engineers are rationalists. Plato was an early rationalist, followed by Descartes, Spinoza, and Leibnitz in more recent times.

regression The main goal of regression algorithms is to predict the value of a discrete or continuous variable[4]. The earliest form of regression was the *method of least squares*, which was published by Legendre (1805) and by Gauss (1809). Legendre and Gauss both applied the method to the problem of determining the orbits of bodies about the Sun from astronomical observations. Regression methods continue to be an area of active research. In recent decades, new methods have been developed for robust regression, regression involving correlated responses such as time series and growth curves, regression in which the predictor (i.e., independent variable) or response variables are curves, images, graphs, or other complex data objects, regression methods accommodating various types of missing data, non-parametric regression, Bayesian methods for regression, regression in which the predictor variables are measured with error, regression with more predictor variables than observations, and causal inference with regression.

[4] In mathematics a variable may be continuous or discrete. If the variable can take on any value from an infinite number of values within a range then it is referred to as a continuous variable (e.g., a variable representing the height of any persons; - i.e., no particular person). If the variable can only take on a value based on specified parameters then it is a discrete variable (e.g., the height of a Jack Jones; - i.e., a specific person).

Machine-Learning Algorithms

We have come a long way since the time that the principal use of the computer as a calculator or data processor required a detailed program of instructions that controlled the step-by-step operation of the computer. Machine-learning uses a different approach. Learning algorithms essentially program themselves by drawing inferences from large volumes of data. The following list of machine-learning algorithms is fairly comprehensive.

1. ***A/B testing***: Also known as *Randomized Controlled Trial (RCT)*; - particularly in respect to drug testing. A very simple method of testing that is however still widely used in business to test consumer preferences. For example, to determine whether a particular new feature on a company's website will increase sales it is tried out on thousands of randomly chosen customers and then compared with the results of the website without the new feature. To apply *A/B testing* to machine-learning we need to first form a hypothesis, work out a randomization strategy, decide on a sample size, and finally chose a measurement method.

2. ***autoencoder***: A neural network whose output is the same as its input. The value of the *autoencoder* is that its hidden layer has encoded the input (e.g., image) with far fewer bits that however include sufficient information for the output to be a replication of the input. In other words, the *autoencoder* has abstracted out of the input the salient characteristics that allow it to regenerate the complete input as output.

3. ***backpropagation***: After a neural network has processed the input to the output during its forward pass, the output is compared with the desired output. The error is propagated back through the layers of the network and the input. Each neurode adjusts its weighting based on the error and the input that it received during the forward pass. The *backpropagation* algorithm can deal with only one hypothesis at any one time. It proceeds deterministically and adjusts (i.e., learns) weights within a predefined structure (i.e., its network architecture).

4. ***chunking***: Our human brain handles the retention of information in chunks and this enables us to process much more information with greater efficiently. In learning a skill, we progressively build chunks of sub-problems that themselves consist of chunks. In a general sense *chunking* solves problems by reference to past experience that is also stored in chunks. Conversely, when a system has insufficient knowledge to solve a problem it forms a sub-goal. Any result that is produced in the sub-goal results in the creation of chunks. Once a chunk has been learnt the results of the impasse can be directly applied with the chunk(s) created in the sub-goal, if the same situation occurs again. In this way *chunking* speeds up problem solving (Saxena et al. 2017).

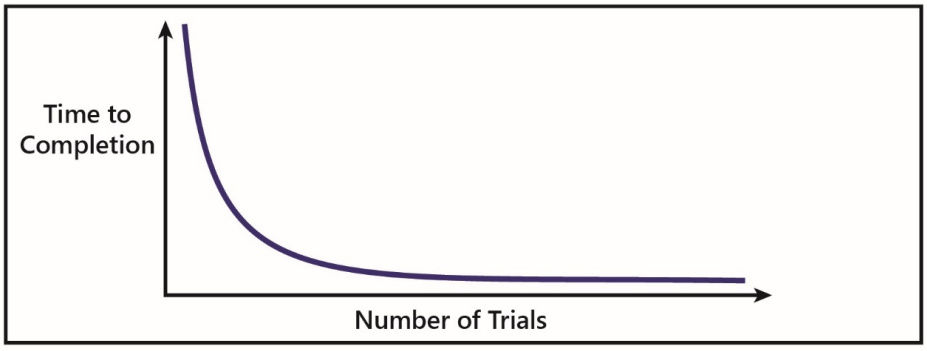

Figure 1: The Power Law

The concept of *chunking* was formally proposed in the late 1970s by Newell and Rosenbloom (1980) to explain why the rate of performance improvement in learning a skill is not constant. The rate falls off after an initial learning period, with the performance varying with time raised to some negative power (Figure 1); - referred to as the *Power Law*.

They argued that our ability to perceive and remember in chunks allows us to process much more information than we could if we had to deal with each information item individually. We can remember the telephone number 805-697-0034 much more easily than without separating hyphens (i.e., 8056970034). In learning a skill, we progressively build chunks that consist of chunks. For example, a major difference between a novice and an experienced chess player is that the experienced player judges a game by the positional pattern, while the novice player sees individual chess pieces.

Protocol studies have shown that we solve problems by decomposing them into multiple levels of sub-problems depending on the complexity of the problem. The solutions of these chunked sub-problems are essentially symbols that are then combined into a solution of the problem as a whole. It was initially thought that since *chunking* is a fundamental component of learning that a *chunking*-based algorithm may be sufficient for machine-learning. This led to the proposal by Newell, Laird and Rosenbloom (Laird et al. 1986; Rosenbloom 2006) of a general theory of cognition and the Soar general problem-solving program (Laird 2012). While Soar worked well within a predefined hierarchy of goals and was even able to define and solve new sub-problems, it ultimately failed to live up to the initial expectations. In particular, as Soar learned more complicated chunks the program slowed down instead of becoming faster. Even though *chunking* is not as prevalent in business applications of machine-learning as clustering, supervised learning and some other learning algorithms, it is recognized as an important contribution from psychology that seems assured of playing a role in the continued evolution of AI.

