Machine Learning Algorithms I

WE'LL COVER THE FOLLOWING

Introduction

In this lesson, we are going to learn about the most popular machine learning algorithms. Note that we are not going to do a technical deep-dive as it would be out of our scope. The goal is to cover details sufficiently enough so that you can navigate through them when needed. ***The key is to know about the different possibilities so that you can then go deeper on a need's basis***.

In this algorithm tour we are going to learn about:

1. Linear Regression
2. Logistic Regression
3. Decision Trees
4. Naive Bayes
5. Support Vector Machines, SVM
6. K-Nearest Neighbors, KNN
7. K-Means
8. Random Forest
9. Dimensionality Reduction

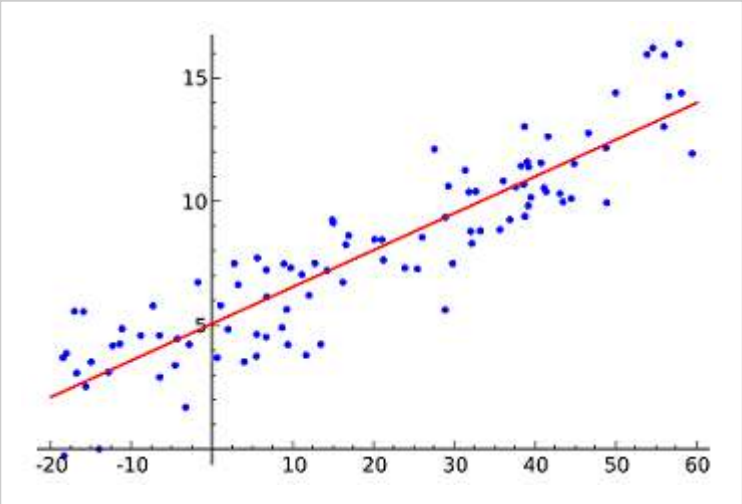
10. Artificial Neural Networks, ANN

1. Linear Regression

Linear Regression is probably *the* most popular machine learning algorithm.

Remember in high school when you had to plot data points on a graph with an *X*-axis and a *Y*-axis and then find the line of best fit? That was a very simple machine learning algorithm, linear regression. In more technical terms, linear regression attempts to represent the relationship between one or more independent variables (points on *X* axis) and a numeric outcome or dependent variable (value on *Y* axis) by fitting the equation of a line to the data:

$$Y = a * X + b$$



Example of simple linear regression, which has one independent variable (x-axis) and a dependent variable (y-axis)

For example, you might want to relate the weights (*Y*) of individuals to their heights (*X*) using linear regression. This algorithm assumes a strong linear relationship between input and output variables as we would assume that if height increases then weight also increases proportionally in a linear way.

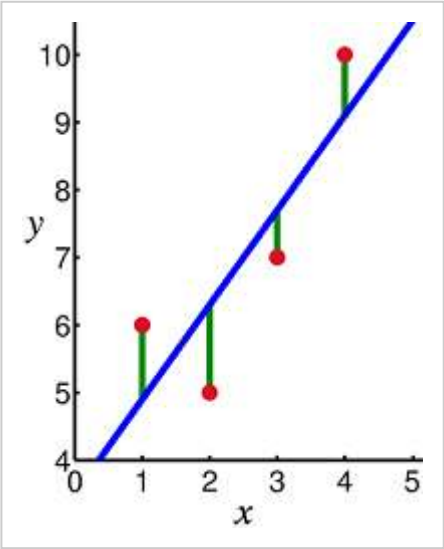
The goal here is to derive optimal values for *a* and *b* in the equation above, so that our estimated values, *Y*, can be as close as possible to their correct values. Note that we know the actual values for *Y* during the training phase because we are trying to learn our equation from the labelled examples given in the training data set.

Once our machine learning model has learned the line of best fit via linear

regression, this line can then be used to predict values for new or unseen data points.
Different techniques can be used to learn the linear regression model. The most popular methods is that of *least squares*:

Ordinary least squares: The method of least squares calculates the best-fitting line such that the vertical distances from each data point to the line are minimum. The distances in green (figure below) should be kept to a minimum so that the data points in red can be as close as possible to the blue line (line of best fit). If a point lies on the fitted line exactly then its vertical distance from the line is 0.

