MACHINE LEARNING CS60050

Instructor: Dr. Aritra Hazra
Department of Computer Science Engineering
Indian Institute of Technology, Kharagpur
Scribed by: Debjyoti Das Adhikary ; 20CS91F01

Notes for class on 4th February 2021

1 Previously

We have studied how the prime objective of Linear Classification problems is to create a hyperplane differentiating the two classes.

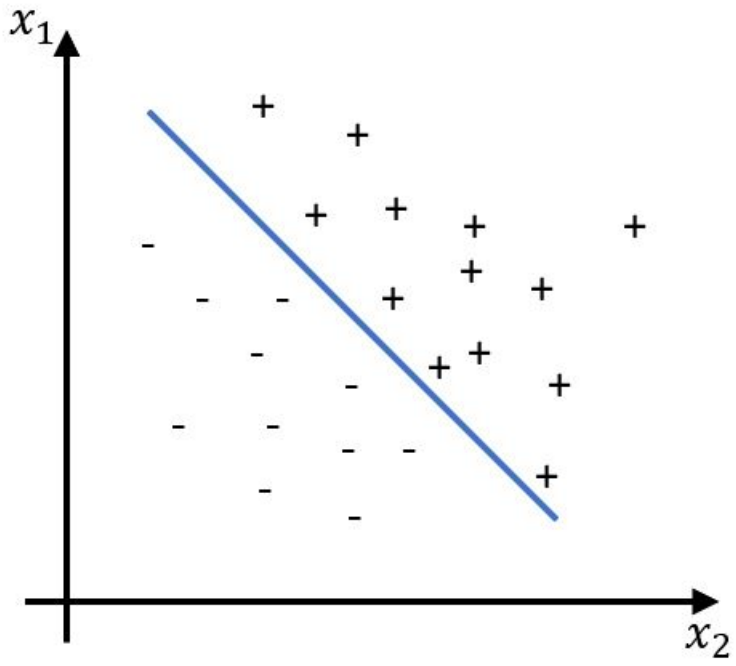

Figure 1: A linear classifier aims to find a single line that differentiates 2 classes.

The model classifies any input x with d number of features as follows:

$$\hat{y} = \begin{cases} 1 & sign(W^T X) > 0 \\ -1 & sign(W^T X) < 0 \end{cases} \tag{1}$$

Where W is the weights of the features which defines the hyperplane. The same method is used for linear regression where the predicted value, $\hat{y} = W^T X$.

The in-sample error, or the error while training is calculated as

$$E_{in} = \frac{1}{N}(\hat{y} - y)^2$$

which can be written as

$$E_{in} = \frac{1}{N}(W^T X - y)^2$$

In order to minimize the error, we calculate its change with respect to the W. Hence, we differentiate with respect to W and minimize it to find the value of W. The $W = [(X^T X)^{-1} Y X^T]Y$
The interesting thing to note is that here the most optimal weights can be calculated in one shot and was no need to update them.

2 Variant of linear Classification

We have discussed how the data points in a dataset are linearly separable and simply computing the most optimum hyperplane will help us classify any unknown data point. But often times, it might happen that the data points might not be linearly separable. For example, consider the following data distribution

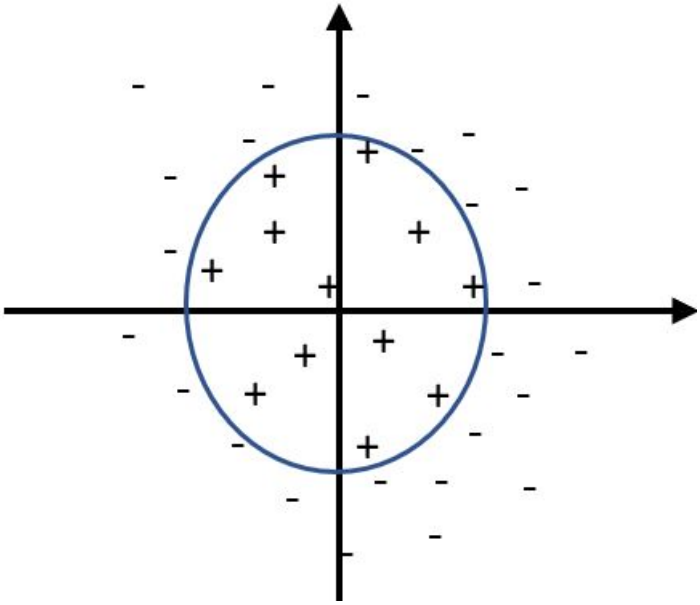

Figure 2: The above distribution is an example of non-linear distribution

In such a case, we try to transform the set of features into new features $x = z = \phi(x)$ so as to make the new features linearly separable. For example, data distribution in figure 2 can be transformed into new feature set where the data $x = x^2$

