Basics of Linear Algebra

**Matrix and Vectors**

To understand machine learning in-depth, let us first understand a few topics related to linear algebra. You do not necessarily need to understand linear algebra in-depth, to get started with machine learning. However, it will be useful in understanding the mathematics behind machine learning algorithms.

In linear algebra, data is mostly represented using matrices and vectors.

**Matrix**

Matrices (plural of matrix) are used throughout machine learning algorithms, specifically for input variables, i.e., the variables we try to understand in machine learning. A matrix of size LaTeX: m\times n is a two-dimensional array that has **m** rows and **n** columns.

**Vector**

A vector is a one-dimensional array that has only one row - called a row vector, or just one column, also called a column vector.

For example, in the below figure, X is a matrix with 3 rows and 2 columns, while Y is a simple column vector.

A number and a number

Description automatically generated with medium confidence

Now, let’s look at some operations that can be applied to vectors and matrices.

**Addition and Subtraction**

For the addition and subtraction of two vectors or two matrices, both should be of the same size. The addition and subtraction operations are performed in an "element-wise" manner between the two objects - that is, the corresponding elements from the two vectors or matrices are added to each other or subtracted from each other.

For example, the below figure shows the addition and subtraction of two column vectors, each having 3 elements.

A number of squares with numbers

Description automatically generated with medium confidence

This kind of "element-wise" addition and subtraction can similarly be performed for two or more matrices of the same size as well.

**Multiplying a matrix with a vector**

To multiply a matrix with a column (or row) vector, **the matrix must have the same number of columns (or rows) as the number of elements in the column (or row) vector.**

Let us look at an example of multiplying a matrix with a vector, where **X** is an LaTeX: n\times1 column vector and **A** is an LaTeX: m\times n matrix.

A yellow rectangular object with black text

Description automatically generated

As the number of columns in **A** is equal to the number of elements in **X**, it is possible to find **AX**(**A** multiplied by **X**).

To do multiplication, we multiply each element of row 1 of the matrix with the corresponding element of the column vector and add them, i.e., (1\*2 + 2\*1 + 3\*3) = 13. Similarly, we can apply the same operation to rows 2 and 3 of matrix **A** with the same three elements of **X**, and get the below result:

A yellow rectangular object with black text

Description automatically generated

**Multiplying a matrix with another matrix**

One way to multiply a matrix with another matrix is what's known as the **Dot Product.**

To be able to multiply two matrices and get a dot product, **the number of columns in the first matrix should equal the number of rows in the second matrix.**

The resultant matrix would have the same number of rows as the first matrix and the same number of columns as the second matrix.

For example, the below figure shows the dot product of two matrices.

1. The first element, i.e., the element at the first row and the first column of the resultant matrix, would be*: (1\*7) + (2\*8) + (3\*9) = 50*
2. The second element, i.e., the element at the first row and the second column of the resultant matrix, would be: *(1\*10) + (2\*11) + (3\*12) = 68*
3. The third element, i.e., the element at the second row and the first column of the resultant matrix, would be: *(4\*7) + (5\*8) + (6\*9) = 122*
4. The last element, i.e., the element at the second row and the second column of the resultant matrix, would be: *(4\*10) + (5\*11) + (6\*12) = 167*

A number line with numbers

Description automatically generated with medium confidence

**Determinant of a matrix**

The determinant of a matrix is a **special number** that can only be calculated from a square matrix, i.e., a matrix with the same number of rows and columns.

For example: A **2 x 2** or **3 x 3** matrix would be considered a square matrix.

**For a 2 × 2 square matrix:**

A diagram of a diagram

Description automatically generated with medium confidence

The determinant is calculated using the below formula:

***det(A)= a\*d - b\*c***  
*"The determinant of A equals a times d minus b times c"*

**Note**: In Python, vectors and matrices are represented as Numpy arrays. Numpy is one of the most popular linear algebra packages in Python and can be used to implement several basic mathematical operations on matrices, such as calculating dot products and determinants.

