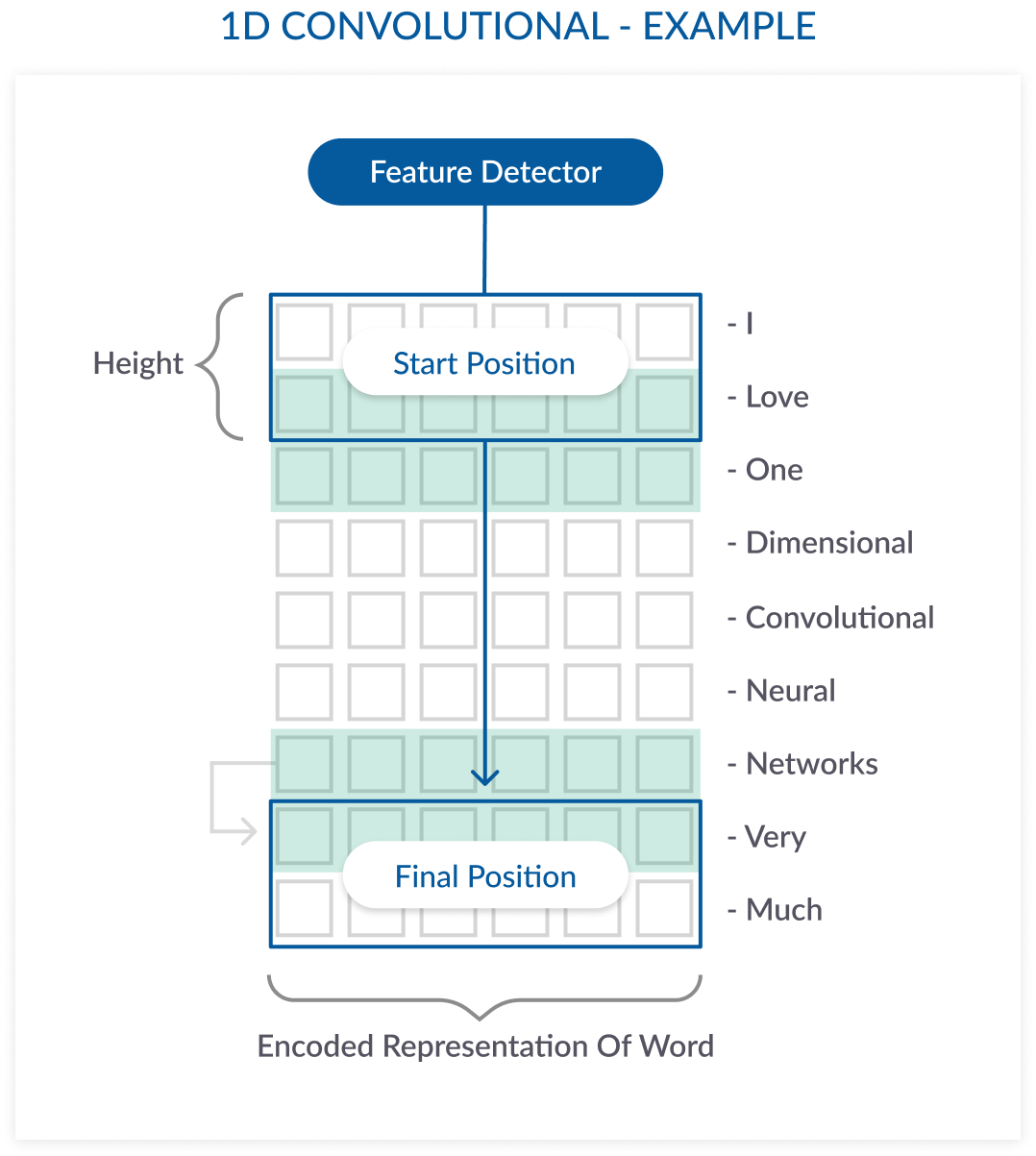
# **1D Convolutional Neural Networks**

**Convolutional Neural Network (CNN)** models were developed for image classification, in which the model accepts a two-dimensional input representing an image’s pixels and color channels, in a process called feature learning.