5. ***clusters:*** A *cluster* is a group of physical objects or non-physical entities that have similar characteristics, or are at least more similar to each other than members of other *clusters*. The ability to classify entities into categories and combine such *clusters* into a hierarchical structure of objects (i.e., treat a *cluster* or set of *clusters* as an object) is fundamental to human intelligence. It allows us to acquire skills by decomposing the skill into sub-skills and then combining those sub-skills to acquire a more comprehensive skill.

   The quest for an algorithm that can automatically group together entities into *clusters* is an intensely pursued area of research in machine-learning. A *cluster* typically has a prototypical set of attributes such as an average height or weight of a person in a *cluster* of people, even though none of its members may be of that exact height or weight. Or, the desirable product specifications of an abstract *cluster* such as a market segment or potential consumer group. Without any preexisting knowledge a clustering algorithm will need to start off by assuming that each new entity is part of a separate *cluster* unless it is in some ways similar to the entities in an existing *cluster*. With a computer being able to perform millions of computations per second such a clustering algorithm is able to fairly rapidly group thousands of entities into a hierarchical structure of *clusters* and sub-clusters based on some rules that define the desired degree of similarity that the members of a sub-*cluster* should adhere to.

6. ***Bayes theorem:*** The core concept of *Bayes theorem* is that the search for a solution should start off with an initial belief (i.e., hypothesis) and a subjective probability that this hypothesis is correct. As the data analysis produces new evidence the probability of the hypothesis is adjusted up or down accordingly. The following probability equation is used by the Bayes approach:

   $$P(\text{cause} \mid \text{effect}) = P(\text{cause}) \times P(\text{effect} \mid \text{cause}) / P(\text{effect})$$

   Example: influenza is the cause and fever is the effect. Data: out of 100 patients, 14 had influenza, 20 had a fever, and 11 had both influenza and a fever. The probabilities are as follows: fever being the effect of influenza is 11/14; patients having influenza is 14/100; and, patients having a fever is 20/100. The probability that patients with a fever have influenza is given by:

   $$P(\text{flu} \mid \text{fever}) = P(14/100) \times P(11/14) / P(20/100) = \mathbf{0.55\ (or\ 55\%)}$$

The initial probability, which is essentially a subjective belief, is referred to as the *prior probability* and the adjusted probability based on new data is referred to as the *posterior probability*. We usually know the probability of the effects given the cause; - i.e., the probability of a fever if the patient has influenza. However, what we would like to know is the probability of the cause given the effect; - i.e., the probability that the patient has influenza if the patient has a fever. If in the above example the physician had started with an intuitive estimate of the *prior probability* of 70% that a patient with fever has influenza, then this probability would now be reduced to 55%. Additional data would either increase or decrease the 55% that has now become the *prior probability*.

The application of *Bayes theorem* to machine-learning would normally require multiple effects such as (in the above example) sore throat, fatigue, prevalence of influenza in proximity of the patient, and so on, to be taken into account. The computational burden increases exponentially. If there are *n* effects and each carries the Boolean value of *yes* or *no*, then there are 2^n combinations that have to be calculated for each data set. In the case of only 10,000 data sets and only ten effects there are already over 10 million calculations (10000 x 2^{10} = 10000 x 1024 = 10,024,000). However, we are faced with two further problems. First, there are dependencies that require combinations of symptoms to be considered. A patient who has a fever and also a sore throat is more likely to have influenza than a patient with only one of those symptoms. Second, we need an enormous amount of data to have some confidence that the available data covers the combinatorial range of different cases and that there are a sufficient number of instances of each case. To deal with the combinatorial explosion problem we are forced to make concessions such as, we assume that the effects are independent of each other given the cause. This is referred to as the Naïve Bayes Classifier. Naïve Bayes is widely applied to e-mail spam filters, search engines, text classifiers, and in many other domains. In fact, at this time (2018) it may still be the most widely used machine-learning algorithm (Russell and Norvig 2012; Domingos 2015, 152).

7. ***Bayesian network:*** If we make the hypothesis (i.e., belief) the cause and the data the effect then the hypothesis can be as complex as a whole network or as simple as the probability that a flipped coin will come up heads.

$$P(\text{hypothesis} \mid \text{data}) = P(\text{hypothesis}) \times P(\text{data} \mid \text{hypothesis}) / P(\text{data})$$

Pearl (1988) introduced a major extension to the application of *Bayes theorem* by proposing the concept of a *Bayesian network* to represent dependencies among variables, with the limitation that each variable depends directly on only a few other variables. One of Pearl's examples deals with a burglar alarm installed in a home that can be triggered by a structural movement caused by a forced entry or some other impacting force. Neighbor A calls to tell you that your burglar alarm has just gone off. However, neighbor B does not. Should you call the police?

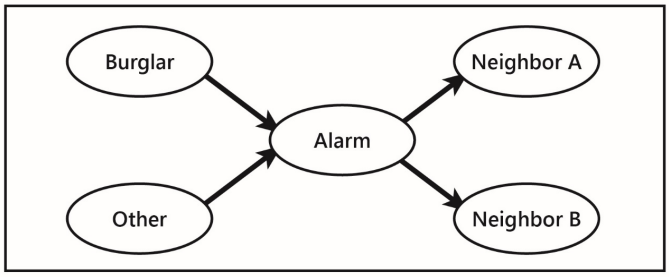

Figure 2: Perl's Bayesian network concept

In the above *Bayesian network* (Figure 2) Alarm depends on Burglar and Other, while Neighbor A and Neighbor B each depend only on Alarm. In other words, Neighbor A's call is *conditionally independent* of Burglar and Other, while the Neighbor A and B effects are independent of each other given the Alarm cause. Each of the variables includes a table listing the probability for each

combination. Neighbors A and B require only two probabilities each (i.e., alarm on or not on), Burglar and Other require only one probability each because they have no parent, and Alarm has four probabilities (i.e., Alarm on due to burglar, Alarm on due to Other, Alarm on without Burglar or Other, and Alarm on due to both Burglar and Other). The key point is that the assumption of *conditional independence* reduces the number of probabilities that need to be considered from 32 (i.e., $2^5 = 32$) to just 10 (i.e., 1+1+4+2+2 = 10), which is a 68.75% reduction in combinations that need to be pursued. This is just a trivial example. With a more typical *Bayesian network* involving over a hundred variables the combinatorial reduction would be much greater.