[**Previous**](https://olympus.mygreatlearning.com/courses/87714/files/8697114?module_item_id=4509035)

[**Next**](https://olympus.mygreatlearning.com/courses/87714/pages/eigenvectors-and-eigenvalues?module_item_id=3460728)

Eigenvectors and Eigenvalues

Consider that we have a vector **V** and a matrix **A**.

If we multiply matrix **A** with vector **V**, we obtain a new, transformed vector.

A graph with a square and a number

Description automatically generated with medium confidence

The below figure shows that multiplying by the matrix **A** has scaled the vector **V** to a new vector, with a different magnitude and a slightly different direction.

A graph of a line graph

Description automatically generated

As we know, vectors have both a magnitude and a direction. The new vector **AV** seems to have a different direction as well as magnitude in comparison to the old vector **V**.

In linear algebra, the operation above is known as a **linear transformation.**It is not only restricted to scaling - linear transformations can be used for flipping, rotating, shearing and other mathematical operations.

Let’s now consider a different vector **V** and multiply it with the same matrix **A**:A graph with a line drawn on it

Description automatically generated

In this example, we notice that something different has happened.

We see that in this instance, while the magnitude of the new vector is certainly different, **its direction has not changed.**

These special vectors are known in linear algebra as **Eigenvectors**.

As illustrated in the example above, given the corresponding matrix A, eigenvectors are directionally-invariant when multiplied with that matrix i.e. they don’t change their direction on multiplication with matrix A - they merely get scaled in terms of magnitude. The value with which the eigenvector gets scaled, is known as the **Eigenvalue,**denoted by the symbol lambda.

In our example, the eigenvector is LaTeX: \binom{1}{1}  and the eigenvalue is 3.

So to summarize, the eigenvector of a matrix is a vector whose direction does not change when a linear transformation (matrix multiplication) is performed on it.

Mathematically, the equation is represented as:



* **A** - Transformation matrix
* **v** - Eigenvector
*  - Eigenvalue

Taking  to the left side:



Lambda is a scalar, so taking **v**out in common:



An eigenvector is a non-zero vector. So, **v** cannot be zero.

Hence, to satisfy the right-hand side condition,  needs to be zero.

If we were to multiply by the [inverse](https://byjus.com/maths/inverse-matrix/) of the matrix on both sides:



**Since the product of a matrix and its inverse is I (the identity matrix)**, equating the left and right sides, we get:



This is contradictory, as we have mentioned above that the Eigenvector cannot be zero.

So that means we cannot actually use the inverse of , as it is not an invertible matrix.

In linear algebra, **if a matrix is not invertible then the determinant of the matrix is equal to zero.**

That means, we only need to solve the equation , to get the eigenvalues and through them, the eigenvectors.

Due to Numpy in Python, we do do not need to perform these operations by hand. Numpy has functions to find the eigenvalues and eigenvectors of a matrix for us.

**Why are Eigenvectors important?**

The directional invariance of Eigenvectors turns out to be incredibly important, and is utilized by many applications - Principal Component Analysis (PCA), for example, is a highly popular data projection technique that can be used to reduce the dimensionality of a dataset and visualize it in lower dimensions.

This technique will be discussed in the first lecture of the week.

To understand the concept of Eigenvectors and Eigenvalues in some more detail, check out this video from 3 Blue 1 Brown:

[Eigenvectors and Eigenvalues](https://www.youtube.com/watch?v=PFDu9oVAE-g)

This video talks about Eigenvectors and Eigenvalues with the help of visual representations and an example.

It includes the calculations that are needed to get these values, and how to interpret them.

The Covariance Matrix

**Variance:**Variance helps us understand how far our random variable is spread out from the mean.

Let's use a hypothetical income distribution of people as an example for reference. The formula for variance is given by:

A number and numbers with black text

Description automatically generated with medium confidence

where **n** is the number of samples (ex: the number of people in the income distribution) and **x̄** is the mean of the random variable **x** (ex: the mean of the income).

