Learning Outcome

# Residuals in Regression

When you perform simple linear regression (or any other type of regression analysis), you get a line of best fit. The data points usually don’t fall exactly on this regression equation line; they are scattered around. A residual is the vertical distance between a data point and the regression line. Each data point has one residual. They are:

* Positive if they are above the regression line,
* Negative if they are below the regression line,
* Zero if the regression line actually passes through the point,

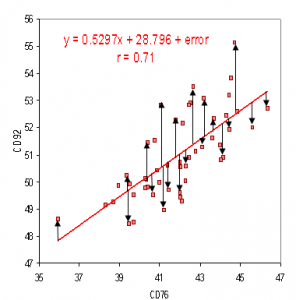


Image: Residuals on a scatter plot.

Reference: nws.noaa.gov

As residuals are the difference between any data point and the regression line, they are sometimes called “errors.” Error in this context doesn’t mean that there’s something wrong with the analysis; it just means that there is some unexplained difference. In other words, the residual is the error that isn’t explained by the regression line.

The residual(e) can also be expressed with an equation. The e is the difference between the predicted value (ŷ) and the observed value. The scatter plot is a set of data points that are observed, while the regression line is the prediction.

**Residual = Observed value – predicted value**

**e = y – ŷ**

## The Sum and Mean of Residuals

The sum of the residuals always equals zero (assuming that your line is actually the line of “best fit.” The mean of residuals is also equal to zero, as the mean = the sum of the residuals / the number of items. The sum is zero, so 0/n will always equal zero.

# Polynomial Features

Polynomial features are those features created by raising existing features to an exponent.

For example, if a dataset had one input feature X, then a polynomial feature would be the addition of a new feature (column) where values were calculated by squaring the values in X, e.g. X2. This process can be repeated for each input variable in the dataset, creating a transformed version of each.

As such, polynomial features are a type of feature engineering, e.g. the creation of new input features based on the existing features.

The “degree” of the polynomial is used to control the number of features added, e.g. a degree of 3 will add two new variables for each input variable. Typically a small degree is used such as 2 or 3.

# Classification techniques

## What Is Classification?

Classification is the process of recognizing, understanding, and grouping ideas and objects into preset categories or “sub-populations.” Using pre-categorized training datasets, machine learning programs use a variety of algorithms to classify future datasets into categories.

Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories. One of the most common uses of classification is filtering emails into “spam” or “non-spam.”

In short, classification is a form of “pattern recognition,” with classification algorithms applied to the training data to find the same pattern (similar words or sentiments, number sequences, etc.) in future sets of data.

Popular Classification Algorithms:

* Logistic Regression
* Naive Bayes
* K-Nearest Neighbors
* Decision Tree
* Support Vector Machines

### Logistic Regression

Logistic regression is a calculation used to predict a binary outcome: either something happens, or does not. This can be exhibited as Yes/No, Pass/Fail, Alive/Dead, etc.

Independent variables are analyzed to determine the binary outcome with the results falling into one of two categories. The independent variables can be categorical or numeric, but the dependent variable is always categorical. Written like this:

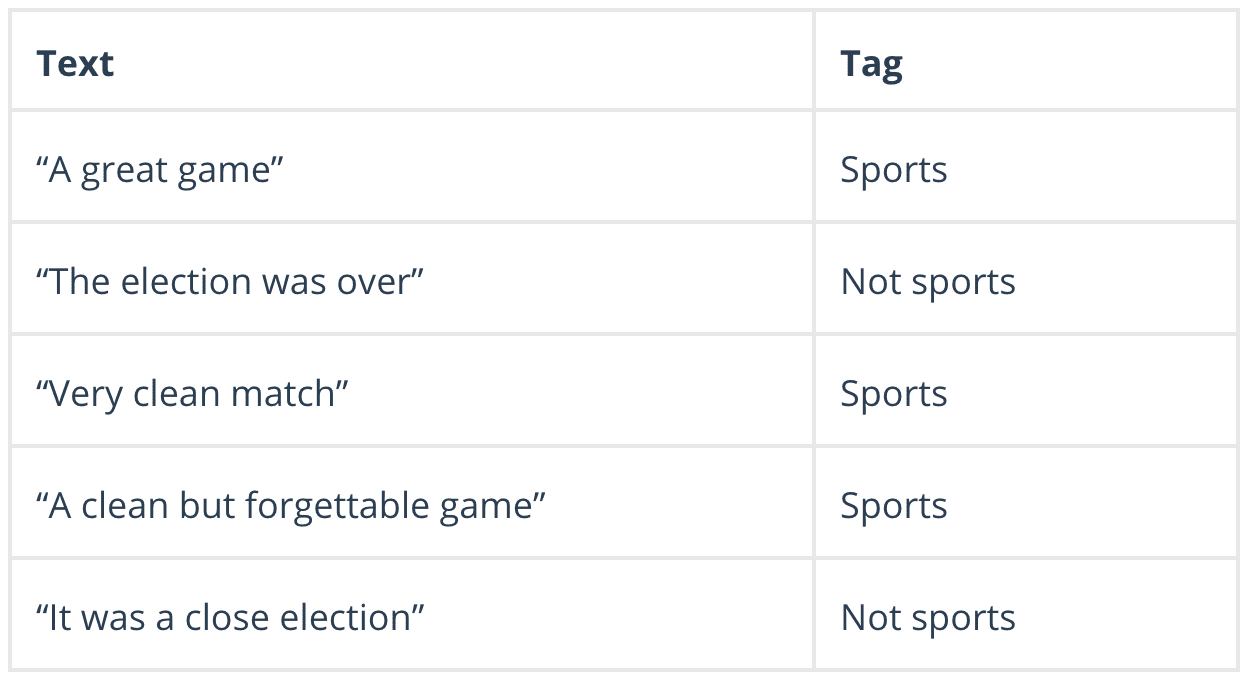
**P(Y=1|X) or P(Y=0|X)**

It calculates the probability of dependent variable Y, given independent variable X.

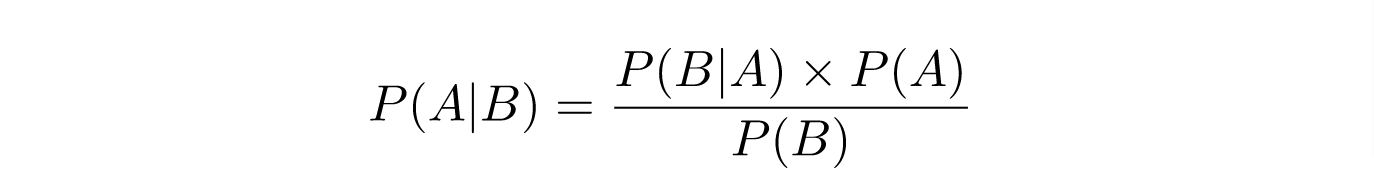
This can be used to calculate the probability of a word having a positive or negative connotation (0, 1, or on a scale between). Or it can be used to determine the object contained in a photo (tree, flower, grass, etc.), with each object given a probability between 0 and 1.

### Naive Bayes

Naive Bayes calculates the possibility of whether a data point belongs within a certain category or does not. In text analysis, it can be used to categorize words or phrases as belonging to a preset “tag” (classification) or not. For example:



To decide whether or not a phrase should be tagged as “sports,” you need to calculate:



Naive Bayes formula.

Or… the probability of A, if B is true, is equal to the probability of B, if A is true, times the probability of A being true, divided by the probability of B being true.

## K-nearest Neighbors

K-nearest neighbors (k-NN) is a pattern recognition algorithm that uses training datasets to find the k closest relatives.

When k-NN is used in classification, you calculate to place data within the category of its nearest neighbor. If k = 1, then it would be placed in the class nearest 1. K is classified by a plurality poll of its neighbors.

## Decision Tree

A decision tree is a supervised learning algorithm that is perfect for classification problems, as it’s able to order classes on a precise level. It works like a flow chart, separating data points into two similar categories at a time from the “tree trunk” to “branches,” to “leaves,” where the categories become more finitely similar. This creates categories within categories, allowing for organic classification with limited human supervision.