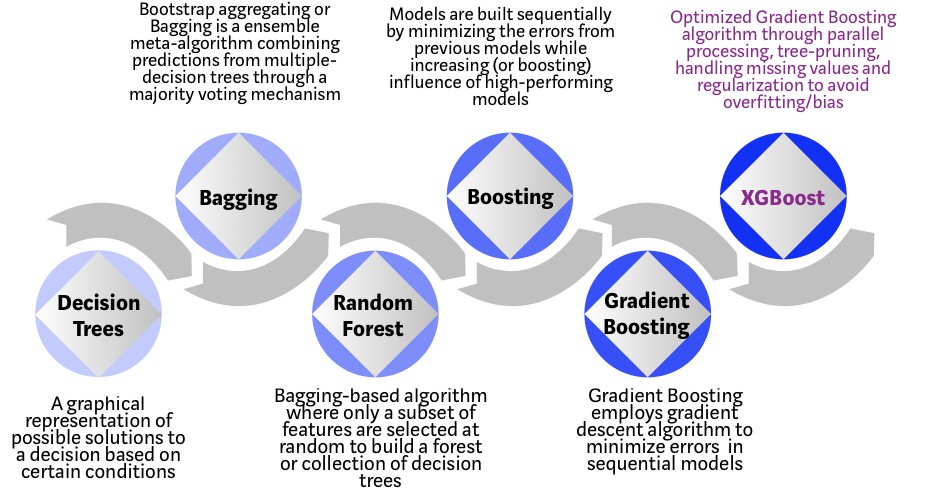
This same process can be applied to one-dimensional sequences of data. The model extracts features from sequences data and maps the internal features of the sequence. A 1D CNN is very effective for deriving features from a fixed-length segment of the overall dataset, where it is not so important where the feature is located in the segment.



In this natural language processing (NLP) example, a sentence is made up of 9 words. Each word is a vector that represents a word. The filter covers at least one word; a height parameter specifies how many words the filter should consider at once. In this example the height is 2, meaning the filter moves 8 times to fully scan the data.

**XGBOOST**

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now. Please see the chart below for the evolution of tree-based algorithms over the years.

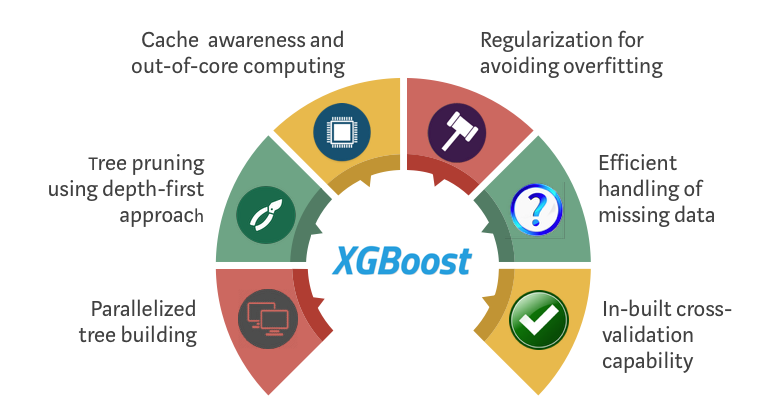


XGBoost algorithm was developed as a research project at the University of Washington. Tianqi Chen and Carlos Guestrin presented their paper at SIGKDD Conference in 2016 and caught the Machine Learning world by fire. Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications. As a result, there is a strong community of data scientists contributing to the XGBoost open source projects with ~350 contributors and ~3,600 commits on GitHub. The algorithm differentiates itself in the following ways:

* A wide range of applications: Can be used to solve regression, classification, ranking, and user-defined prediction problems.
* Portability: Runs smoothly on Windows, Linux, and OS X.
* Languages: Supports all major programming languages including C++, Python, R, Java, Scala, and Julia.