**Covariance:**Covariance, on the other hand, measures the extent to which two random variables vary together.

For a similar example, let's consider the income of a person and the expenditure of that person in a population. The formula for covariance is given by:

A mathematical equation with numbers and symbols

Description automatically generated

where **n** is the number of samples (ex: the number of people) and **x̄** is the mean of the random variable **x** (represented as a vector). The variance, **σ2(x)**, of a random variable, **x**, can also be expressed as the covariance with itself, i.e, **σ(x,x)**.

**Covariance Matrix:**Following from the previous equations, the covariance matrix for two dimensions is given by:

A group of black letters

Description automatically generated

In this matrix, the variances appear along the diagonal and the covariances appear in the off-diagonal elements.

**Note:**The function **numpy.cov()**in Python's Numpy variable can be used to get the covariance matrix in Python.

Dimensionality Reduction

In real-world situations, we often deal with **high-dimensional** data, with lots of columns or "features" that represent the information collected about each observation. High-dimensional data is disadvantageous for a couple of reasons:

* It is generally difficult to analyze or visualize high-dimensional data and identify hidden patterns.
* Not all the features or dimensions of the data are equally important.

Therefore, we need to reduce the dimensionality of the dataset in such a way that by losing only a minimal amount of information, we can visualize the data and identify patterns more easily with a smaller number of features.

Two important techniques that we can use for dimensionality reduction are:

* PCA
* t-SNE

**PCA**

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a dataset consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset.

This is done by transforming the variables to a new coordinate space of variables, which are known as principal components (or simply, the PCs), and are orthogonal to each other.

The selection of principal components is such that the retention of variation present in the original variables is the maximum for the first principal component and decreases as we move down in the order. The principal components are the eigenvectors of the covariance matrix, and hence they are orthogonal.

A graph of a line with arrows and dots

Description automatically generated with medium confidence

[Image Source](https://medium.com/@raghavan99o/principal-component-analysis-pca-explained-and-implemented-eeab7cb73b72)

**t-SNE**

The t-SNE algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space. This is done by setting the probabilities from the low-dimensional space similar to those of the high dimensional space. We measure the difference between the probability distributions of the two-dimensional spaces using Kullback-Leibler divergence and try to optimize it.

A graph showing a number of dots

Description automatically generated with medium confidence

[Image Source](http://datasciencecornwall.blogspot.com/2020/04/visualising-high-dimensional-data-with.html)

We will learn about KL divergence in the next section.

[**Previous**](https://olympus.mygreatlearning.com/courses/87714/pages/the-covariance-matrix?module_item_id=3460729)

[**Next**](https://olympus.mygreatlearning.com/courses/87714/pages/kullback-leibler-kl-divergence?module_item_id=3460731)

Kullback-Leibler (KL) Divergence

Sometimes, in machine learning, it is desirable to analyze the actual and observed probability distribution to quantify the difference in the distributions of a random variable. We calculate KL divergence to measure that difference.

First, let us understand what divergence is.

Divergence is a measure of how one distribution differs from another. It is not symmetrical in nature i.e. score of the divergence for distributions p and q would give a different score from q and p.

Here we will talk about one specific type of divergence, the Kullback-Leibler (KL) divergence.

**KL divergence:**It quantifies how much one probability distribution differs from another probability distribution.

The KL divergence between the two distributions p and q can be calculated using the formula:

A mathematical equation with numbers and symbols

Description automatically generated

Two important points to note about KL divergence:

1. The lower the KL divergence value, the better the two distributions match. If two distributions perfectly match, then it is zero.
2. It is not symmetrical i.e.

LaTeX: D_{KL}\left(p\left|\right|q\right)!=D_{KL}\left(q\left|\right|p\right)

**The intuition behind the KL divergence score:** When the probability for an event from p is large, but the probability for the same event in q is small, there is a large divergence. When the probability from p is small and the probability from q is large, there is also a large divergence, but not as large as in the former case.

To know more about KL divergence, please refer to this video [here.](https://youtu.be/xjKm4BcwqX8)

This video talks about the intuition behind KL divergence by explaining it with a numerical example. It also provides a real-life application of this concept.