To continue with the sports example, this is how the decision tree works:

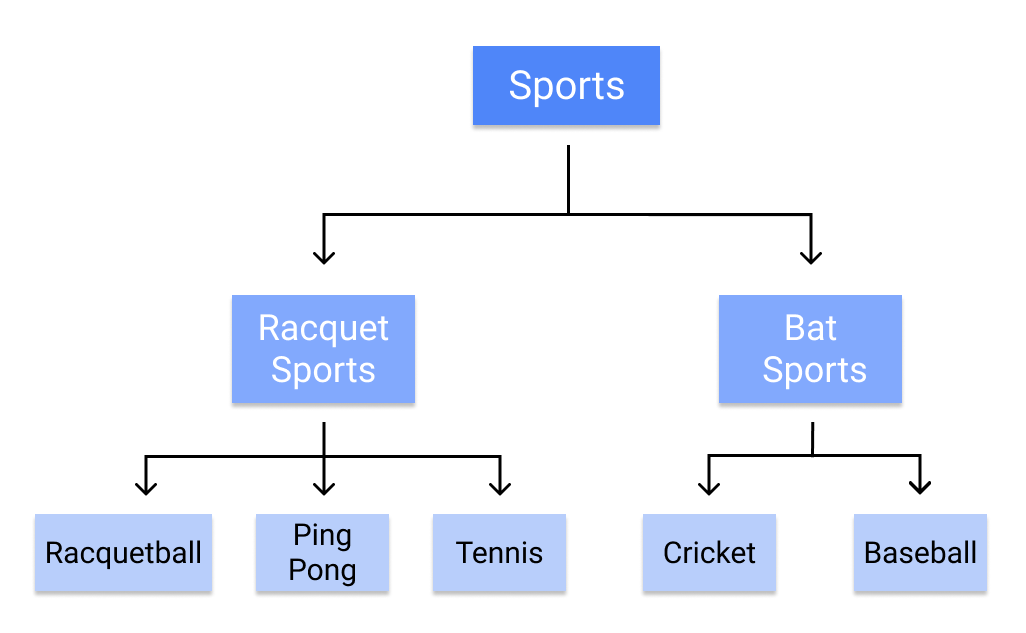


Image: Working of a Decision Tree.

Reference: nws.noaa.gov

### Random Forest

The random forest algorithm is an expansion of decision tree, in that you first construct a multitude of decision trees with training data, then fit your new data within one of the trees as a “random forest.”

It, essentially, averages your data to connect it to the nearest tree on the data scale. Random forest models are helpful as they remedy for the decision tree’s problem of “forcing” data points within a category unnecessarily.

### Support Vector Machines

A support vector machine (SVM) uses algorithms to train and classify data within degrees of polarity, taking it to a degree beyond X/Y prediction.

For a simple visual explanation, we’ll use two tags: red and blue, with two data features: X and Y, then train our classifier to output an X/Y coordinate as either red or blue.

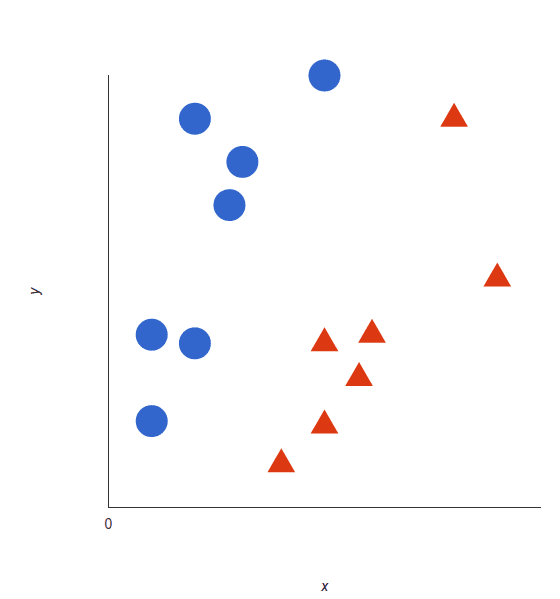


Image: Working of a Decision Tree.

Reference: <https://monkeylearn.com/blog/classification-algorithms/>

The SVM then assigns a hyperplane that best separates the tags. In two dimensions this is simply a line. Anything on one side of the line is red and anything on the other side is blue. In sentiment analysis, for example, this would be positive and negative.