Cloud Integration: Supports AWS, Azure, and Yarn clusters and works well with Flink, Spark, and other ecosystems.

**Why does XGBoost perform so well?**

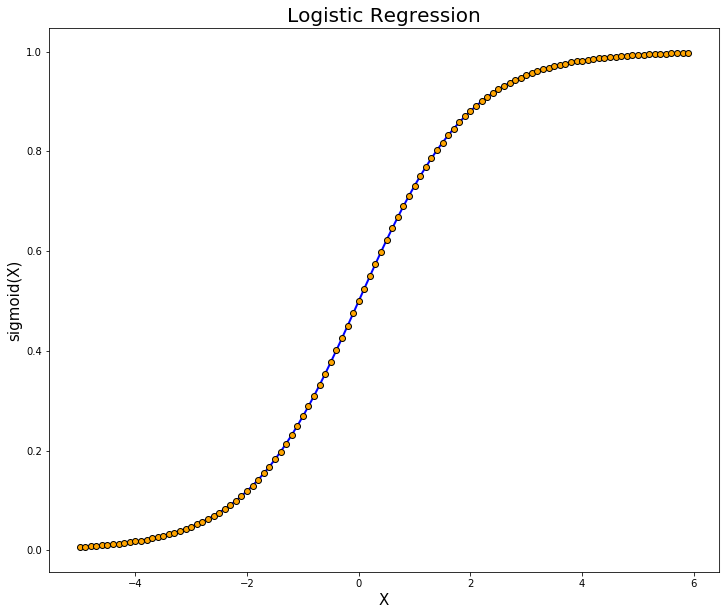
XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

# Logistic Regression

In the Machine Learning, Logistic Regression is a kind of parametric classification model, despite having the word ‘regression’ in its name.

This means that logistic regression models are models that have a certain fixed number of parameters that depend on the number of input features, and they output categorical prediction, like for example if a plant belongs to a certain species or not.

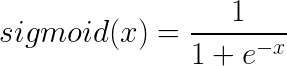
In Logistic Regression, we don’t directly fit a straight line to our data like in linear regression. Instead, we fit a S shaped curve, called Sigmoid, to our observations.



Logistic Regression models are classification models; specifically binary classification models (they can only be used to distinguish between 2 different categories — like if a person is obese or not given its weight, or if a house is big or small given its size). This means that our data has two kinds of observations (Category 1 and Category 2 observations) like we can observe in the figure.

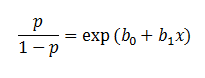
Secondly, as we can see, the Y-axis goes from 0 to 1. This is because the sigmoid function always takes as maximum and minimum these two values, and this fits very well our goal of classifying samples in two different categories. By computing the sigmoid function of X (that is a weighted sum of the input features, just like in Linear Regression), we get a probability (between 0 and 1 obviously) of an observation belonging to one of the two categories.

The formula for the sigmoid function is the following:

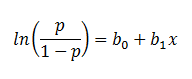




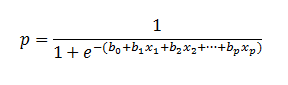
In the logistic regression the constant (b0) moves the curve left and right and the slope (b1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.



Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (b1) is the amount the logit (log-odds) changes with a one unit change in x.



Logistic regression can handle any number of numerical and/or categorical variables.



