Day 07, 08 & 09

30 Days of Machine Learning

Multi-Layer Perceptron, Unsupervised Learning

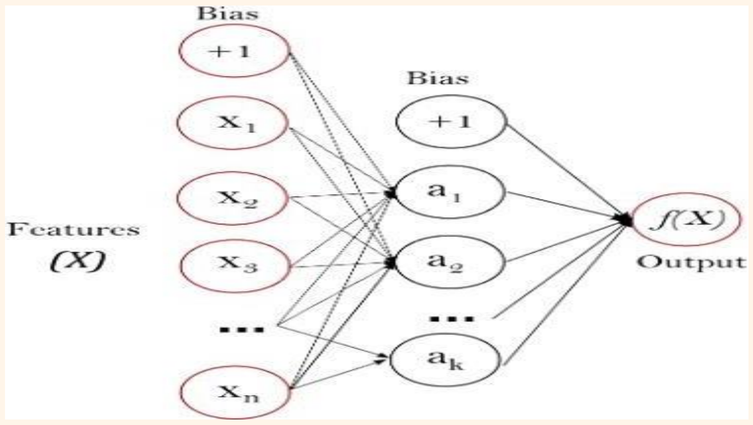
Prepared By

Ahmed Ali

Quaid-e-Awam University of Engineering Science & Technology Nawabshah

Multi-Layer Perceptron:

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function f(⋅):Rm→Ro by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X=x1,x2,...,xm and a target y, it can learn a non- linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure shows a one hidden layer MLP with scalar output.



The leftmost layer, known as the input layer, consists of a set of neurons {xi|x1,x2,...,xm} representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation w1x1+w2x2+...+W_mX_m, followed by a non-linear activation function g(⋅):R→R - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

The module contains the public attributes coefs_ and intercepts_. coefs_ is a list of weight matrices, where weight matrix at index i represents the weights between layer i and layer i+1. intercepts_ is a list of bias vectors, where the vector at index i represents the bias values added to layer i+1.

The advantages of Multi-layer Perceptron are:

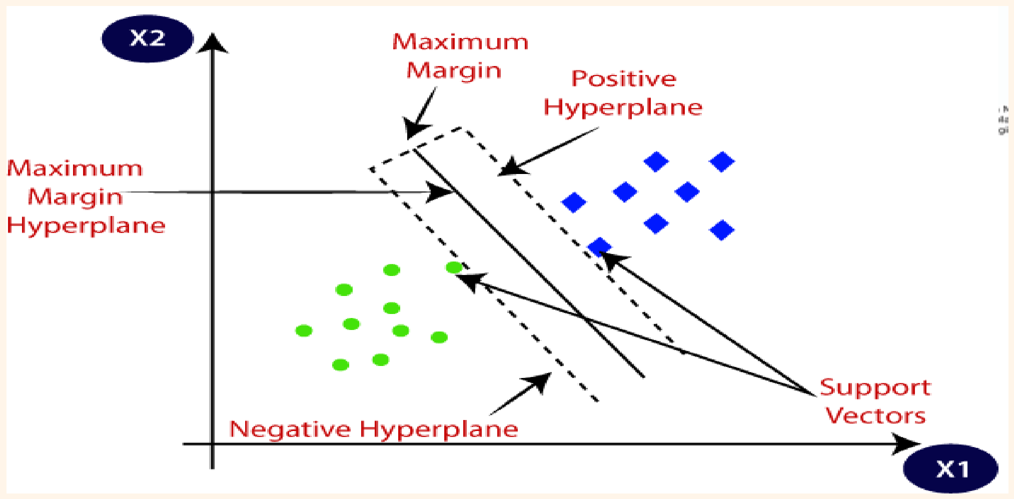
- Capability to learn non-linear models.
- Capability to learn models in real-time (on-line learning) using partial_fit.