After using *Bayesian networks* to develop an e-mail spam filter in 1997 at Microsoft, David Heckerman applied a similar approach in the fight against AIDS (Kim 2016). Since the AIDS virus mutates rapidly it is difficult for any one drug or vaccine to be effective. In the spam filter project Heckerman had realized that to identify spam it is sufficient to look for particular words, without needing to understand the e-mail message. However, this became a cat and mouse game because spammers will change specific characters in words to bypass spam filters based on this strategy. For example, the word *Viagra* that may be an indicator of spam would be changed to *Viagr@*. To overcome this problem Heckerman changed his approach and looked for any basic vulnerability in a spam message. He found a vulnerability that was connected to the ultimate aim of the spammer. To collect payment from the recipient it is necessary to include a URL[5] address. Similarly, in the case of AIDS there are small regions in the virus protein that cannot change without hurting the virus. Heckerman and his team used a *Bayesian network* to identify such vulnerable regions as a basis for the development of a vaccine that would allow the immune system to attack just those regions.

It can be argued that virtually every problem can be formulated in terms of probabilistic inferencing and a Bayes approach is often the only practical method available. Conditions that are particularly amenable to a Bayes method include problems involving multiple sources of information (e.g., image and laser range data), hierarchical models in which the hyper-parameters for the dataset are unknown, complex models, and problems that require the use of domain knowledge. Critics of Bayes methods point to the subjective assignment of the prior probability, the false *conditional independence* assumption, the inability to combine hypotheses, and the many computational difficulties that can be involved in the *Bayes network*-based machine-learning approach.

8. ***Boltzmann Machine:*** A type of recurrent neural network in which there are sensory and hidden neurons. The weightings between neurons are adjusted statistically according to the Boltzmann distribution, which is a probability measure that a system will be in a certain state as a function of given criteria. *Boltzmann Machines* can be strung together to make more sophisticated systems such as *deep belief networks*. In 1985, Ackley, Hinton and Sejnowski replaced the deterministic neurodes in Hopfield networks with probabilistic neurodes (Ackley et al. 1985). This gave neural networks a probability distribution over its states. Since the probability of the neural network being in a particular state could now be determined with the Boltzmann distribution from thermodynamics, their neural network became referred to as a *Boltzmann Machine*.

9. ***Decision Tree:*** A hierarchical (tree-like) structure in which the properties of each node differ by at least one attribute. Therefore, each branch of a *Decision Tree* can be represented by one rule. A *Decision Tree* model is a classification system that classifies future observations based on a set of decision rules, with maximum accuracy. The *Decision Tree* algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the *Decision Tree* algorithm can be used for solving both regression and classification problems. Typically, the *Decision Tree* algorithm is used to create a training model for predicting the class or value of

[5] Uniform Resource Locator (URL).

variables by learning decision rules inferred from prior data (i.e., training data). The representation is in the form of a tree, with each internal node of the tree corresponding to an attribute and each leaf node corresponding to a class label (Figure 3).

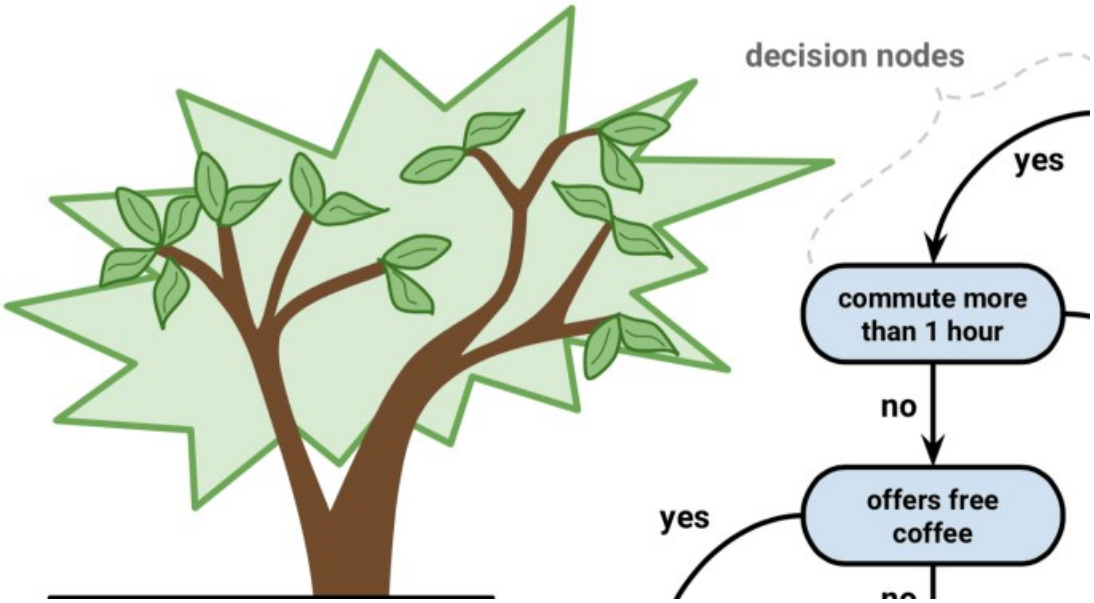

Figure 3: Decision Tree classifier (*Image credit:* www.packtpub.com)

The *Decision Tree* algorithm proceeds as follows: (a) places the best attribute of the dataset at the root of the tree: (b) splits the training set into subsets so that each subset contains data with the same attribute value; and, (c) repeats steps (a) and (b) for each subset until it finds leaf nodes in all the branches of the tree. In predicting the class label for a data record the algorithm starts at the root of the tree. It compares the values of the root attributes with the data record's attributes, follows the branch corresponding to that value, and then jumps to the next node. The algorithm continues to compare the data record's attribute values with other internal nodes of the tree until it reaches a leaf node with the predicted class value.

10. ***Deep Belief Networks:*** A class of multi-layer neural networks in which the layers, but not the neurons within the layers, are connected. When trained on a set of examples without supervision, a *deep belief network* can learn to probabilistically reconstruct its inputs, with the layers acting as feature detectors. After this initial learning step, a *deep belief network* can be further trained with supervision to perform classification.