To be more specific, in ordinary least squares, the overall distance is the sum of the squares of the vertical distances (green lines) for all the data points. The idea is to fit a model by minimizing this squared error or distance.



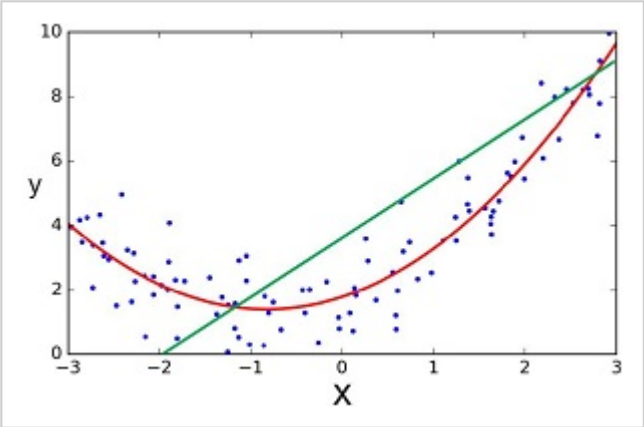
In linear regression, the observations (red) are assumed to be the result of random deviations (green) from an underlying relationship (blue) between a dependent variable (y) and an independent variable (x). While finding the line of best fit, the goal is to minimize the the distance shown in green -- red points as close as possible to the blue line.

> **Note**: ***While using libraries like Scikit-Learn, you won't have to implement any of these functions yourself. Scikit Learn provides out of the box model fitting! We will see this in action once we reach our Projects section.***

When we are dealing with only one independent variable, like in the example above, we call it *Simple Linear Regression*. When there are more than one independent variables (e.g., we want to predict weight using more variables

than just the person's height) then this type of regression is termed as *Multiple*

Linear Regression. As shown in the figure below, while finding the line of best fit, we can use a polynomial, or a curved line instead of a straight line; this is called *Polynomial Regression*.



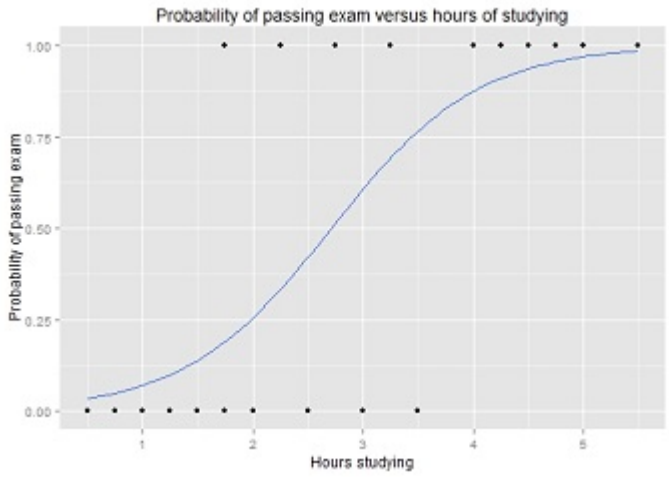
Linear Regression (green) vs Polynomial Regression (red)

2. Logistic Regression

Logistic regression has the same main idea as linear regression. The difference is that this technique is used when the output or dependent variable is binary meaning the outcome can have only two possible values. For example, let's say that we want to predict if age influences the probability of having a heart attack. In this case, our prediction is only a "yes" or "no", only two possible values.

In logistic regression, the line of best fit is not a straight line anymore. The prediction for the final output is transformed using a non-linear S-shaped function called the *logistic function*, *g()*. This logistic function maps the intermediate outcome values into an outcome variable *Y* with values ranging from 0 to 1. These 0 to 1 values can then be interpreted as the probability of occurrence of *Y*.

In the heart attack example, we have two class labels to be predicted, "yes" as 1 and "no" as 0. Say we set a threshold or cut-off value to 0.5, all instances with a probability higher than this threshold will be classified as instances belonging to class 1 , while all those having probability below the threshold will be assigned to class 0. In other words, the properties of the S-shaped logistic function make logistic regression suitable for classification tasks.



Graph of a logistic regression curve showing probability of passing an exam versus hours studying

3. Decision Trees

Decision Trees also belong to the category of supervised learning algorithms, but they can be used for solving both regression and classification tasks.