**Support Vector Machines**

Support vector machines are a core machine learning tech-nology. They have strong theoretical foundations and excel-lent empirical successes. They have been applied to taskssuch as handwritten digit recognition [35], object recogni-tion [25], and text classification [14]

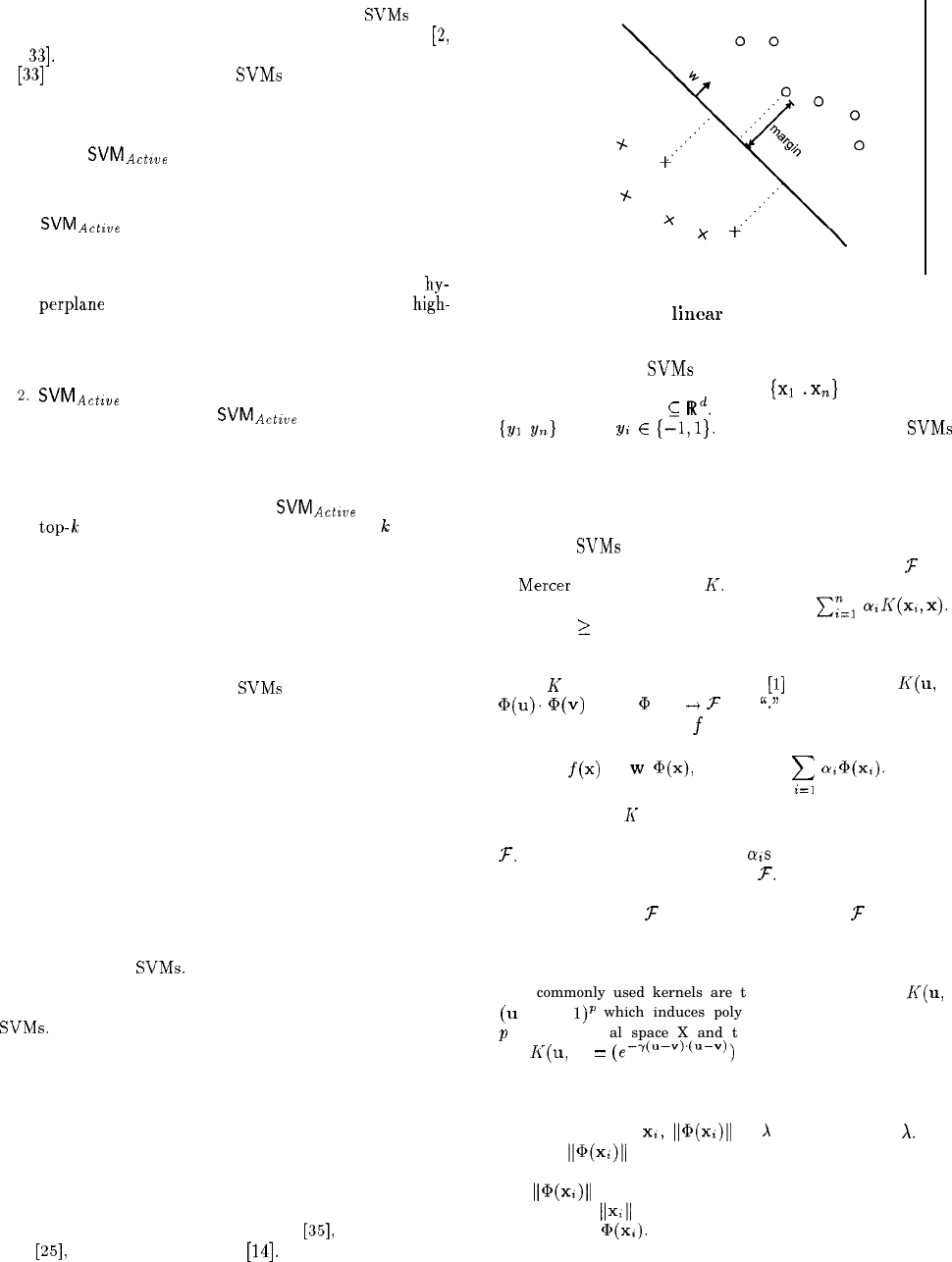


Figure 1: A simple linear Support Vector Machine

We shall consider SVMs in the binary classification setting. We are given training data (x1 . xn} that are vectors in some space .We are also given their labels{y1 yn} where . In their simplest form, SVMs are hyperplanes that separate the training data by a maximal margin (see Fig. 1). All vectors lying on one side of the hyper plane are labeled as -1, and all vectors lying on the other side are labeled as 1. The training instances that lie closest to the hyper plane are called support vectors. More generally, SVMs allow one to project the original training data in space X to a higher dimensional feature space F via **Mercer kernel operator K**.