    A *deep belief network* can be viewed as a composition of simple, unsupervised networks such as restricted Boltzmann Machines[6] or Autoencoders, in which the hidden layer of each sub-network serves as the visible layer for the next. This composition leads to a fast, layer-by-layer unsupervised training procedure, where contrastive divergence is applied to each sub-network in turn, starting with the lowest pair of layers. The lowest visible layer is a training set. The fact that *deep belief networks* can be trained greedily[7], one layer at a time, led to one of the first effective *deep learning* algorithms.

11. ***dimensionality reduction:*** Used to reduce the number of dimensions of data to a more manageable set by defining characteristic attributes. In trying to automatically match the photograph of a robbery suspect with the photographs of thousands of known criminals, pixel by pixel, we can establish a few

[6] A restricted Boltzmann Machine is an undirected, generative energy-based model with a visible input layer and a hidden layers and connections between but not within layers.

[7] A greedy algorithms makes the locally optimum choice at each stage, with the intent of finding the global optimum.

dozen facial attributes to distinguish the faces of different persons. For example, since a face has only about 50 major muscles it is possible to represent virtually all possible expressions in less than a hundred numbers[8] instead of over 100,000 pixels.

12. ***Expectation Maximization (EM) algorithm:*** A statistical classification model that is capable of clustering cases when crucial information such as the classes of the training data are unknown. EM can fractionally assign a case to two clusters and at the same time update their descriptions accordingly. Like the *k-means algorithm* it alternates between assigning cases to clusters and updating the descriptions of the clusters.

13. ***Hebb's Law:*** The law states that if cell A repeatedly takes part in firing cell B then some metabolic change occurs in A and/or B so that A's efficiency in causing B to fire is increased. Hebb's law is the foundation of connectionism (Hebb 1949). It was the amalgamation of concurrent ideas from psychology (James 1890) and neuroscience with a certain amount of speculation, at a time when learning by association became a prevalent theory and Cajal was for the first time able to observe the action of neurons using the Golgi stain (Cajal 1894).

14. ***genetic algorithm:*** A heuristic search method based on the theory of natural selection (i.e., selection, mutation, inheritance, and recombination) that incorporates a fitness function. The fitness function assigns a numeric score as a measure of how well the current state fits the purpose. In this way a *genetic algorithm* replaces the selective breeding process of plants and animals with a learning algorithm and reduces the time between generations to a few seconds of computer time. Holland (1992) recognized the importance of incorporating the mechanism of sexual reproduction in the learning algorithm. In reproduction two new chromosomes are produced (Figure 4); - one consisting of the mother's chromosome before and the father's after reproduction and the other consisting of the father's chromosome before and the mother's after reproduction.

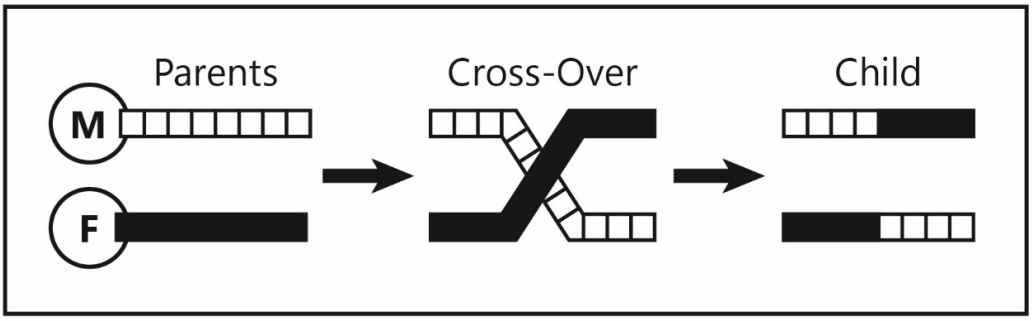

Figure 4: Sexual reproduction (*Image credit:* Domingos 2015)

Based on a fitness function the *genetic algorithm* creates variations that can be evaluated according to meeting the fitness goal. Similar to the way that in nature DNA encodes an organism as a sequence of chromosome pairs, the *genetic algorithm* uses a string of bits. During each generation the algorithm creates the fittest cases in the data by crossing over their bit strings at a random point in time. In this virtual computational world, each mutated case receives a fitness score, with the result that each computed generation is fitter than the previous one. The process terminates either when the desired fitness level has been reached or time runs out.

15. ***genetic programming:*** Is a special form of the *genetic algorithm* concept where the output are programs instead of functions (Koza 1992). Koza took Holland's work a major step forward by replacing the bit strings in *genetic algorithms* with software code, arguing that a computer program

[8] With only 10 choices for each facial feature, a law enforcement artist can draw a portrait of a suspect that is often good enough to recognize that suspect.

is really a tree of subroutines. Using crossover (Figure 5) and mutation to swap subroutines between program trees he was able to show that genetic programs can include a wide range of programming constructs such as if-then comparisons, loops, and recursion.

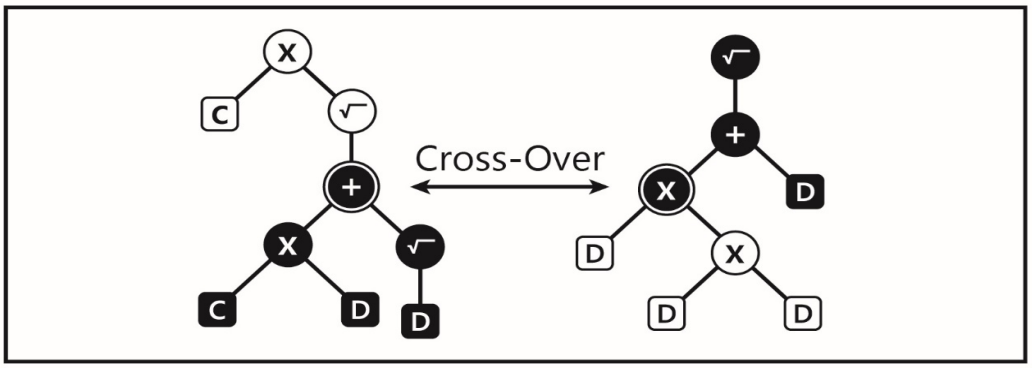

Figure 5: Crossover in genetic programming (*Image credit:* Domingos 2015)

Genetic programming's first major success in the mid-1990s was in the design of electronic circuits (Koza et al. 1999). Koza was able to reinvent a previously patented design for a low-pass filter that could be used to selectively enhance a particular frequency band in a musical recording (Carnett and Heinz 2006). In 2005, the US Patent Office awarded a patent to a factory optimization system that was designed using *genetic programming*.