**Note:**In Python, we use the rel\_entr function to calculate the KL divergence.

[**Previous**](https://olympus.mygreatlearning.com/courses/87714/pages/dimensionality-reduction?module_item_id=3460730)

[**Next**](https://olympus.mygreatlearning.com/courses/87714/pages/the-binomial-and-bernoulli-distributions?module_item_id=3460732)

The Binomial and Bernoulli Distributions

**The Binomial Distribution**

A Bernoulli trial (or a Binomial trial) is a random experiment with exactly 2 possible outcomes, “success” or “failure”, in which the probability of success is the same every time the experiment is conducted.

**Example 1:**

Let’s say we perform an experiment of tossing a coin. In this scenario, the outcome of the experiment can be either heads or tails. Here, we have 2 possible outcomes and we can consider any 1 out of 2 outcomes as a success (depending on the experiment).

If we consider heads as success, the probability of getting heads (probability of success) is always the same, i.e., 0.5. Similarly, if we consider tails as success, the probability of getting tails (probability of success) is always the same, i.e., 0.5.  Hence, this experiment of tossing a coin can be considered a Bernoulli trial.

**Example 2:**

Let’s assume that the probability of a woman having breast cancer all over the world is 0.2, hence, the probability of not getting breast cancer is 1 - 0.2 = 0.8.

Let’s say we perform an experiment of randomly selecting a female candidate from anywhere in the world, to check whether she has breast cancer or not. Here, we have 2 possible outcomes and we can consider any 1 out of 2 outcomes a success. The probability is also the same every time. Hence, this experiment can also be considered a Bernoulli trial.

The binomial distribution represents the probability for '**x**' **successes** of an experiment in '**n**' **trials**, given a **success probability** '**p**' for each trial of the experiment.

The formula for the binomial distribution for a random variable X is given as,

A black lines with a white background

Description automatically generated with medium confidence

Where:

p = Probability of Success

q = 1 - p = Probability of Failure

n = Number of Trials

x = Number of successes desired

The Notation for a binomial distribution is

X ~ B (n, p)

which is read as ‘X follows a binomial distribution with n trials and probability of success in each trial is equal to p’.

There are some of the assumptions that need to be satisfied for a random variable to follow a binomial distribution. They are as follows:

1. There are a fixed number of trials. (represented by the variable n)
2. Each trial has 2 outcomes (called "success" and "failure" for convenience)
3. The trials are all independent of one another.
4. The probability of success, i.e., p is the same for all trials.

**Let's consider an example.** We can use the above formula to find the probability of 2 heads in 6 coin tosses

Here the experiment of tossing a coin is **repeated** 6 times.

* Success = ‘heads’
* n = 6 trials
* p = 0.5
* x = number of heads in 6 tosses which are 2 here
* X follows a binomial distribution with n = 6 and p = 0.5, i.e., X ~ B (6, 0.5)

A graph with different colored bars

Description automatically generated

This is computed using the formula below.

A black lines with a white background

Description automatically generated with medium confidence



So, the probability of getting 2 heads out of 6 coin tosses (trials) is 0.234. Hence, it means that there is a ~23% chance that we will get exactly 2 heads if we repeatedly toss a coin 6 times.

**Let's consider another example.**Suppose 80% of people who purchase shirts are men, then what would be the probability that exactly 7 are men if 10 shirt owners are randomly selected?

Here the experiment of selecting random shirt owners is repeated 10 times,

* Success = “person is male”
* n = 10 trials
* p = 0.8
* x = 7 (Number of male shirt owners desired)
* X has a binomial distribution with n=10 & p=0.8, i.e., X ~ B(10,0.8)

A graph of a bar graph

Description automatically generated with medium confidence

This is computed using the formula below,

A black lines with a white background

Description automatically generated with medium confidence

A black line with numbers

Description automatically generated

So, the probability of getting 7 male shirt owners out of 10 randomly picked shirt owners(trials) is 0.201. Hence, it means that there is a ~20% chance that we will get exactly 7 male shirt owners if we randomly select 10 shirt owners.

**The Bernoulli Distribution**

**A white rectangular object with black text

Description automatically generated**

The Bernoulli Distribution is a special case of the Binomial Distribution where the number of trials is equal to 1. Hence it represents the probability of an event occurring in 1 single trial, for example, the probability of getting heads when a coin is tossed only 1 time is 0.5.