In order to maximize machine learning, the best hyperplane is the one with the largest distance between each tag:

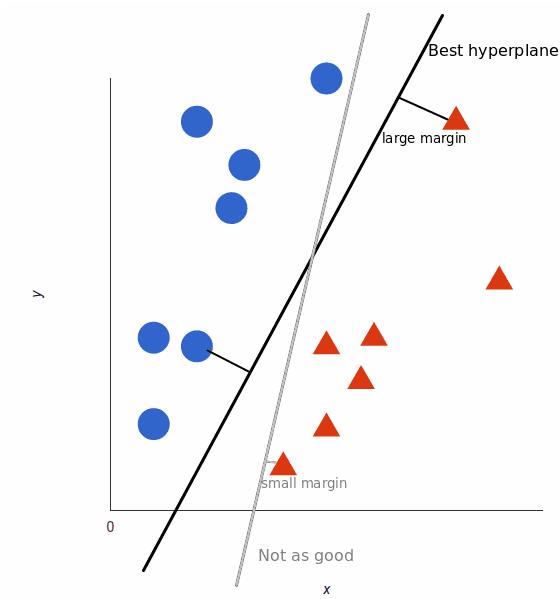


Image: SVM

Reference: <https://monkeylearn.com/blog/classification-algorithms/>

Using SVM, the more complex the data, the more accurate the predictor will become.

# Types of distance metrics

Distance measures play an important role in machine learning.

They provide the foundation for many popular and effective machine learning algorithms like k-nearest neighbors for supervised learning and k-means clustering for unsupervised learning.

Different distance measures must be chosen and used depending on the types of the data. As such, it is important to know how to implement and calculate a range of different popular distance measures and the intuitions for the resulting scores.

## Role of Distance Measures

A distance measure is an objective score that summarizes the relative difference between two objects in a problem domain.

Most commonly, the two objects are rows of data that describe a subject (such as a person, car, or house), or an event (such as a purchase, a claim, or a diagnosis).

Perhaps the most likely way you will encounter distance measures is when you are using a specific machine learning algorithm that uses distance measures at its core. The most famous algorithm of this type is the k-nearest neighbors algorithm, or KNN.

Following are the 4 most commonly used distance measures in machine learning:

1. Hamming Distance
2. Euclidean Distance
3. Manhattan Distance
4. Minkowski Distance

### Hamming Distance

Hamming distance calculates the distance between two binary vectors, also referred to as binary strings or bitstrings for short.

You are most likely going to encounter bitstrings when you one-hot encode categorical columns of data.

For example, if a column had the categories ‘red,’ ‘green,’ and ‘blue,’ you might one hot encode each example as a bitstring with one bit for each column.

red = [1, 0, 0]

green = [0, 1, 0]

blue = [0, 0, 1]

The distance between red and green could be calculated as the sum or the average number of bit differences between the two bitstrings. This is the Hamming distance.

### Euclidean Distance

Euclidean distance calculates the distance between two real-valued vectors.

You are most likely to use Euclidean distance when calculating the distance between two rows of data that have numerical values, such a floating point or integer values.

If columns have values with differing scales, it is common to normalize or standardize the numerical values across all columns prior to calculating the Euclidean distance. Otherwise, columns that have large values will dominate the distance measure.

Euclidean distance is calculated as the square root of the sum of the squared differences between the two vectors.

EuclideanDistance = sqrt(sum for i to N (v1[i] – v2[i])^2)

If the distance calculation is to be performed thousands or millions of times, it is common to remove the square root operation in an effort to speed up the calculation. The resulting scores will have the same relative proportions after this modification and can still be used effectively within a machine learning algorithm for finding the most similar examples.

EuclideanDistance = sum for i to N (v1[i] – v2[i])^2

### Manhattan Distance (Taxicab or City Block Distance)

The Manhattan distance, also called the Taxicab distance or the City Block distance, calculates the distance between two real-valued vectors.

It is perhaps more useful to vectors that describe objects on a uniform grid, like a chessboard or city blocks. The taxicab name for the measure refers to the intuition for what the measure calculates: the shortest path that a taxicab would take between city blocks (coordinates on the grid).

It might make sense to calculate Manhattan distance instead of Euclidean distance for two vectors in an integer feature space.

Manhattan distance is calculated as the sum of the absolute differences between the two vectors.

ManhattanDistance = sum for i to N sum |v1[i] – v2[i]

### Minkowski Distance

Minkowski distance calculates the distance between two real-valued vectors.