16. ***gradient descent:*** Using a *backpropagation* algorithm, after thousands of repetitions the neural network's output neurodes will closely match the desired output that recognizes the particular input that it was trained to recognize. During this *gradient descent* operation there is no certainty that the neural network has reached the best (i.e., optimum) output condition even after thousands of iterations; - i.e., it may have reached a *local minimum* of the error. However, experience has shown that this is not as serious a drawback to *backpropagation* as originally thought. First, a local minimum may be quite acceptable since in most cases the error plane resembles a quilt with many peaks and troughs. Second, the local minimum is less likely to have *overfitted* the data than if we were to insist on reaching the global minimum.

17. ***Hidden Markov Model (HMM):*** A *Markov Chain* where the states are hidden. For example, in a speech-recognition system such as Siri the observations are the sounds spoken to Siri and the hidden states are the written words that the sounds are intended to represent. The probability of the next word given the current word is a *Markov Chain*.

18. ***Hopfield network:*** A simple neural network model with feedback connections, in which the neurodes are binary threshold units (i.e., they take on only two different values for their states and the values are determined by whether or not the unit's input exceeds its threshold). Hopfield nets normally have units that take on values of 1 or -1. The significance of the Hopfield network lies in the fact that it proposed a model of associative memory (i.e., to simulate human memory). Computer memory is indexed memory like a telephone directory that lists the name, address and telephone number of each subscriber. If we know the name, we can find the telephone number and the address. However, if we know either the telephone number or the address the telephone directory does not help us to find the name. In associative memory the information is stored as a network of items (e.g., name, address and telephone number linked together). Knowing any one of those information items we can find either of the other two.

    The Hopfield Network is composed of a single group of neurodes (Figure 6) that are all connected to each other and the external world, as both input and output. There are no hidden layers. While the connections are bidirectional, the weightings at the connections are the same for both directions. In a

common variation, a network would have activation levels that can be only 0 or 1. Typically, if the sum of the weighted inputs is above 0 then the activation is 1, otherwise the activation is 0. Updates are implemented asynchronously, with neurodes selected randomly for update purposes. Figure 7 shows how the Hopfield Network would learn to recognize the alphabetic character "H". An image of the "H" (i.e., the data) is stored in the memory of the network. Then a distorted image of "H" is entered into the network as input (Craig Will refers to this image in Figure 7 as a "noisy version".)

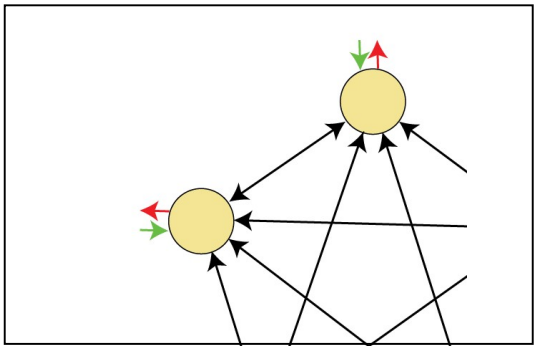

Figure 6: Illustration of a Hopfield Network

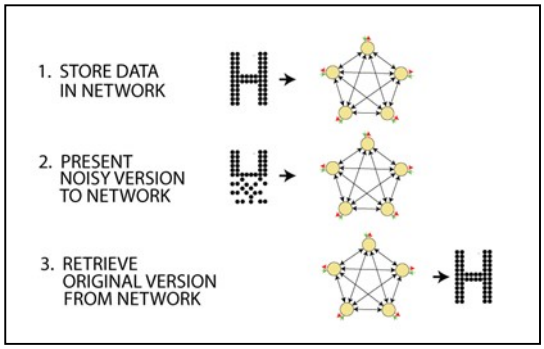

Figure 7: How a Hopfield Network works

(*Image credit:* Craig Will [https://principlesofdeeplearning.com/index.php/2018/10/11/historical-material-the-hopfield-network])

The initial state of the network will be the distorted image. As the network goes through its cycles the weights of the neurodes will be updated as each neurode receives input signals from the other neurodes. During this repetitive process the network is learning to replicate the good image of "H" through the updated values of the weights that are applied to each neurode.

19. ***Kalman filter:*** A Hidden Markov Model (HMM) in which the states and the observations are continuous variables instead of discrete variables.

20. ***k-means algorithm:*** The *k-means* learning algorithm is used to cluster unlabeled data (i.e., data without defined categories). It is an unsupervised clustering algorithm that aims to partition cases into *k* clusters with a case going to the cluster with the nearest mean. If *k* is predetermined then the greater the value of *k* the smaller the value of the metric that defines the differences between clusters. One of the metrics that is commonly used to compare results across different values of *k* is the mean distance between data points and their cluster centroid.

    The *k-means algorithm* was first proposed by Stuart Lloyd at Bell Labs in 1957 (Lloyd 1982). While it is simple and popular, it has at least one serious hurdle to overcome. The *k-means algorithm* works only if the clusters are well differentiated. This hurdle can be overcome to some extent by providing external assistance. For example, by training the *k-means algorithm* with the attributes of members of existing clusters and the probability associated with each attribute. Another approach is to reduce the number of possible similarity dimensions through a process of *dimensionality reduction*. In the case of clustering images, we can reduce the number of visible differences (i.e., dimensions) at the pixel level to a much smaller number of combined features.

21. ***k-nearest-neighbor:*** A refinement of the *nearest-neighbor* algorithm in which not one but multiple nearest neighbors of the case under consideration determine (by vote) whether the case belongs to the training test case.