The disadvantages of Multi-layer Perceptron (MLP) include:

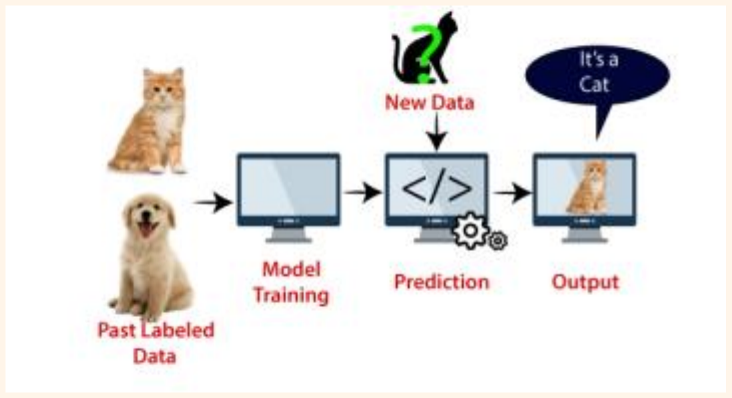
- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore, different random weight initializations can lead to different validation accuracy.
- MLP requires tuning a few hyperparameters such as the number of hidden neurons, layers, and iterations.
- MLP is sensitive to feature scaling.

Support Vector Machines:

The Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. Based on the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for Face detection, image classification, text categorization, etc.

Types of Support Vector Machine Algorithm:

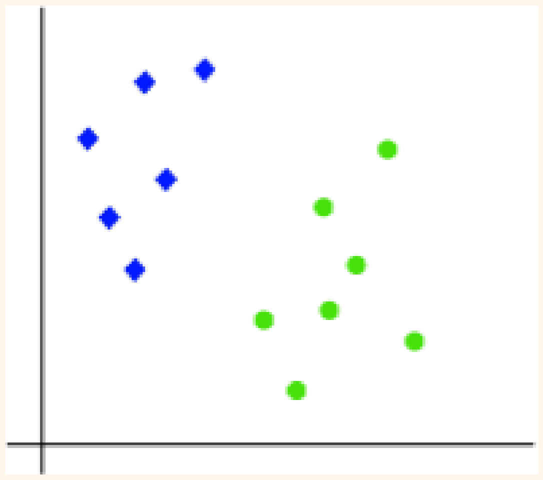
1. **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

2. **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

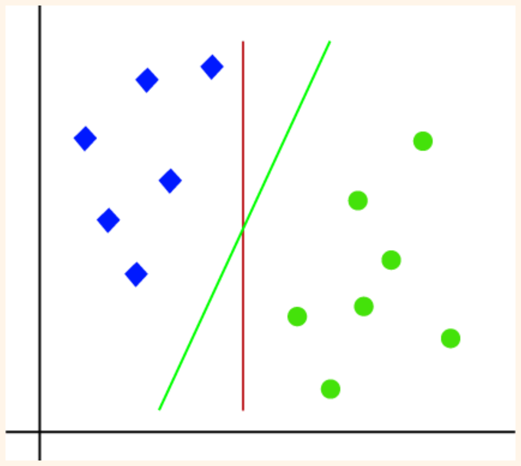
Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n- dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.
The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.
We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:
The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector. How does SVM works?
Linear SVM:
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:
So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Ahmed Ali

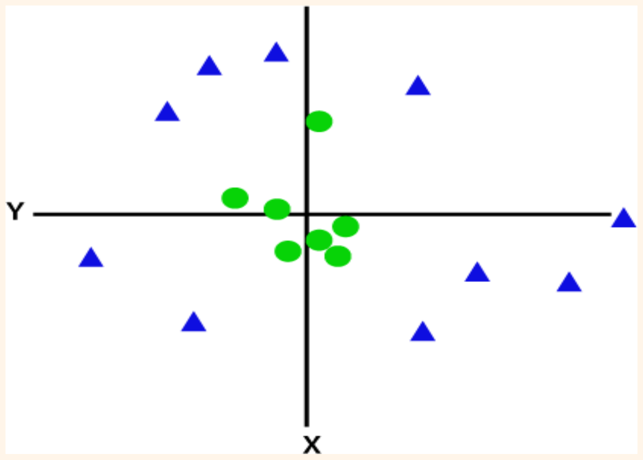
So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

