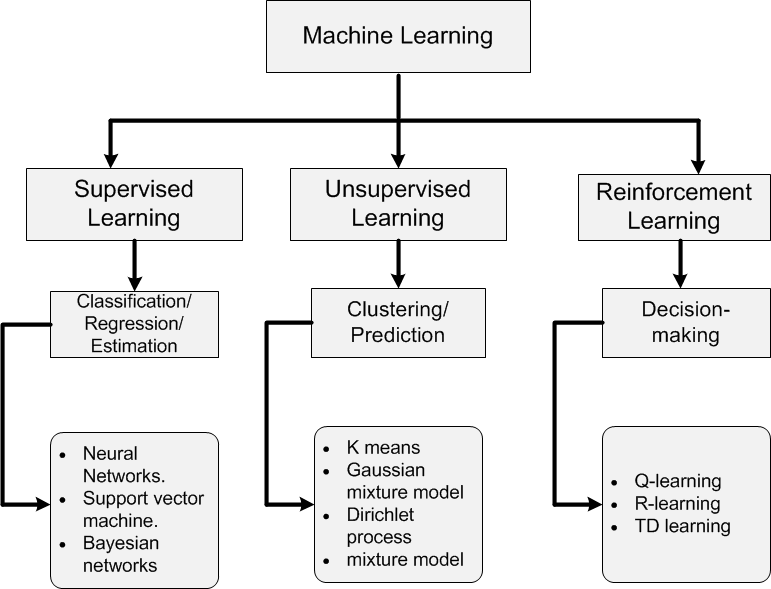
Machine learning and its models

**Machine learning** (ML) is the scientific study of algorithms and statistical models that computer systems use to progressively improve their performance on a specific task. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

Algorithms used in machine learning fall roughly into three categories: supervised, unsupervised, and reinforcement learning. Supervised learning involves feedback to indicate when a prediction is right or wrong, whereas unsupervised learning involves no response: The algorithm simply tries to categorize data based on its hidden structure. Reinforcement learning is similar to supervised learning in that it receives feedback, but it's not necessarily for each input or state. This tutorial explores the ideas behind these learning models and some key algorithms used for each.

Machine-learning algorithms continue to grow and evolve. In most cases, however, algorithms tend to settle into one of three models for learning. The models exist to adjust automatically in some way to improve their operation or behavior.



In supervised learning, a data set includes its desired outputs (or labels) such that a function can calculate an error for a given prediction. The supervision comes when a prediction is made and an error produced (actual vs. desired) to alter the function and learn the mapping.

In unsupervised learning, a data set doesn't include a desired output; therefore, there's no way to supervise the function. Instead, the function attempts to segment the data set into "classes" so that each class contains a portion of the data set with common features.

Finally, in reinforcement learning, the algorithm attempts to learn actions for a given set of states that lead to a goal state. An error is provided not after each example (as is the case for supervised learning) but instead on receipt of a reinforcement signal (such as reaching the goal state). This behavior is similar to human learning, where feedback isn't necessarily provided for all actions but when a reward is warranted.

Linear Regression

Regression attempts to predict one dependent variable (usually denoted by Y) and a series of other changing variables (known as independent variables, usually denoted by X).

The objective of a linear regression model is to find a relationship between one or more features (independent variables) and a continuous target variable (dependent variable). When there is only feature it is called Uni-variate Linear Regression or Simple Linear Regression and if there are multiple features, it is called Multiple Linear Regression or Multivariate Linear Regression.

Hypothesis of Linear Regression

The linear regression model can be represented by the following equation:

https://cdn-images-1.medium.com/max/800/1*A32yDrkQKIU0Z-Nf7JD9Pg.png

* is the predicted value
* is the bias term.
* are the model parameters
* are the feature values.

The above hypothesis can also be represented by:

https://cdn-images-1.medium.com/max/800/1*7pjcby7gESbxk54njwCUZg.png

Where,

* is the model’s parameter vector including the bias term
* is the feature vector with

#### ****Training a Linear Regression Model****

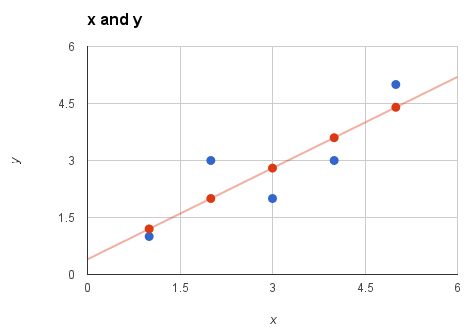
#### Training of the model here means to find the parameters so that the model best fits the data.

#### Let us consider a dataset (scatter plot is given below) with x (dependent variable) and y ( independent variable ). It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

#### C:\Users\subham\Desktop\55.PNG

#### Determining best fit line

#### The line for which the the error between the predicted values and the observed values is minimum is called the best fit line or the ****regression****line. These errors are also called as residuals. The residuals can be visualized by the vertical lines from the observed data value to the regression line.



**Cost function**

To define and measure the error of our model we define the cost function as the sum of the squares of the residuals. The cost function is denoted by :

https://cdn-images-1.medium.com/max/800/1*cUnU05zgo8BEzjm22oszLw.png

Where, the hypothesis function *h(x)*is denoted by :

https://cdn-images-1.medium.com/max/800/1*U918Cc_opofphkgbP9Kkpg.png

and *m* is the total number of training examples in our data-set.