In this algorithm, the training model learns to predict values of the target variable by learning *decision rules* with a tree representation. A tree is made up of nodes corresponding to a feature or attribute. At each node we ask a question about the data based on the available features, e.g., *Is it raining or not raining?*. The left and right branches represent the possible answers. The final nodes, *leaf nodes*, correspond to a class label/predicted value. The importance for each feature is determined in a top-down approach — **the higher the node, the more important its attribute/feature**. This is easier understood with a visual representation of an example problem.

Say we want to predict whether or not we should wait for a table at a restaurant. Below is an example decision tree that decides whether or not to wait in a given situation based on different attributes:

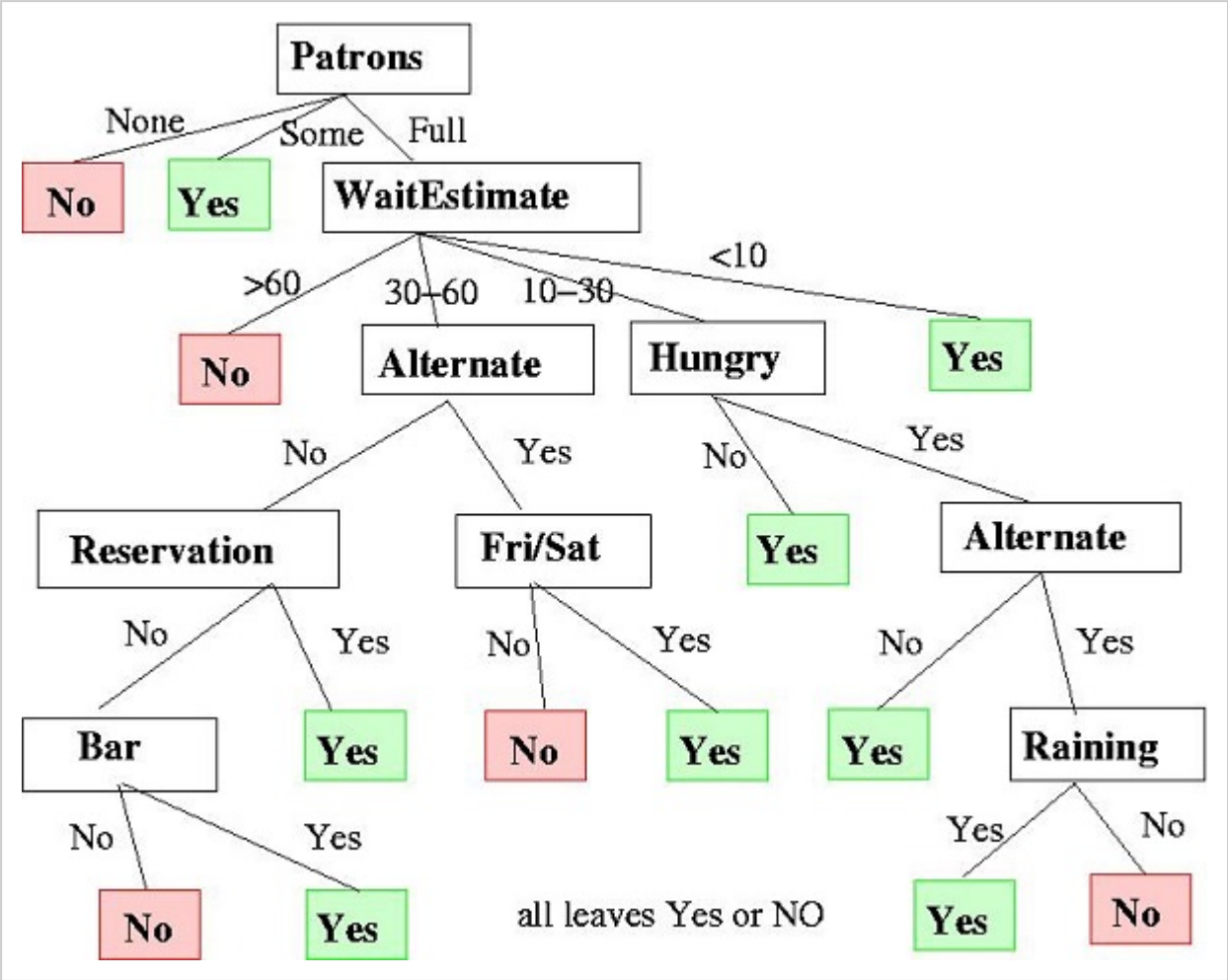


Image Credits: http://www.cs.bham.ac.uk/~mmk/Teaching/AI/

In this example, our attributes are:

- ***Alternate***: alternative restaurant nearby
- ***Bar***: bar area to wait
- ***Fri/Sat***: true on Fridays and Saturdays
- ***Hungry***: whether we are hungry
- ***Patrons***: how many people in restaurant (none, some, or full)
- ***Raining***: raining outside
- ***Wait-Estimate***: estimated waiting time (<10,10-30,30-60,>60)

Our data instances are then classified into “wait” or “leave” based on the attributes listed above. From the visual representation of the decision tree, we can see that “wait-estimate” is more important than “raining” because it is present at a relatively higher node in the tree.

4. Naive Bayes

Naive Bayes is a simple yet widely used machine learning algorithm based on the **Bayes Theorem** Remember we talked about it in the Statistics section? It is called naive because the classifier assumes that the input variables are independent of each other, quite a strong and unrealistic assumption for real data!. The Bayes theorem is given by the equation below:

$$P(c|x) = \frac{P(x|c) * P(c)}{P(x)}$$

where,
P(c | x) = probability of the event of class *c*, given the predictor variable *x*,
P(x | c) = probability of *x* given *c*,
P(c) = probability of the class,
P(x) = probability of the predictor.

To put it simply, the model is composed of two types of probabilities:

- The probability of each class;
- The conditional probability for each class given each value of *x*

Say we have a training data set with weather conditions, *x*, and the corresponding target variable “Played”, *c*. We can use this to obtain the probability of “Players will play if it is rainy”, *P(c | x)*. Note that even if the answer is a numerical value ranging from 0 to 1, this is an example of a classification problem — we can use the probabilities to reach a “yes/no” outcome.

Naive Bayes classifiers are actually a popular statistical technique of spam e-mail filtering. It works by correlating the use of tokens typically words, with spam and non-spam e-mails and then using Bayes’ theorem to calculate a probability that an email is spam or not.

5. Support Vector Machine (SVM)

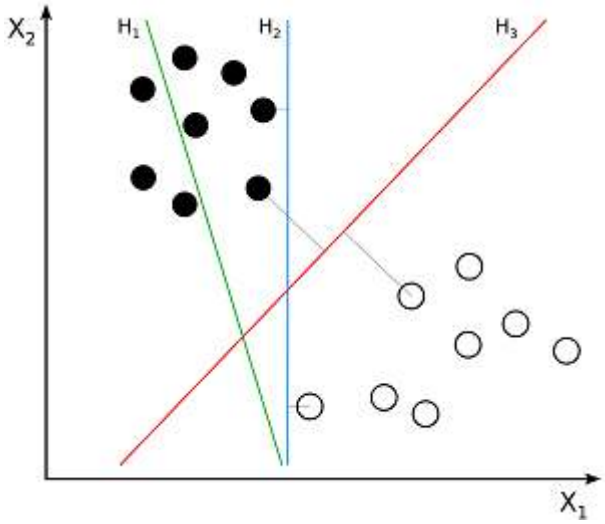
Support Vector Machines is a supervised algorithm used mainly for classification problems. In this algorithm, we plot each data item as a point in n-dimensional space, where *n* is the number of input features. For example, with two input variables, we would have a two-dimensional space. Based on these transformations, SVM finds an optimal boundary, called a *hyperplane*, that best separates the possible outputs by their class label. In a two-

dimensional space, this hyperplane can be visualized as a line although not

necessarily a straight line. The task of the SVM algorithm is to find the coefficients that provide the best separation of classes by this hyperplane.

The distance between the hyperplane and the closest class point is called the *margin*. The **optimal hyperplane** is one that has the largest margin that classifies points in such a way that **the distance between the closest data point from both classes is maximum**.

In simple words, SVM tries to draw two lines between the data points with the largest margin between them. Say we are given a plot of two classes, black and white dots, on a graph as shown in the figure below. The job of the SVM classifier would then be to decide the best line that can separate the black dots from the white dots, as shown in the figure below:



H1 does not separate the two classes. H2 does, but only with a small margin. H3 separates them with the maximal margin.