Linear Classifiers

Linear classifiers classify data into labels based on a linear combination of input features.

Therefore, these classifiers separate data using a line or plane or a hyperplane (a plane in more than 2 dimensions).

They can only be used to classify data that is linearly separable.

They can be modified to classify non-linearly separable data

3 major algorithms in linear binary classification –

Perceptron

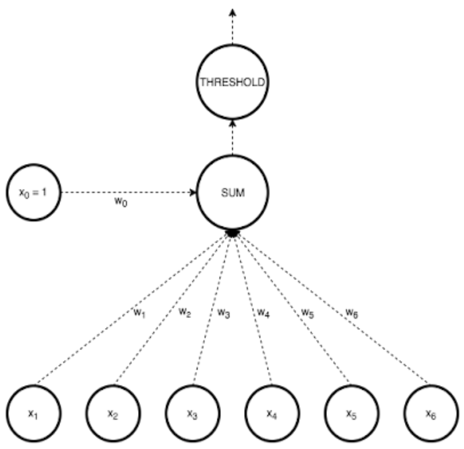
In Perceptron, we take weighted linear combination of input features and pass it through a thresholding function which outputs 1 or 0.

The sign of $w^T x$ tells us which side of the plane $w^T x=0$, the point x lies on.

Thus by taking threshold as 0, perceptron classifies data based on which side of the plane the new point lies on.

The task during training is to arrive at the plane (defined by w) that accurately classifies the training data.

If the data is linearly separable, perceptron training always converges



Perceptron is linear binary classification algorithm. Its functioning is similar to the functioning of a single neuron in our brain. A neuron in our brain takes input signals from many other neurons and based on those inputs, it decides whether to fire or not

Algorithm

Let the input feature vector be -

A perceptron consists of a transfer function and an activation function. The transfer function takes the weighted sum of features from feature vector as input. Let the weight vector be -

To include a bias term, we augment the feature vector with x0 = 1 and weight vector by w0

The transfer function then becomes

The output of transfer function is fed to the activation function. The activation function in this case acts like a threshold

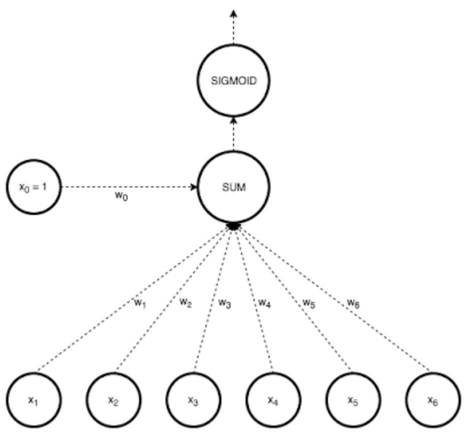
Logistic Regression

In Logistic regression, we take weighted linear combination of input features and pass it through a sigmoid function which outputs a number between 1 and 0.

Unlike perceptron, which just tells us which side of the plane the point lies on, logistic regression gives a probability of a point lying on a particular side of the plane.

The probability of classification will be very close to 1 or 0 as the point goes far away from the plane.

The probability of classification of points very close to the plane is close to 0.5



Overfitting

Training data often contains random noise. The noise could come from various sources like measurement errors, etc. When an optimization algorithm tries to reduce cost, it tries to fit the hypothesis function exactly for all points in the training data. Thu the algorithm ends up overfitting. Such a hypothesis function performs poorly on the test data.

Bayesian Perspective

From Bayesian point of view, avoiding overfitting is similar to adding a prior probability to the data distribution. In case of figure 1, we add a prior which states that the output is most probably a linear function of input. Bayesian learning section describes the Bayesian perspective in detail

Avoiding Overfitting

There are various ways to avoid overfitting in optimization problems. Most popular are -

- l2 Regularization
- l1 Regularization
- Limiting iterations of optimization algorithm

SVM

There can be multiple hyperplanes that separate linearly separable data.

SVM calculates the optimal separating hyperplane using concepts of geometry

SVM is another linear classification algorithm (One which separates data with a hyperplane) just like logistic regression and perceptron algorithms we saw before.

Given some linearly separable data, we can have multiple hyperplanes that can act as separation boundary as shown in figure 1. SVM chooses the "optimal" hyperplane amongst all candidate hyperplanes

To understand concept of "optimal" hyperplane, let us first define some concepts we will use -

- Margin: It is the distance of the separating hyperplane to its nearest point/points
- Support Vectors: The point/points nearest to the separating hyperplane. We will understand why they are called support vectors later

The optimal hyperplane is defined as the one which maximizes the margin.

Thus SVM is posed as an optimization problem where we have to maximize margin subject to the constraint that all points lie on the correct side of the separating hyperplane

If all candidate hyperplanes correctly classify the data, why is maximum margin hyperplane the optimal one?

One intuitive explanation is –
If the incoming samples to be classified contain noise, we do not want them to cross the boundary and be classified incorrectly