22. ***Markov Chain:*** A stochastic model[9] describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event (i.e., satisfies the

[9] In probability theory, a stochastic model is defined as a set of random variables. It assumes that future states depend only on the current state and not on any events that occurred before the current state.

Markov property). A process satisfies the Markov property if one can make predictions for the future of the process based solely on its present state, just as well as one could knowing the full history of the process. For example, Google's PageRank algorithm is a *Markov Chain*[10]. The following is a simple example of the application of the *Markov Chain* model to a business case. Assume that company A and company B are the predominant commercial aircraft manufacturers that share almost 100% of the global commercial aircraft market (i.e., company A has 60% share and company B has 40% share). A major airline company is considering a multi-year exclusive contract for the renewal of its aircraft fleet with one of these two aircraft manufacturers. The company hires a market research consultant to determine which of the two manufacturers is likely to increase its market share after one year. The consultant's research indicates the following results:

P(A to A): Probability of airline customer staying with company A over 12 months …….. = 0.7
P(A to B): Probability of airline customer switching from company A to B over 12 months ….. = 0.3
P(B to B): Probability of airline customer staying with company B over 12 months …….. = 0.9
P(B to A): Probability of airline customer switching from company B to A over 12 months ….. = 0.1

Applying the *Markov Chain* algorithm to determine the likely market shares of the two aircraft manufacturers (A and B) after one year:

market share in 12 months of A = current share of A x P(A to A) + current share of B x P(B to A)
= (0.60 x 0.7) + (0.40 x 0.1) = **0.46**

market share in 12 months of B = current share of B x P(B to B) + current share of A x P(A to B)
= (0.40 x 0.9) + (0.60 x 0.3) = **0.54**

Although aircraft manufacturer A has a higher market share today that market share is likely to reduce within the next year. If the transition probabilities determined by the marketing consultant do not change over the near term then we can predict the market shares in 24 months, as follows:

market share (24 months) of A = share (12 months) of A x P(A to A) + share (12 months) of B x P(B to A)
= (0.46 x 0.7) + (0.54 x 0.1) = **0.38**

market share (24 months) of B = share (12 months) of B x P(B to B) + share (12 months) of A x P(A to B)
= (0.54 x 0.9) + (0.46 x 0.3) = **0.62**

This simple example amplifies the utility of the Markov property assumption, namely; - *the next state of the process depends only on the previous state and not the sequence of states*. This assumption makes the calculation of conditional probabilities easy and enables the *Markov Chain* algorithm to be applied in a number of scenarios. However, in most cases the more evolved *Hidden Markov Model (HMM)* is preferred.

23. ***Markov Chain Monte Carlo (MCMC):*** While *Markov Chain* involves a sequence of steps each of which depends only on the previous step, MCMC adds an element of chance to the *Markov Chain* method with random sampling. For example, to determine the area taken up by a circle that is inscribed within a square (of known dimensions) we could take a felt pen, close our eyes and randomly make a mark inside the square. If we repeat this many times then the proportion of dots that fell into the circle will allow us to estimate the area of the circle.

[10] Google's proprietary algorithm PageRank ranks the relative *quality* of every site on the Internet in the range of 0 to 10. A site with a score of 1 has twice the *quality* of a site with a score of 0 and a site with a score of 2 has twice the *quality* of a site with a score of 1, and so on. *Quality* is based on many factors such as size, content, number of links, download options, etc. While Google performs a hypertext-matching analysis to find the site most relevant to a search query, it also analyzes how the search words are used by the page and neighboring pages of that site. Since PageRank has already served as a filter to identify the most likely relevant sites Google does not have to search the entire Internet.

24. ***Markov Network:*** Also referred to as *Markov Random Fields*, is capable of compactly representing and visualizing a probability distribution that is based on the language of *undirected graphs*[11]. A *Markov Network* represents relationships between random variables. For example, Roger and Sue are both musicians and friends. However, while these two variables depend on each other neither is the cause of the other.

25. ***Nearest-Neighbor algorithm:*** Looks for a similar image or case and then assumes the image or case under consideration to be of the same classification as the similar one. For example, when a Facebook user uploads a photograph of a person how does Facebook recognize the image as being the face of a person? It looks through its database of images to find another image that contains enough similar features to make it highly probable that it is of the same kind. If this image is known by Facebook to be the face of a person then the just uploaded photograph is also classified as containing the face of a person.

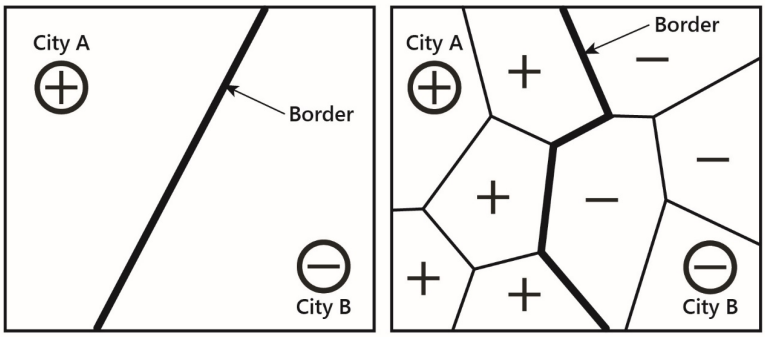

Figure 8: Nearest-neighbor algorithm example (*Image credit:* Domingos 2015)

The apparently trivial capabilities the *Nearest-Neighbor algorithm* can be applied in a surprisingly sophisticated manner. Let's assume that we want to determine the approximate border between two states in the US. The *Nearest-Neighbor algorithm* might start with the hypothesis that the border is a straight-line half-way between the capital cities of the two states (Figure 8). By taking into consideration a large number of towns on either side of the border it is then able to construct an intricate border based on just the location of each town and the state that the town belongs to.

26. ***Newton's Third Principle:*** Whatever is true of everything we have seen is true of everything in the universe. This principle is foundational for machine-learning. The learning process normally starts with the most widely applicable rules and then progressively narrows the solution space by adding constraints in the form of additional rules.