Non-Linear SVM:

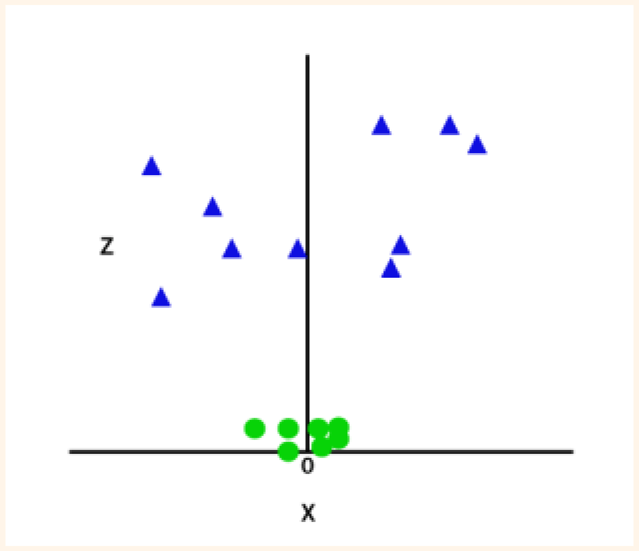
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



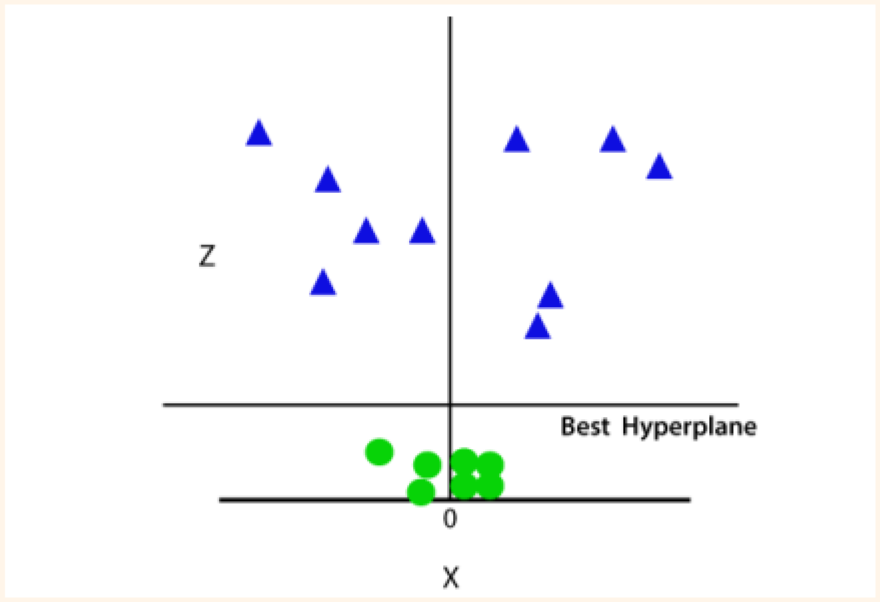
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. It can be calculated as:

$$z=x2 +y2$$

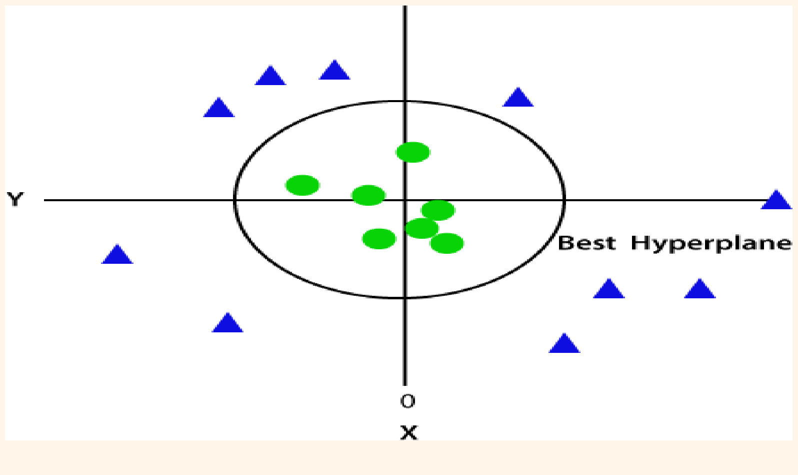
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



SVM Kernels:

In practice, SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, kernel converts non separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible, and accurate. The following are some of the types of kernels used by SVM.