In process of transforming our features from one form to another, there could be multiple options. Basically, all the possibilities in the quadratic domain of the current features can be the new possible features. $\phi(1, x_1, x_2) = \{1, x_1, x_2, x_1 x_2, x_1^2, x_2^2\}$
But having all these features in our feature set have their own disadvantages which are as follows:

1. This feature set is the most generic feature set. Having these many features might make the data overfit over the training set

2. It is computationally heavy to compute all the features and then to train our model over it.

Further, we can reduce the number of features in such a way that $\phi(1, x_1, x_2) = \{1, x_1^2 + x_2^2\}$ This will transform the data distribution as shown in fig

Next, we can further reduce the feature set to $\phi(1, x1, x2) = x_1^2 + x_2^2$–0.6 < 0 then -1 else +1

The above stated transformation might even produce better results than that of the raw features. However, one should note that this process of transformation has to be done by the user, and in order to do that the user has to dive into the data and uncover the relations between the data. This method is called *Data Snooping*. The major drawback of such practice is that the user, instead of the machine, learns the data.

3 Variant of Linear Regression

As we have discussed, linear regression tries to create a single line for all the data points we have.

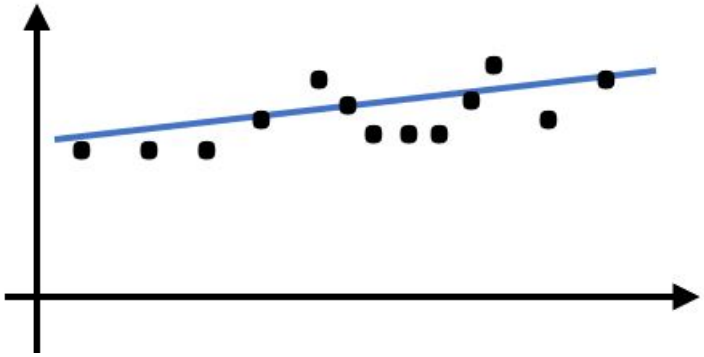

Figure 3: Linear Regression tries to find one line with least error

However one interesting idea is to break the entire training set into small groups of neighbourhoods and then have a linear regression line for all those groups of δ interval.

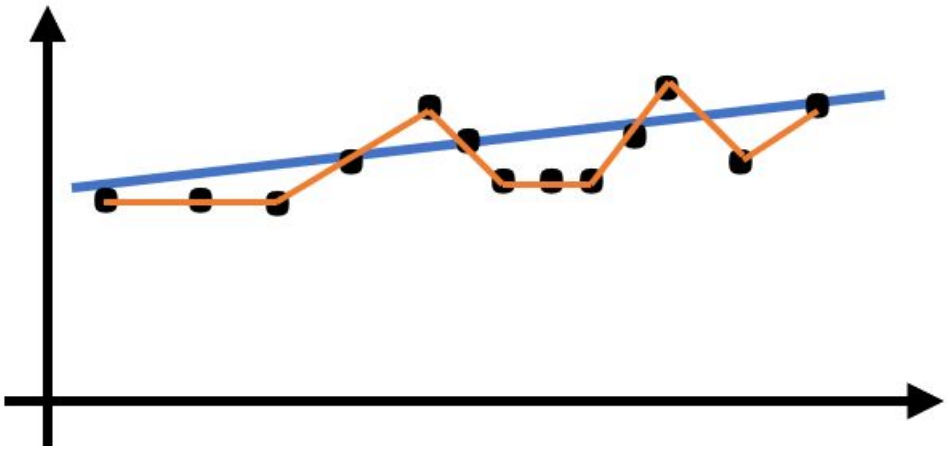

Figure 4: Linear Regression variant which fits multiple lines for different intervals