27. ***perceptron:*** A perceptron is essentially a simplistic mathematical simulation of a biological neuron. First proposed in 1943 (McCulloch and Pitts 1943) the original model of an artificial neuron (or neurode) was able to only represent OR, AND, and NOT gates. By adding variable weights to the connections between neurodes Rosenblatt (1958) showed that a network of neurodes (i.e., neural network) can learn. The name *perceptron* was coined due to Rosenblatt's research focus on perceptual tasks such as speech and character recognition. While the *perceptron* was a major step forward in AI it was also found to have serious shortcomings, which Minsky and Papert drew attention to in their book published in 1969 (Minsky and Papert 1969). In particular, the *perceptron* was not able to deal with an exclusive-OR function (i.e., XOR)[12].

[11] In an *undirected graph* all connections between nodes are bidirectional. A graph where the connections between nodes point in a direction is sometimes referred to as a *directed graph*.

[12] For example, best customers of Nike sports shoes are teenage boys and middle-aged women. Young is good and female is good, but young and female is not good.

28. ***Power Law:*** (see *S-curve*).

29. ***Principal Component Analysis (PCA):*** A method used in *dimensionality reduction* to reduce the number of variables in a corpus of data by representing the data in a different format. For example, creating a linear graph by plotting non-linear data in a logarithmic form or changing (i.e., rotating) the axis of a graph to reduce the distance of data points from the axis (Domingos 2015, 211-214).

30. ***reinforcement learning:*** Replaces instant gratification with longer term reward seeking. While most machine-learning algorithms are designed to take advantage of any immediate rewards in their decision-making process, a *reinforced learning* algorithm is typically willing to forego an immediate reward in favor of a potential future more favorable reward; - even to the extent of sometimes choosing a random action (Domingos 2015, 218-223). The analogy is with an experienced chess player who may be willing to sacrifice a chess piece in favor of a positional advantage that may lead to a winning game.

31. ***relational learning:*** Treats data not as unrelated entities but as a complex network of related nodes that can be applied from one situation to another similar situation. For example, having learned how electricity consumption varies with the time of day and season in one city it can be applied to another city with somewhat similar characteristics. However, whereas in regular learning all cases must adhere to the same number of attributes, in *relational learning* the networks can vary in size (Domingos 2015, 227-233).

32. ***S-curve:*** Also referred to as the *Power Law*. Reflects the phase transition of all kinds of phenomena (e.g., magnetization of iron, electron flipping its spin, ice melting, water evaporating, developments in technology, rumors, epidemics, etc.). As shown in Figure 9, at first the output increases very slowly with the input but then gradually increases more and more until it reaches a first upward knee where it increases exponentially. It then reaches a second downward knee where the output rapidly decreases, only to gradually level out again until the output decreases slowly with the input.

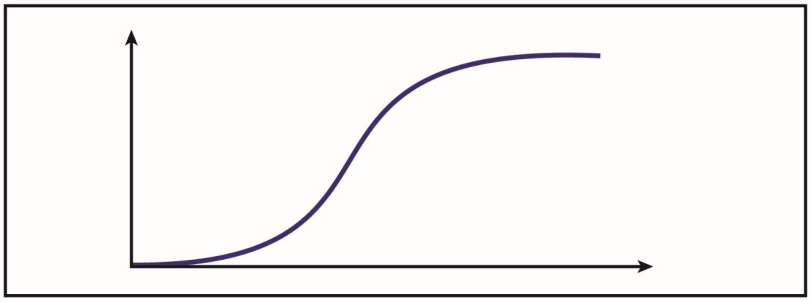

Figure 9: The sigmoid or S-curve

33. ***Support Vector Machine (SVM):*** Also known as *Support Vector Network*, *SVM* is a supervised learning model with a learning algorithm that classifies data. Given a training set that includes two categories of cases, the *SVM* learning algorithm builds a model that classifies new cases into either of the two categories represented in the training data. The *SVM* concept was developed by Vladimir Vapnik at Bell Labs after he immigrated to the US from Russia in 1990 (Ben-Hur et al. 2001). *SVM* is similar to a weighted *k-nearest-neighbor* algorithm. The division between the positive and negative cases[13] is defined by a set of weighted examples together with a similarity measure. A test example is considered to belong to the positive class if, on average, it looks more like the positive examples than the negative examples.

    SVM algorithms normally use constrained optimization to limit the growth of the weights applied to

[13] The technical term is classes (i.e., instead of cases) because the *nearest-neighbor*, *k-nearest-neighbor* and *SVM* algorithms are categorized as classifiers.

examples. The reason is that the problems that the *SVM* methodology is typically applied to in industry have resource constraints (e.g., the number of items that a factory can produce is constrained by the availability of tools, material and labor). Domingos (2015, 193) explains the concept of constrained optimization with the following example. If you are driving in a car with the objective of getting as close as possible to the top of the mountain, then constrained optimization means that you will go as high as possible without driving off the road. Unconstrained optimization would require you to reach the top regardless of whether or not the road goes right up to the top of the mountain[14].

References:

Ackley D, G. Hinton and T. Sejnowski (1985); 'A Learning Algorithm for Boltzmann Machines'; Cognitive Science, 9 (pp. 147-169).

Ben-Hur A., D. Horn, H. Siegelmann and V. Vapnik (2001); 'Support vector clustering'; Journal of Machine Learning Research, 2 (pp. 125–137).

Box G. (1976); 'Science and Statistics'; Journal of the American Statistical Association, 71(356), December (pp. 791-9).

Cajal S. (1894); 'The Croonian Lecture: La fine structure des centres nerveux'; Proceedings of the Royal Society of London, 55 (pp. 444-468).

Carnett J. and E. Heinz (2006); 'John Koza Has Built an Invention Machine'; Popular Science, 18 April.

Cohen B. and A. Whitman (1999); 'Isaac Newton, The Principia: Mathematical Principles of Natural Philosophy (The Authoritative Translation)'; University of California Press, Oakland, California.

Dietterich T. and R. Michalski (1986); 'Learning to Predict Sequences'; in Michalski, Carbonell and Mitchell (eds.), Machine Learning: An Artificial Intelligence Approach, Vol. II, Morgan Kaufman, Los Altos, California (pp. 63-106).

Domingos P. (2015); 'The Master Algorithm: How the Quest for the Ultimate Learning Machine will Remake the World'; Basic Books, New York, New York.