Linear Kernel:

It can be used as a dot product between any two observations. The formula of linear kernel is as below

$$K(x,xi)=sum(x*xi)$$

From the above formula, we can see that the product between two vectors say x & xi is the sum of the multiplication of each pair of input values.

Unsupervised Machine Learning

" Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision."

Introduction to clustering
As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:
"Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision."
Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

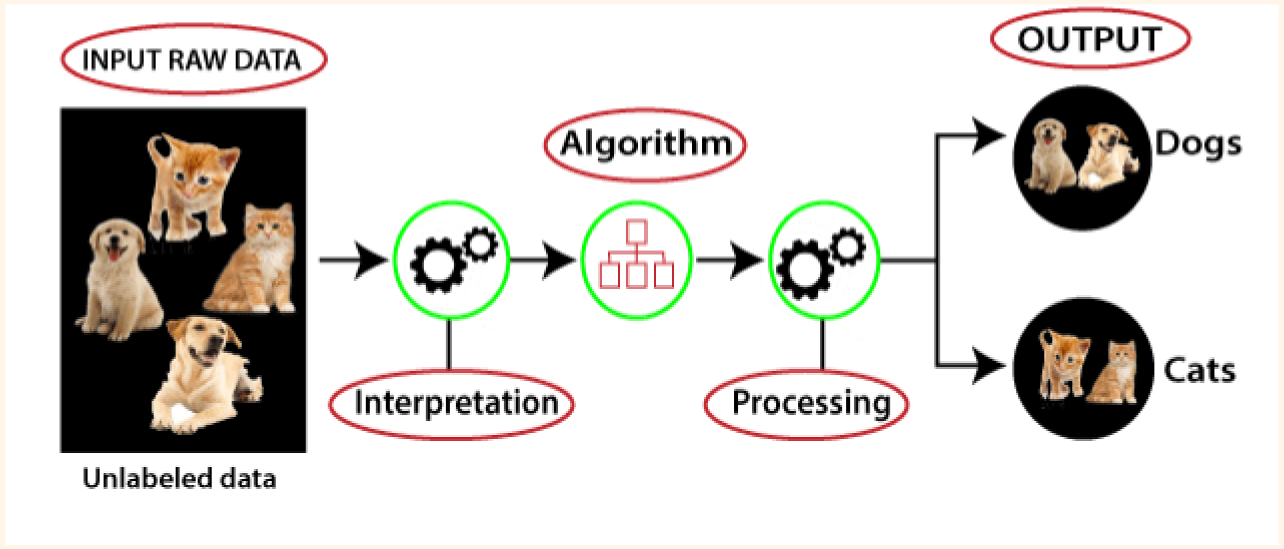
Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:



Ahmed Ali

Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

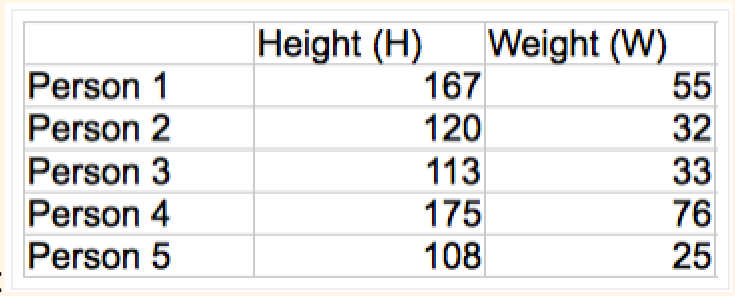
Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** problems.	Unsupervised Learning can be classified in **Clustering** and **Associations** problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is closer to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

Ahmed Ali

k-means clustering algorithm

One of the most used clustering algorithms is k-means. It allows to group the data according to the existing similarities among them in k clusters, given as input to the algorithm. I’ll start with a simple example.