[**Previous**](https://olympus.mygreatlearning.com/courses/87714/pages/kullback-leibler-kl-divergence?module_item_id=3460731)

[**Next**](https://olympus.mygreatlearning.com/courses/87714/files/8728169?module_item_id=4509036)

Curated External Resources - Data Exploration and Visualization

Please find below some curated external links to books, articles, and open-source video content about hypothesis testing and dimensionality reduction.

### Hypothesis Testing

* [This excerpt from a book](https://www.sagepub.com/sites/default/files/upm-binaries/40007_Chapter8.pdf) takes a more detailed look at the foundations of hypothesis testing.
* [This is a good summary document](http://onlinestatbook.com/2/logic_of_hypothesis_testing/logic_hypothesis.pdf) covering the foundational concepts of hypothesis testing and statistical inference.

### PCA / t-SNE

* [An end-to-end comprehensive guide for PCA](https://www.analyticsvidhya.com/blog/2020/12/an-end-to-end-comprehensive-guide-for-pca/)
* Check out this video on PCA by StatQuest: [Click here](https://www.youtube.com/watch?v=FgakZw6K1QQ)
* This article explains [Principal Component Analysis with example](https://www.analyticssteps.com/blogs/introduction-principal-component-analysis-machine-learning).
* This article shows [how to use t-SNE](https://distill.pub/2016/misread-tsne/).
* This is a good summary of the [differences between PCA and t-SNE](https://www.geeksforgeeks.org/difference-between-pca-vs-t-sne/).
* This resource has an excellent [Q&A on t-SNE](https://lvdmaaten.github.io/tsne/)

Happy Learning!

Below is from Live virtual class 2

-----------------------------------------------------------------------------------------------

Curated External Resources - Network Analysis

Please refer to this additional material if you want to know more about Graphs and Networks.

1. **Basics of networks:** This article talks about the structure and different types of networks like directed, undirected, etc.  <https://www.ebi.ac.uk/training/online/courses/network-analysis-of-protein-interaction-data-an-introduction/introduction-to-graph-theory/graph-theory-graph-types-and-edge-properties/>
2. **Overview of networks with python:**This article emphasizes on the overview of networks and their applications in the industry with Python implementations.  [https://www.analyticsvidhya.com/blog/2018/04/introduction-to-graph-theory-network-analysis-python-codes/ (Links to an external site)](https://www.analyticsvidhya.com/blog/2018/04/introduction-to-graph-theory-network-analysis-python-codes/)
3. **Centrality measures in networks:**This short video talks about the different centrality measures and their importance in the networks.

<https://www.youtube.com/watch?v=NgUj8DEH5Tc>

Happy Learning!

Curated External Resources - Unsupervised Learning

These are curated external links to articles, and videos, that provide an opportunity to augment the learning, provided by the live lecture, on various clustering algorithms.

1. **K-Means Algorithm:**This is a centroid-based clustering algorithm that tries to minimize the distance of the points within a cluster with their centroid. To know more about the K-Means clustering, its applications, and its implementation in python, please refer to this article. <https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/>
2. **K-Medoids (PAM) Algorithm:**This is an alternative to K-Means clustering which focuses on the drawbacks of K Means. It is robust to outliers. To know more about the algorithm, please refer to this video. <https://youtu.be/GApaAnGx3Fw>
3. **GMM Clustering:**This algorithm assumes that there are some Gaussian distributions and each of them represents a cluster. It also tries to overcome the drawbacks of K-Means. To know more about GMM clustering and its implementation in python, please refer to this article. <https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/>
4. **Hierarchical Clustering:**This algorithm creates clusters in an ordered manner i.e. top to bottom or bottom to top. It groups similar objects into clusters. To know more about the Hierarchical clustering algorithm and its implementation in python, please refer to this article. <https://www.kdnuggets.com/2019/09/hierarchical-clustering.html>
5. **DBSCAN:**This is a density-based clustering algorithm that aims to create clusters based on the density of a region. It is very robust to outliers. To know more about the DBSCAN algorithm and its implementation in python, please refer to this article. <https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html>

Happy Learning!