It is a generalization of the Euclidean and Manhattan distance measures and adds a parameter, called the “order” or “p“, that allows different distance measures to be calculated.

The Minkowski distance measure is calculated as follows:

EuclideanDistance = (sum for i to N (abs(v1[i] – v2[i]))^p)^(1/p)

Where “p” is the order parameter.

When p is set to 1, the calculation is the same as the Manhattan distance. When p is set to 2, it is the same as the Euclidean distance.

p=1: Manhattan distance.

p=2: Euclidean distance.

Intermediate values provide a controlled balance between the two measures.

It is common to use Minkowski distance when implementing a machine learning algorithm that uses distance measures as it gives control over the type of distance measure used for real-valued vectors via a hyperparameter “p” that can be tuned.

## KNN Classification

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Image: KNN Classifier

Reference: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

## Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



Image: KNN Classifier

Reference: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

## How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



Image: KNN Classifier

Reference: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

Firstly, we will choose the number of neighbors, so we will choose the k=5.

Next, we will calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Image: KNN Classifier

Reference: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



Image: KNN Classifier

Reference: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

# Gradient Descent

Gradient descent (GD) is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning(DL) to minimise a cost/loss function (e.g. in a linear regression). Due to its importance and ease of implementation, this algorithm is usually taught at the beginning of almost all machine learning courses.

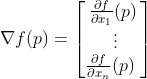
However, its use is not limited to ML/DL only, it’s being widely used also in areas like:

* control engineering (robotics, chemical, etc.)
* computer games
* mechanical engineering

## What is Gradient?

In the case of a univariate function, it is simply the first derivative at a selected point. In the case of a multivariate function, it is a vector of derivatives in each main direction (along variable axes). Because we are interested only in a slope along one axis and we don’t care about others these derivatives are called partial derivatives.

A gradient for an n-dimensional function f(x) at a given point p is defined as follows:



The upside-down triangle is a so-called nabla symbol and you read it “del”. To better understand how to calculate it let’s do a hand calculation for an exemplary 2-dimensional function below.

https://miro.medium.com/max/274/1*z-6cHPM1q_HtB4HrAXZ-FQ.gif

## How does Gradient Descent work?

Before starting the working principle of gradient descent, we should know some basic concepts to find out the slope of a line from linear regression. The equation for simple linear regression is given as:

Y=mX+c

Where 'm' represents the slope of the line, and 'c' represents the intercepts on the y-axis.

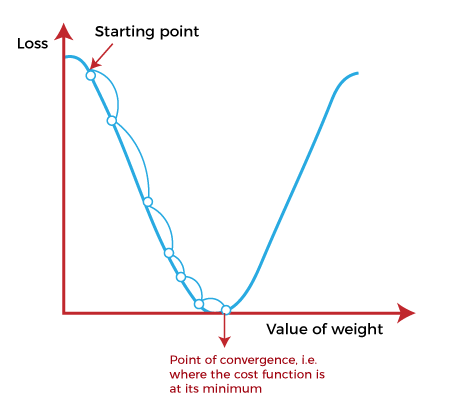


Image: Gradient Descent

Reference: <https://www.javatpoint.com/gradient-descent-in-machine-learning>

The starting point is used to evaluate the performance as it is considered just as an arbitrary point. At this starting point, we will derive the first derivative or slope and then use a tangent line to calculate the steepness of this slope. Further, this slope will inform the updates to the parameters (weights and bias).

The slope becomes steeper at the starting point or arbitrary point, but whenever new parameters are generated, then steepness gradually reduces, and at the lowest point, it approaches the lowest point, which is called a point of convergence.

The main objective of gradient descent is to minimize the cost function or the error between expected and actual. To minimize the cost function, two data points are required:

**Direction & Learning Rate**

These two factors are used to determine the partial derivative calculation of future iteration and allow it to the point of convergence or local minimum or global minimum. Let's discuss learning rate factors in brief;

### Learning Rate:

It is defined as the step size taken to reach the minimum or lowest point. This is typically a small value that is evaluated and updated based on the behavior of the cost function. If the learning rate is high, it results in larger steps but also leads to risks of overshooting the minimum. At the same time, a low learning rate shows the small step sizes, which compromises overall efficiency but gives the advantage of more precision.