Let’s imagine we have 5 objects (say 5 people) and for each of them we know two features (height and weight). We want to group them into k=2 clusters. Our Dataset will look like This:

:


	Height (H)	Weight (W)
Person 1	167	55
Person 2	120	32
Person 3	113	33
Person 4	175	76
Person 5	108	25

First, we must initialize the value of the centroids for our clusters. For instance, let’s choose Person 2 and Person 3 as the two centroids c1 and c2, so that c1= (120,32) and c2= (113,33).

Now we compute the Euclidian distance between each of the two centroids and each point in the data. If you did all the calculations, you should have come up with the following numbers:

	Distance of object from *c1*	**Distance of object from *c2***
Person 1	52.3	58.3
Person 2	0	7.1
Person 3	7.1	0
Person 4	70.4	75.4
Person 5	13.9	9.4

At this point, we will assign each object to the cluster it is closer to (that is taking the minimum between the two computed distances for each object).

We can then arrange the points as follows:

Person 1 → cluster 1

Person 2 → cluster 1

Person 3 → cluster 2

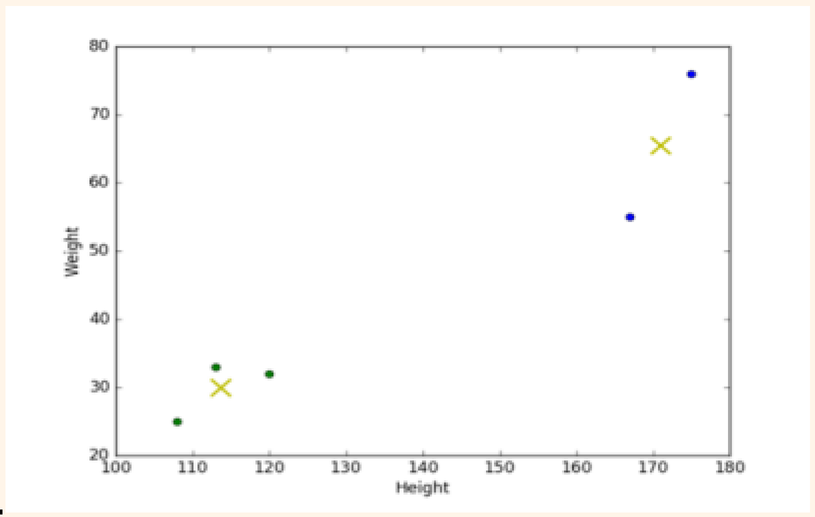
Person 4 → cluster 1 Person 5→ cluster 2

Let’s iterate, which means to redefine the centroids by calculating the mean of the members of each of the two clusters.

So c’1 = ((167+120+175)/3, (55+32+76)/3) = (154, 54.3) and c’2 = ((113+108)/2, (33+25)/2) = (110.5, 29)

Ahmed Ali

Then, we calculate the distances again and re-assign the points to the new centroids.
We repeat this process until the centroids don’t move anymore (or the difference between them is under a certain small threshold).
In our case, the result we get is given in the figure below. You can see the two different clusters labelled with two different colors and the position of the centroids, given by the crosses.



How to apply k-means?
As you probably already know, I’m using Python libraries to analyze my data. The k-means algorithm is implemented in the scikit-learn package. To use it, you will just need the following line in your script:

What if our data is… non-numerical?
At this point, you will maybe have noticed something. The basic concept of k-means stands on mathematical calculations (means, Euclidian distances).
But what if our data is non-numerical or, in other words, categorical? Imagine, for instance, to have the ID code and date of birth of the five people of the previous example, instead of their heights and weights.
We could think of transforming our categorical values in numerical values and eventually apply k- means.
But beware: k-means uses numerical distances, so it could consider close two distant objects that merely have been assigned two close numbers.
k-modes is an extension of k-means. Instead of distances it uses dissimilarities (that is, quantification of the total mismatches between two objects: the smaller this number, the more similar the two objects). And instead of means, it uses modes.
A mode is a vector of elements that minimizes the dissimilarities between the vector itself and each object of the data. We will have as many modes as the number of clusters we required, since they act as centroids.