In this case, while training the model, the objective should be same as that of the linear regression, i.e., to minimize the error between the line and the data points. However, in this variant we need to minimize the error within a certain neighborhood (or interval, δ). So this makes the error as
Fit

$$\text{W:} \min \sum_{i=0}^{n} \alpha_i (W^T X_i - y_i)^2$$

Where

$$\alpha = \exp(\frac{-(x_i - x)^2}{2\tau^2})$$

If we try to trace the value of α for all x, we find that the we get a bell shaped curve indicating the gradual change in the influence of a data point x_i as we go away from out point of interest x

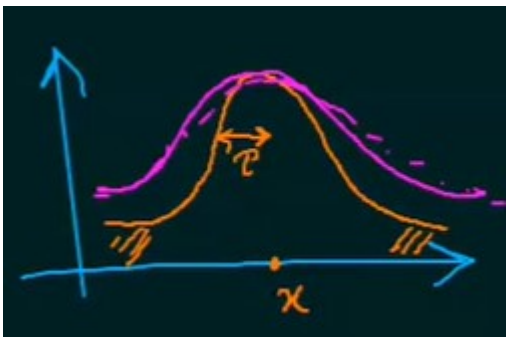

Figure 5: The change in α as we go away from the point of interest

4 Logistic Regression

The most prime objective in any classification problem is to classify a function between 0 and 1. However, we need a certain function f which yields the structure shown in figure 6 for any data distribution.

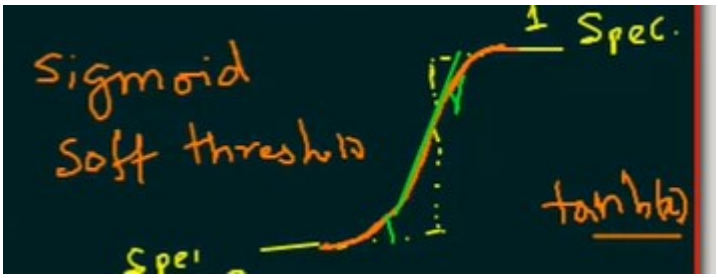

Figure 6: We try to find some function f which yields one after a certain threshold

The main objective behind going for a smooth function is to make our function derivable and to add a certain gradient for predicting the labels.
One such function is:

$$\theta(s) = \frac{e^s}{1 - e^{-s}}$$

Interesting thing to note about the above function is that $\theta(-s) = 1–\theta(s)$ So, now our target is to fit our input into this function to get the output interpretation resemble as the probability for that particular input. Hence our hypothesis becomes

$$H(s) = \theta(s)$$

where $s = W^T X$
So if we consider the maximum likelihood of the function, we will have to compute the probability of the function

$$Prob(y|x) = \begin{cases} h(x) & \text{,y=+1} \\ 1 - h(x) & \text{y=-1} \end{cases} \tag{2}$$

or

$$Prob(y|x) = \begin{cases} \theta(s) & \text{,y=+1} \\ \theta(-s) & \text{y=-1} \end{cases} \tag{3}$$

If we consider the Maximum Likelihood Estimate of the function, we'll have to maximize the product of the probability of all the instances i.e,
Maximize

$$\frac{1}{N} \prod_{i=1}^{N} \theta(y_i W^T x_i)$$

or, minimize

$$-\frac{1}{N}\prod_{i=1}^{N}\theta(y_i W^T x_i)$$

which is equal to

$$-\frac{1}{N}\prod_{i=1}^{N}\frac{1}{(1+e^{y_i W^T x_i})}$$

which can be written as

$$-\frac{1}{N}\sum_{i=1}^{N} ln(\frac{1}{1+e^{y_i W^T x_i}})$$

The above equation gives us the in-sample error

$$E_{in} = \frac{1}{N}\sum_{i=1}^{N} ln(1+e^{y_i W^T x_i})$$

So, now the problem boils down to reduce the above equation in every iteration. Since the objective function is smooth, we can gradually change the feature weights after every iteration based on our E_{in}. So our function becomes

$$W_{t+1} = W_t - \eta\hat{v}$$

Where $\hat{v} = \frac{\nabla E_{in}(W)}{||E_{in}(W)||}$ and η is *learning rate*. The interesting thing to note here is the adaptive change in the weights. This means that initially our step size would be larger at first but as we are approaching towards the minima, the step size or the 'jump' keeps on diminishing.