[35] V. Vapnik. Statistical Learning Theory. Wiley, 1998.

[25 ] C. Papageorgiou, M. Oren, and T. Poggio. A general

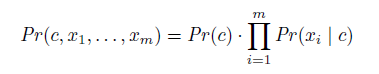
framework for object detection. In Proceedings of the

International ConJerence on Computer Vision, 1998.

[14] T. Joachims. Text categorization with support vector machines. In Proceedings of the European C~njerence on Machine Learning. Springer-Verlrtg, 1998.

**NAIVE BAYES CLASSIFIER**

A naïve Bayes classifier corresponds to a Bayesian network, as in Eq (1). Here, a single class variable C and m attribute variables Xi (for simplicity of exposition, attributes are discrete). Let c denote a class label and x i denote a value of an attribute Xi . A naïve Bayes induces a distribution:

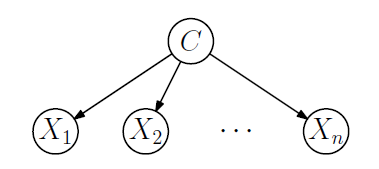


**Eq (1)**

Where we have a class prior Pr(C) and conditional distributions Pr(Xi|C). We can estimate these parameters from (labeled) data, using maximum likelihood or MAP estimation. Once we have learned a naïve Bayes classifier from data, we can label new instances by selecting the class label c\* that has maximum posterior probability given observation sx 1 ,...,xm . **[1]** Select:



**Eq (2)**



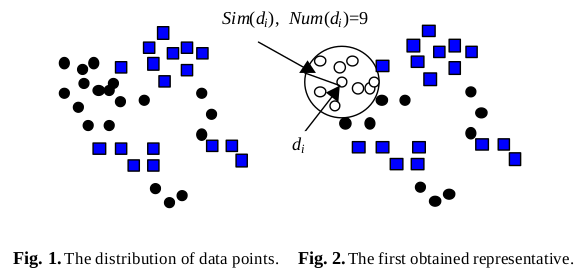
**Figure 1. A Naïve Bayes Classifier**

**[1] Pouria Kaviani and Mrs. Sunita Dhotre. Short Survey on Naive Bayes Algorithm, 2017.**

**K-Nearest-Neighbours (KNN)**

The k-Nearest-Neighbours (kNN) is a non-parametric classification method, which is simple but effective in many cases [1]. For a data record t to be classified, its k nearest neighbours are retrieved, and this forms a neighbourhood of t. Majority voting among the data records in the neighbourhood is usually used to decide the classification for t with or without consideration of distance-based weighting. However, to apply kNN we need to choose an appropriate value for k, and the success of classification is very much dependent on this value. In a sense, the kNN method is biased by k. There are many ways of choosing the k value, but a simple one is to run the algorithm many times with different k values and choose the one with the best performance.

In order for kNN to be less dependent on the choice of k, Wang [2] proposed to look at multiple sets of nearest neighbours rather than just one set of k-nearest neighbours. The proposed formalism is based on contextual probability, and the idea is to aggregate the support of multiple sets of nearest neighbours for various classes to give a more reliable support value, which better reveals the true class of t. However, in its basic form the method is relatively slow, which needs O(n2 ) to classify a new instance, though it is indeed less dependent on k and is able to achieve classification performance close to that for the best k.