Drosnin M. (1997); 'The Bible Code'; Touchstone, New York, New York.

Eisenberg J. (2018); 'John Stuart Mill on History: Human Nature, Progress, and the Stationary State'; Lexington Books, Lanham, Maryland.

Fix E. and J. Hodges (1951); 'Discriminatory analysis – nonparametric discrimination – consistency properties'; Technical Report, DTIC Document.

Gentner D. (1983); 'Structure-mapping: A theoretical framework for analogy'; 7(2), Cognitive Science, April-June (pp. 155-170).

Gauss C. (1809); 'Theoria Motus Corporum Coelestium in Sectionibus Conices Solem Ambientum'; English translation by C. Davis reprinted in 1963, Dover, New York, New York.

Hebb D. (1949); 'The Organization of Behavior'; John Wiley & Sons, New York, New York.

Holland J. (1998); 'Emergence: From Chaos to Order'; Perseus Books, Reading, Massachusetts.

Hofstadter D. and E. Sander (2013); 'Surfaces and Essences: Analogy as the Fuel; and Fire of Thinking'; Basic Books, New York, New York.

[14] It follows that *gradient descent* is therefore an unconstrained optimization technique.

Hofstadter D. (1999); 'Gödel, Escher, Bach: An Eternal Golden Braid'; Basic Books, New York, New York.

Holland J. (1992); ,Genetic Algorithms: Computer Programs that evolve in ways that resemble natural selection can solve complex problems even their creators do not fully understand'; Scientific American, July (pp.66-72).

Hume D. (2000); 'A Treatise of Human Nature', in Norton D. and M. Norton (eds.) Oxford Philosophical Texts, Oxford University Press, Oxford, UK.

Husain A. (2017); 'The Sentient Machine: The Coming Age of Artificial Intelligence'; Simon & Schuster, New York, New York.

Hutter M. (2004); 'Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability'; Springer, Heidelberg, Germany.

Ioannidis J. (2005); 'Why Most Published Research Findings are False'; *PLoS Medicine*, 2(8):e124, August (pp. 696-701).

James W. (1890); ' The Principles of Psychology'; Volumes 1 and 2, Henry Holt and Company, New York, New York.

Kim M. (2016); 'Microsoft Research scientist David Heckerman on how we could attack HIV like spam'; The Washington Post, Health Section, 4 March.

Koza J. (1992); 'Genetic Programming'; MIT Press, Cambridge, Massachusetts.

Koza J., F. Bennett, D. Andre and M. Keane (1999); 'Genetic Programming III'; Morgan Kaufmann, San Francisco, California.

Laird J. (2012); 'The Soar Cognitive Architecture'; MIT Press, Cambridge, Massachusetts.

Laird J., P. Rosenbloom and A. Newell (1986); 'Chunking in Soar: The anatomy of a general learning mechanism'; Machine Learning, 1(1) (pp. 11-46).

Legendre A. (1805); 'Nouvelles methodes pour la determination des orbites des cometes'; (Appendix: Sur la methode des moindres quarres), Firmin Didot, Paris, France.

Lloyd S. (1982); 'Least squares quantization in pcm'; IEEE Transactions on Information Theory, 28(129) (pp. 137).

McCulloch W. and W. Pitts (1943); 'A logical calculus of the ideas imminent in nervous activity. Bulletin of Mathematical Biophysics, vol. 5 (pp. 115-133).

McGrayne S. (2011); 'The Theory that Would Not Die: How Bayes' rule cracked the enigma code, hunted down Russian submarines, and emerged triumphant from two centuries of controversy'; Yale University Press, New Haven, Connecticut.

Michalski R. (1980); 'Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning data into Conjunctive Concepts'; *Journal of Policy Analysis and Information Systems*'; 4(3), September (pp. 219-44).

Michalski R. (1983); 'A Theory and Methodology of Inductive Learning'; Artificial Intelligence, vol. 20 (pp. 111-61).

Minsky M. and S. Papert (1969); 'Perceptrons: An Introduction to Computational Geometry'; MIT Press, Cambridge, Massachusetts.

Mitchell T. (1997); 'Machine Learning', McGraw-Hill, New York, New York.

Newell J. and P. Rosenbloom (1980); 'Mechanisms of Skill Acquisition and the Law of Practice'; Technical Report, September, Computer Science Department , School of Computer Science, Carnegie Mellon University,

Pittsburg, Pennsylvania.

Pearl J. (1988); 'Probabilistic Reasoning in Intelligence Systems'; Morgan Kaufmann, San Mateo, California.

Rosenblatt F. (1958); 'The Perceptron: A probabilistic model for information storage and organization in the brain'; Psychological Review, 65(6) (pp. 386-408).

Rosenbloom P. (2006); 'A Cognitive Odyssey: From the Power Law of Practice to a General Learning Mechanism and Beyond'; Tutorials in Quantitative Methods for Psychology, vol. 2.

Rumelhart D., G. Hinton and R. Williams (1986); 'Learning representations by backpropagating errors'; Nature, 323, October (pp. 533-6).

Russell S. and P. Norvig (2016); 'Artificial Intelligence: A Modern Approach'; 3rd Edition, Pearson Education Limited, Edinburgh Gate, Harlow, Essex, UK.

Saxena S., R. Raperya and N. Malik (2017); 'Machine Learning Using Chunking'; International Journal of Advanced Research in Science and Engineering, 6(2), February (pp. 285-292).

Silver N. (2012); 'The Signal and the Noise'; Penguin Press, New York, New York.

Sutton R. and A. Barto (1998); 'Reinforcement Learning: An Introduction'; MIT Press, Cambridge, Massachusetts.

Thorndike E. (1898); 'Animal Intelligence: an Experimental Study of the Associative Processes in Animals'; Psychological Monographs #8.

Valiant L. (1984); 'A Theory of the Learnable'; Communications of the ACM, 27(11), November (pp. 1134-42).

Valiant L. (2013); 'Probably Approximately Correct'; Basic Books, New York, New York.

Walpert D. (1996); 'The Lack of A Priori Distinctions Between Learning Algorithms'; *Neural Computation*, 8(7), October (pp. 1341-90).