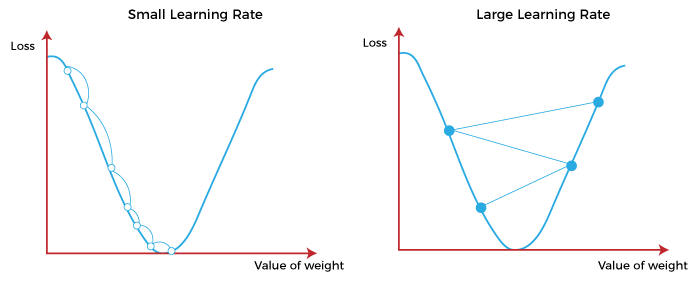


Image: Learning rate

Reference: <https://www.javatpoint.com/gradient-descent-in-machine-learning>

## Types of Gradient Descent

Based on the error in various training models, the Gradient Descent learning algorithm can be divided into :

1. Batch gradient descent,
2. stochastic gradient descent, and
3. mini-batch gradient descent.

### Batch Gradient Descent:

Batch gradient descent (BGD) is used to find the error for each point in the training set and update the model after evaluating all training examples. This procedure is known as the training epoch.

**Advantages of Batch gradient descent:**

* It produces less noise in comparison to other gradient descent.
* It produces stable gradient descent convergence.
* It is Computationally efficient as all resources are used for all training samples.

### Stochastic gradient descent

Stochastic gradient descent (SGD) is a type of gradient descent that runs one training example per iteration. Or in other words, it processes a training epoch for each example within a dataset and updates each training example's parameters one at a time. As it requires only one training example at a time, hence it is easier to store in allocated memory. However, it shows some computational efficiency losses in comparison to batch gradient systems as it shows frequent updates that require more detail and speed. Further, due to frequent updates, it is also treated as a noisy gradient. However, sometimes it can be helpful in finding the global minimum and also escaping the local minimum.

**Advantages of Stochastic gradient descent:**

In Stochastic gradient descent (SGD), learning happens on every example, and it consists of a few advantages over other gradient descent.

* It is easier to allocate in desired memory.
* It is relatively fast to compute than batch gradient descent.
* It is more efficient for large datasets.

### MiniBatch Gradient Descent:

Mini Batch gradient descent is the combination of both batch gradient descent and stochastic gradient descent. It divides the training datasets into small batch sizes then performs the updates on those batches separately. Splitting training datasets into smaller batches make a balance to maintain the computational efficiency of batch gradient descent and speed of stochastic gradient descent. Hence, we can achieve a special type of gradient descent with higher computational efficiency and less noisy gradient descent.

**Advantages of Mini Batch gradient descent:**

* It is easier to fit in allocated memory.
* It is computationally efficient.
* It produces stable gradient descent convergence.

## Gradient Descent Algorithm

Gradient Descent Algorithm iteratively calculates the next point using gradient at the current position, then scales it (by a learning rate) and subtracts obtained value from the current position (makes a step). It subtracts the value because we want to minimise the function (to maximise it would be adding). This process can be written as:

There’s an important parameter η which scales the gradient and thus controls the step size. In machine learning, it is called learning rate and have a strong influence on performance.

The smaller learning rate the longer GD converges, or may reach maximum iteration before reaching the optimum point

If learning rate is too big the algorithm may not converge to the optimal point (jump around) or even to diverge completely.

In summary, Gradient Descent method’s steps are:

* choose a starting point (initialisation)
* calculate gradient at this point
* make a scaled step in the opposite direction to the gradient (objective: minimise)
* repeat points 2 and 3 until one of the criteria is met:
* maximum number of iterations reached
* step size is smaller than the tolerance.

References

1. <https://www.statisticshowto.com/residual/>
2. <https://machinelearningmastery.com/polynomial-features-transforms-for-machine-learning/#:~:text=Polynomial%20features%20are%20those%20features,X%2C%20e.g.%20X%5E2>.
3. <https://monkeylearn.com/blog/classification-algorithms/>
4. <https://machinelearningmastery.com/distance-measures-for-machine-learning/#:~:text=Perhaps%20four%20of%20the%20most,Manhattan%20Distance>
5. <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
6. <https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21>
7. <https://www.javatpoint.com/gradient-descent-in-machine-learning>