To get the error in terms of percentage we will be using MAPE cost function. The **mean absolute percentage error** (**MAPE**), also known as **mean absolute percentage deviation** (**MAPD**), is defined by the formula:

{\displaystyle {\mbox{M}}={\frac {100\%}{n}}\sum \_{t=1}^{n}\left|{\frac {A\_{t}-F\_{t}}{A\_{t}}}\right|,}

|  |  |
| --- | --- |
| \[ \mathrm{MAPE}=\frac{100}{N}\times \sum_{i=1}^N \left | \frac{x_i - \hat x_i}{x_i} \right | \] |  |

Where:

* is the actual value.
* is the predicted value.
* $ N $ is the number of fitted data points

After finding the cost, now our objective is to find the model parameters so that the cost function is minimum. There are several techniques to minimize the cost. We will use Gradient Descent optimization algorithm to do this.

**Gradient Descent**

Gradient descent is a generic optimization algorithm used in many machine learning algorithms. It iteratively tweaks the parameters of the model in order to minimize the cost function. The steps of gradient descent is outlined below.

1. We first initialize the model parameters with some random values. This is also called as ***random initialization***.
2. Now we need to measure how the cost function changes with change in it’s parameters. Therefore we compute the partial derivatives of the cost function w.r.t to the parameters.

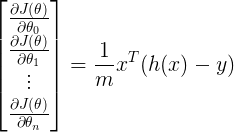
https://cdn-images-1.medium.com/max/800/1*1DbFTHQGx1yW_bygxQjDCw.png

https://cdn-images-1.medium.com/max/800/1*CuDapkX9GEW7kTWd5g6uoA.png

Similarly, the partial derivative of the cost function w.r.t to any parameter can be denoted by

https://cdn-images-1.medium.com/max/800/1*lM1cKDM6kxfIhPWVRf2n_g.png

We can compute the partial derivatives for all parameters at once using



Where, *h(x)*is given by:

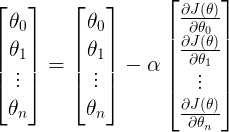
https://cdn-images-1.medium.com/max/800/1*U918Cc_opofphkgbP9Kkpg.png

3. After computing the derivative we update the parameters as given below:

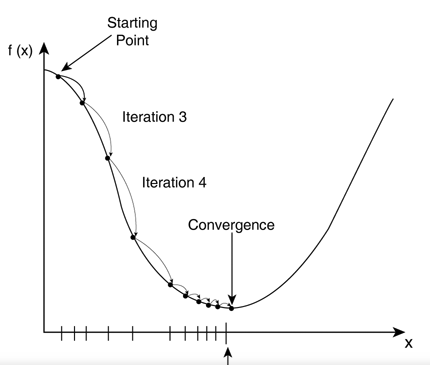
https://cdn-images-1.medium.com/max/800/1*eCIJ2D7iEzr16XjNIIAX-Q.png

https://cdn-images-1.medium.com/max/800/1*Gv_g5WHAGZUIeZLBmYncLw.png

Where,  ***α***is the ***learning parameter***. We can update all the parameters at once using,



We repeat the steps 2,3 until the cost function converges to the minimum value.If the value of *α* is too small, the cost function takes larger time to converge. If *α* is too large, gradient descent may overshoot the minimum and may finally fail to converge.



In the above plott, f(x) represents cost when we have only one parameter x.

**Feature Scaling**

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations, this is a problem.

If left alone, these algorithms only take in the magnitude of features neglecting the units. The results would vary greatly between different units, 5 km and 5000 meters. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes.

**Feature scaling** is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

There are four common methods to perform Feature Scaling:

**Standardization:** The result of standardization (or **Z-score normalization**) is that the features will be rescaled so that they’ll have the properties of a standard normal distribution with μ=0 and σ=1.

Where, μ is the mean (average) and σ is the standard deviation from the mean; standard scores (also called ***z*** scores) of the samples are calculated as follows:

Standardizing the features so that they are centered around 0 with a standard deviation of 1 is not only important if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms.

**Min-max scaling:** It isalso known as min-max normalization, is the simplest method and consists in rescaling the range of features to scale the range in [0, 1] or [−1, 1]. Selecting the target range depends on the nature of the data. The general formula is given as:

{\displaystyle x'={\frac {x-{\text{min}}(x)}{{\text{max}}(x)-{\text{min}}(x)}}}

Where,  {\displaystyle x} is an original value, {\displaystyle x'} is the normalized value.

**Mean normalization:**

This distribution will have values between **-1 and 1** with**μ=0**.

### **Scaling to unit length:** Another option that is widely used in machine-learning is to scale the components of a feature vector such that the complete vector has length one. This usually means dividing each component by the Euclidean length of the vector:

**Standardization**and **Mean Normalization** can be used for algorithms that assumes zero centric data like **Principal Component Analysis (PCA). Whereas, Min-Max Scaling** and **Unit Vector** techniques produces values of range [0, 1]. When dealing with features with hard boundaries this is quite useful. For example, when dealing with image data, the colors can range from only 0 to 255.