1. D. Hand, H. Mannila, P. Smyth.: Principles of Data Mining. The MIT Press. (2001)

2. H. Wang.: Nearest Neighbours without k: A Classification Formalism based on

Probability, technical report, Faculty of Informatics, University of Ulster,

N.Ireland, UK (2002)

**Stochastic gradient descent**

The word ‘stochastic‘ means a system or a process that is linked with a random probability. Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called “batch” which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy or less random manner, but the problem arises when our datasets get really huge.

There are three variants of gradient descent, which differ in how much data we use to compute the

gradient of the objective function. Depending on the amount of data, we make a trade-off between

the accuracy of the parameter update and the time it takes to perform an update.

Stochastic gradient descent (SGD) performs a parameter update for each training example

*x(i)* and label *y(i)* :

θ = θ − η · ∇ θ J(θ; x(i) ; y (i) ) (1)

Batch gradient descent performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update. SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn

online. SGD performs frequent updates with a high variance that cause the objective function to

fluctuate heavily.

While batch gradient descent converges to the minimum of the basin the parameters are placed in,

SGD’s fluctuation, on the one hand, enables it to jump to new and potentially better local minima.

On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep

overshooting. However, it has been shown that when we slowly decrease the learning rate, SGD

shows the same convergence behaviour as batch gradient descent, almost certainly converging to a

local or the global minimum for non-convex and convex optimization respectively.

Reference:

[1] Sebastian Ruder Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin. An overview of gradient descent optimization algorithms, 2017.

**Decision Tree**

A normal tree includes root, branches and leaves. The same structure is followed in Decision Tree. It contains root node, branches, and leaf nodes. Testing an attribute is on every internal node, the outcome of the test is on branch and class label as a result is on leaf node [3, 4]. A root node is parent of all nodes and as the name suggests it is the topmost node in Tree. A decision tree is a tree where each node shows a feature (attribute), each link (branch) shows a decision (rule) and each leaf shows an outcome (categorical or continues value) [4]. As decision trees mimic the human level thinking so it’s so simple to grab the data and make some good interpretations. The whole idea is to create a tree like this for the entire data and process a single outcome at every leaf.

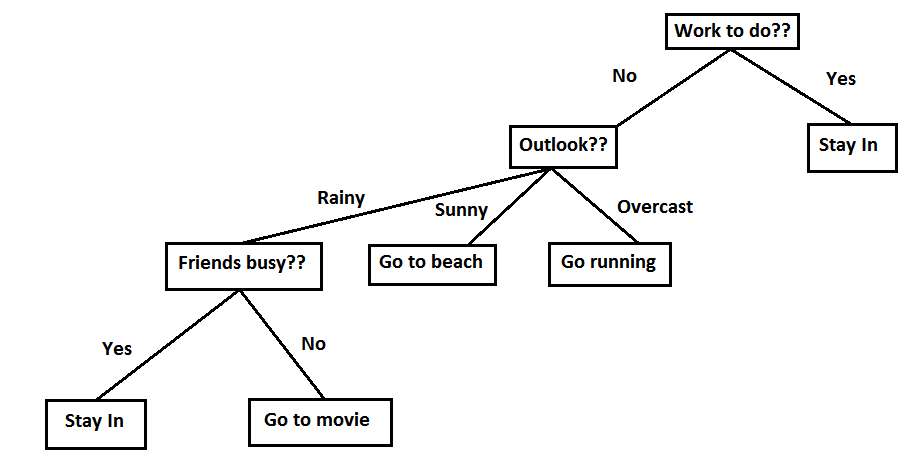


Fig. 2: Example of Decision Tree on what to do when

different situations occur in weather.

When data does not offer benefits while splitting, it directly stops the execution. Try to find one test at a time rather than optimize the whole tree together.

[3] Gershman A, Meisels A, Lüke KH, Rokach L, Schclar A, Sturm A. A Decision Tree Based Recommender System. InIICS 2010 Jun 3 (pp. 170-179).

[4] Jadhav SD, Channe HP. Efficient recommendation system using decision tree classifier and collaborative filtering. Int. Res. J. Eng. Technol. 2016;3:2113-8.