Ensemble and Probabilistic Learning

Ensemble Learning: Model Combination Schemes, Voting, Error-Correcting Output Codes, Bagging: Random Forest Trees, Boosting: Ad boost, Stacking.

Probabilistic Learning: Gaussian mixture models - The Expectation-Maximization (EM) Algorithm, Information Criteria, Nearest neighbor methods - Nearest Neighbor Smoothing, Efficient Distance Computations: the KD-Tree, Distance Measures.

Ensemble learning usually produces more accurate solutions than a single model would.

Ensemble Learning is a technique that create multiple models and then combine them to produce improved results. Ensemble learning usually produces more accurate solutions than a single model would.

Ensemble learning methods is applied to regression as well as classification.

- Ensemble learning for regression creates multiple repressors i.e., multiple regression models such as linear, polynomial, etc.
- Ensemble learning for classification creates multiple classifiers i.e., multiple classification models such as logistic, decision tress, KNN, SVM, etc.

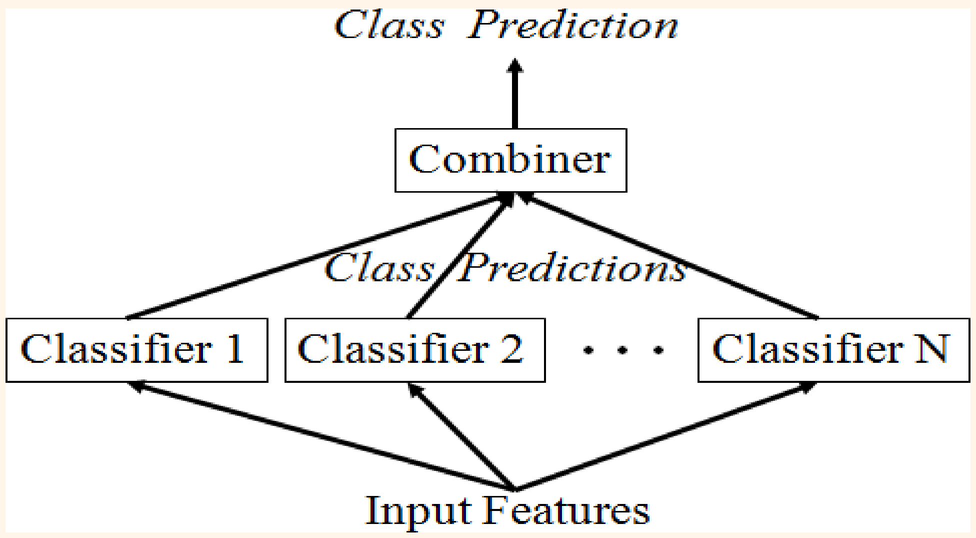


Figure 1: Ensemble learning view

Which components to combine?

- different learning algorithms
- same learning algorithm trained in different ways
- same learning algorithm trained the same way

There are two steps in ensemble learning:

Multiples machine learning models were generated using same or different machine learning algorithm. These are called "base models". The prediction performs based on base models.

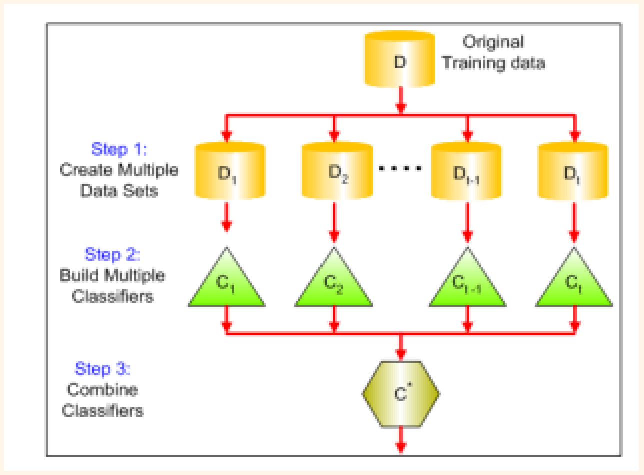
Techniques/Methods in ensemble learning

Voting, Error-Correcting Output Codes, Bagging: Random Forest Trees, Boosting: Adaboost, Stacking.