[**Previous**](https://olympus.mygreatlearning.com/courses/87714/files/8697107?module_item_id=3460820)

[**Next**](https://olympus.mygreatlearning.com/courses/87714/files/7038975?module_item_id=3460824)

Gaussian Mixture Models

Gaussian Mixture Models (GMMs) are an unsupervised machine learning technique used for clustering data. K-means performs clustering taking only the mean of the data into consideration, but Gaussian Mixture Models use the mean as well as the variance of the data for clustering.

Gaussian Mixture Models assume that the underlying data represents a mixture of Gaussians, as shown in the image below.

A diagram of a cluster of clusters

Description automatically generated

Gaussian Mixture Models are considered probabilistic models for representing normally distributed subpopulations within an overall population. The Gaussian Mixture Model tries to learn the parameters (mean, variance) of each Gaussian in the dataset to cluster the data points. The parameters mean (LaTeX: \mu) and variance (LaTeX: \sigma^2), of each Gaussian, are learned using the **Expectation-Maximization (EM) algorithm**.

## ****How does the EM algorithm work?****

The EM algorithm has 2 steps:

* The E-step
* The M-step

The E-Step and the M-Step are repeated until the parameters mean (LaTeX: \mu) and variance (LaTeX: \sigma^2) **converge**.

#### **E-step:**

1. It will start by placing K Gaussians randomly with mean LaTeX: \mu_k and variance LaTeX: \sum_k for the **k**th Gaussian as shown below. The value of k is passed as a parameter.

2. Then for each Gaussian/cluster we calculate the cluster weights LaTeX: \pi_k which show us how much each cluster is represented over all data points (each cluster's relative size) and can be given as,



3. Then for each datapoint assign a soft assignment,

LaTeX: r_{ik}\:=\:\frac{\pi_k\:\:N\left(x_i\:\mid\:\mu_k,\Sigma_k\right)}{\sum_{j=1}^K\:\pi_j\:N\left(x_i\:\mid\:\mu_j,\:\Sigma_j\right)}

Where, LaTeX: r_{ik} is the probability that LaTeX: x_i is generated from the **k**th cluster and K is the total number of clusters.

#### **M-step:**

After computing the soft assignments for the current iteration, **we use it to update parameters - mean and variance**.

First, we calculate the sum of probabilities of each data point for each cluster.

Let’s say we have a point LaTeX: x_i, its probability vector can be given as

LaTeX: \left[r_{i1},r_{i2},...,r_{ik}\right]

Where,

LaTeX: r_{i1} = probability of point i belonging to cluster 1

LaTeX: r_{i2} = probability of point i belonging to cluster 2

LaTeX: r_{ik} = probability of point i belonging to cluster k

This vector is calculated for each and every data point in the dataset.

Next, we calculate the sum of probabilities for all points belonging to a particular cluster. It can be mathematically represented as,

LaTeX: N^{soft}_k\:=\:\sum^N_{i=1}r_{ik}

LaTeX: N^{soft}_k = sum of all probabilities for all points belonging to cluster k

Then, the cluster weights LaTeX: \pi_1,\:\pi_2,\:...,\:\pi_k,  show us how much each cluster is represented over all data points (each cluster's relative size). The cluster weight of each cluster is given by the ratio of the soft count LaTeX: N^{\:soft}_1,\:N^{\:soft}_2,\:...,\:N_k^{\:soft} over the total number of data points LaTeX: N_k. It's mathematical representation can be given as,

LaTeX: \pi_k\:=\:\frac{N^{\:soft}_k}{N_k}

Now we re-estimate the mean LaTeX: \mu_1,\:\mu_2,\:...,\:\:\mu_k of each cluster using the formula below,

LaTeX: \mu_k\:=\:\frac{1}{N^{\:soft}_k}\:\sum^N_{i=1}\:r_{ik}x_i

Where,

= sum of all probabilities of cluster for which we are re-estimating the mean multiplied by its true value

Now we re-estimate the variance  LaTeX: \Sigma_1,\:\Sigma_2,\:...,\:\Sigma_k, of each cluster using the formula below,

LaTeX: \Sigma_k\:=\:\frac{1}{N^{\:soft}_k}\:\sum^N_{i=1}r_{ik}\left(x_i\:-\:\mu_k\right)\left(x_i-\mu_k\right)^T

Finally, we repeat the E-step and the M-step until the estimates of parameters converge.