Model Combination Schemes - Combining Multiple Learners We discussed many different learning algorithms in the previous chapters. Though these are generally successful, no one single algorithm is always the most accurate. Now, we are going to discuss models composed of multiple learners that complement each other so that by combining them, we attain higher accuracy.

There are also different ways the multiple base-learners are combined to generate the final output:

Figure2: General Idea - Combining Multiple Learners:


D
Original Training data
Step 1: Create Multiple Data Sets
D1
D2
Dt-1
Dt
Step 2: Build Multiple Classifiers
C1
C2
Ct-1
Ct
Step 3: Combine Classifiers
C*

Multiexpert combination

Multiexpert combination methods have base-learners that work in parallel. These methods can in turn be divided into two:

In the global approach, also called learner fusion, given an input, all base-learners generate an output, and all these outputs are used.

Examples are voting and stacking.

In the local approach, or learner selection, for example, in mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output.

Multistage combination

Multistage combination methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. The idea is that the base-learners (or the different representations they use) are sorted in increasing complexity so that a complex base-learner is not used (or its complex representation is not extracted) unless the preceding simpler base-learners are not confident.

An example is cascading.

Let us say that we have L base-learners. We denote by dj(x) the prediction of base-learner Mj given the arbitrary dimensional input x. In the case of multiple representations, each Mj uses a different input representation xj . The final prediction is calculated from the predictions of
the base-learners:

$$y = f(d1, d2, \ldots, dL \mid \Phi)$$

where f (·) is the combining function with Φ denoting its parameters.

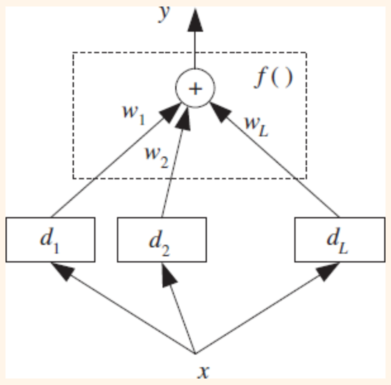


Figure 1: Base-learners are dj and their outputs are combined using f (·). This is for a single output; in the case of classification, each base-learner has K outputs that are separately used to calculate yi, and then we choose the maximum. Note that here all learners observe the same input; it may be the case that different learners observe different representations of the same input object or event.

When there are K outputs, for each learner there are dji(x), i = 1, . . . , K,
j = 1, . . . , L, and, combining them, we also generate K values, yi, i = 1, . . . , K and then for example in classification, we choose the class with the maximum Yi value:

$$\text{Choose } C_i \text{ if } y_i = \max_{k-1}^{K} y_k$$

Voting

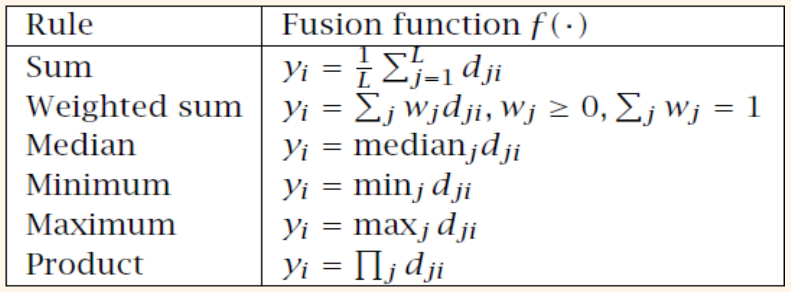
The simplest way to combine multiple classifiers is by voting, which corresponds to taking a linear combination of the learn
ers, Refer figure 1.

$$y_i = \sum_i w_j d_{ji} \text{ where } w_j \geq 0, \sum_i w_j = 1$$

Ahmed Ali

This is also known as ensembles and linear opinion pools. In the sim plest case, all learners are given equal weight and we have simple voting that corresponds to taking an average. Still, taking a (weighted) sum is only one of the possibilities and there are also other combination rules, as shown in table 1. If the outputs are not posterior probabilities, these rules require that outputs be normalized to the same scale.

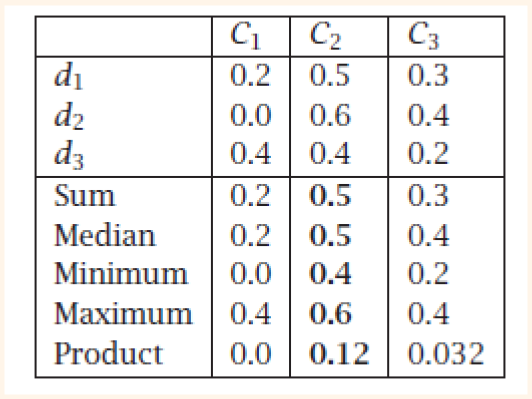
Table 1 – **Classifier combination rules**



Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

An example of the use of these rules is shown in table 2, which demonstrates the effects of different rules. Sum rule is the most intuitive and is the most widely used in practice. Median rule is more robust to outliers; minimum and maximum rules are pessimistic and optimistic, respectively. With the product rule, each learner has veto power; regardless of the other ones, if one learner has an output of 0, the overall output goes to 0. Note that after the combination rules, yi do not necessarily sum up to 1.

Table 2: Example of combination rules on three learners and three classes.



	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.4
Minimum	0.0	**0.4**	0.2
Maximum	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.032

In weighted sum, dji is the vote of learner j for class Ci and wj is the weight of its vote. Simple voting is a special case where all voters have equal weight, namely, wj = 1/L. In classification, this is called plurality voting where the class having the maximum number of votes is the winner.

When there are two classes, this is majority voting where the winning class gets more than half of the votes. If the voters can also supply the additional information of how much they vote for each class (e.g., by the posterior probability), then after normalization, these can be used as weights in a weighted voting scheme. Equivalently, if dji are the class posterior probabilities, P(Ci | x,Mj), then we can just sum them up (wj = 1/L) and choose the class with maximum yi .

In the case of regression, simple or weighted averaging or median can be used to fuse the outputs of base-regressors. Median is more robust to noise than the average.

Another possible way to find wj is to assess the accuracies of the learners (regressor or classifier) on a separate validation set and use that information to compute the weights, so that we give more weights to more accurate learners.
Voting schemes can be seen as approximations under a Bayesian framework with weights approximating prior model probabilities, and model decisions approximating model-conditional likelihoods.

$$P(C_i|x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i|x, \mathcal{M}_j)P(\mathcal{M}_j)$$

Simple voting corresponds to a uniform prior. If we have a prior distribution preferring simpler models, this will give larger weights to them. We cannot integrate over all models; we only choose a subset for which we believe P(Mj) is high, or we can have another Bayesian step and calculate P(Ci | x,Mj), the probability of a model given the sample, and sample high probable models from this density.

Let us assume that dj are iid with expected value E[dj] and variance Var(dj), then when we take a simple average with wj = 1/L, the expected value and variance of the output are

$$\begin{aligned} E[y] &= E\left[\sum_j \frac{1}{L}d_j\right] = \frac{1}{L}LE[d_j] = E[d_j] \\ \mathrm{Var}(y) &= \mathrm{Var}\left(\sum_j \frac{1}{L}d_j\right) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}L\mathrm{Var}(d_j) = \frac{1}{L}\mathrm{Var}(d_j) \end{aligned}$$

We see that the expected value does not change, so the bias does not change. But variance, and therefore mean square error, decreases as the number of independent voters, L, increases. In the general case,

$$\mathrm{Var}(y) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \mathrm{Var}(d_j) + 2\sum_j \sum_{i<j} \mathrm{Cov}(d_j, d_i)\right]$$

which implies that if learners are positively correlated, variance (and error) increase. We can thus view using different algorithms and input features as efforts to decrease, if not eliminate, the positive correlation.

Ahmed Ali

Up Next:
Error-Correcting output codes

Ahmed Ali