### **Convergence**

We use the below function, called log-likelihood, to decide the convergence of parameters.  
  
A number of mathematical symbols

Description automatically generated

When the value of  this formula does not improve by a tolerance/threshold level then we can say that the algorithm has converged.

A graph of a log-like plot

Description automatically generated with medium confidence

In the above graph, we can see that the change in value after 15 iterations is very small, hence we can say that the values have converged and the model has fit successfully.

The below graphic is a representation of the Gaussian Mixture Model learning the parameters at each iteration to resemble the true distribution.  
A graph of a function

Description automatically generated

[**Previous**](https://olympus.mygreatlearning.com/courses/87714/pages/k-means-clustering?module_item_id=3460800)

[**Next**](https://olympus.mygreatlearning.com/courses/87714/pages/hierarchical-clustering?module_item_id=3460802)

DBSCAN

### **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (**DBSCAN)**

DBSCAN is another type of clustering algorithm. It works based on the concept of density difference. Density is interpreted as the concentration of the samples present in a region. The algorithm differentiates between high and low-density regions and clusters the points accordingly.

There are two parameters to the algorithm which is used to define the term dense.

**epsilon (eps)**: It specifies how close points should be to each other to be considered as part of a cluster.

Two points are considered neighbors if the distance between the two points is below the threshold epsilon.

For example, let's say we are defining epsilon = 5, then all the points having a distance less than 5 will be considered a single cluster. The Euclidean distance is calculated by default but we can also use distance measurements.

**minPoints**: The minimum number of points to form a dense region. For example, if we set the minPoints parameter as 4, then we need at least 4 points to form a dense region.

**Let's understand how the algorithm works by considering the data points shown in the below image:**

A group of black dots

Description automatically generated

**Step 1:** We can take any random point and draw a circle around it. The radius of the circle (epsilon) should be passed as a parameter.

A black and red dot

Description automatically generated

**Step 2:** The above step should be continued for all the data points present

A group of black dots with red circles

Description automatically generated

**Step 3:** Identifying **Core points**. To Classify a point as a Core, it should satisfy two criteria. The point which we are considering (red) should **at least** be

* surrounded by the minimum number of points specified (minpoints)
* within a circle of radius (epsilon)

A black and red circle with a white circle

Description automatically generated                                    A red circle with black dots

Description automatically generated

Example: We will take a point and draw a circle (epsilon=5) around it and minPoints as 4 . In the first image above we can find the conditions being satisfied as the circle is overlapping at least 4 points but in the second image the condition is not satisfied as the overlapping is less than 4, such points are called **non-core points**.

**Step 4a:** The below image shows core points (red) and non-core points (black). We randomly pick a Core point and assign it to a cluster.

A black and red dots

Description automatically generatedA red and black dots

Description automatically generated

**Step 4b:** The core points that are within the radius of epsilon are all added to the cluster. This continues until there is no core point in the radius of epsilon. Initially, it adds only the core points to the cluster.

A group of black and red dots

Description automatically generatedA black and green dots

Description automatically generated

**Step 5:** When the cluster reaches a non-core point and if it falls in the epsilon range, it will be added to the cluster but can't be used to extend the cluster.

A black and green dots

Description automatically generatedA diagram of a non-core point

Description automatically generated

**Step 6:** Once a cluster is completely formed, then a new core point will be chosen to form another cluster and the process continues. There will be some points that will not be a part of any cluster and they will be treated as outliers.

A group of green and blue dots

Description automatically generated

In the above images, the green represents points that belong to the first cluster, blue points represent points that belong to the second cluster whereas the black points represent outliers.