Machine Learning CS60050

Instructor : Dr. Aritra Hazra
Department of Computer Science Engineering
Indian Institute of Technology, Kharagpur
Scribed By : Sukanya Das, Roll No.: 20CS71P03

Notes for class on 3rd February 2021

1 Previous

Our primary goal is to learn an unknown function $f : X- > Y$, which given the attribute X will predict Y which is a classification problem. Therefore, $f : X- > Y = \{+1, -1\}$. This problem could also be defined as $Prob(Y = 1)$ given X which is represented as $P(y = 1| < X >)$ and $Prob(Y = 0)$ given X which is represented as $P(Y = 0| < X >)$.

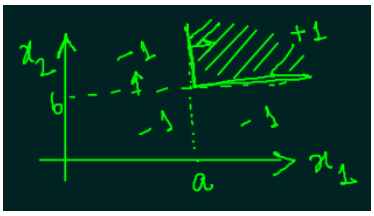


Figure 1: Concept Learning Decision Boundary Surface

When we classify something using concept learning with attribute x_1 and x_2, and suppose we have the hypothesis ($x_1 \geq$a) $\wedge$ ($x_2 \geq$b), We get a decision boundary surface.

When we move into decision tree

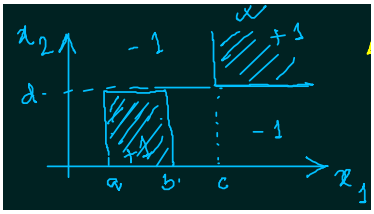
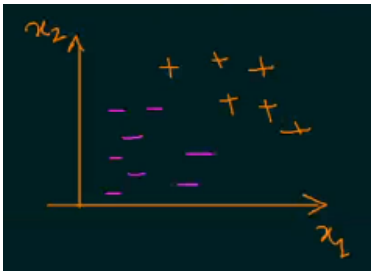


Figure 2: Decision Tree Learning Decision Boundary Surface

This gives us $[(x_1 \geq c) \wedge (x_2 \geq d)] \vee [a \leq x_1 \leq b \wedge x_2 \leq d]$.
Here, decision boundaries are parallel to axis.

2 Linear operators and Hyperplane



In general classification problem, if we want to classify this, the most simplified model to classify it , is a line.

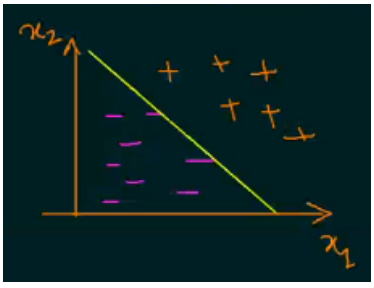
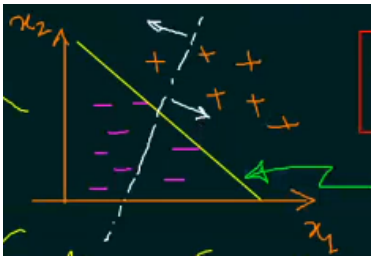


Figure 3: Linear separator

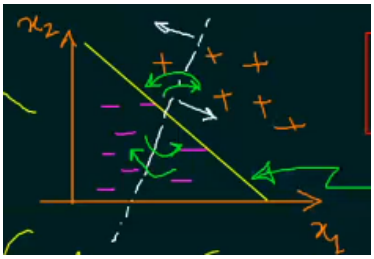
In case of 2-attribute space, we call it as a **linear separator**. It could be a line in 2D space or plane in 3D space, this type of operators called **Hyperplanes**.

In every type of classification problem, we can express the line as, $w_1x_1+w_2x_2+w_0=0$



We'll start with an arbitrary line, if we shift the line like the figure, that

means the value of w_0 is changing $w_1x_1+w_2x_2+w_0=0$, i.e. shifting the line without changing the gradient means we are changing the constant value.



When we're trying to rotate the line, the value of w_1 and w_2 is changing, $w_1x_1+w_2x_2+w_0=0$. So, effectively fitting a particular separator so that it classifies two data, boils down to the fact that how you learn these parameters(w_0, w_1, w_2) from your training examples.

Suppose, we've d no. of attributes, such that,
$w_0+w_1x_1+w_2x_2+....+w_dx_d=0$
Depending on the training data, if we train and find out x,and
when $w_0+w_1x_1+w_2x_2+....+w_dx_d \geq 0$, we'll classify it as + class,
and when $w_0+w_1x_1+w_2x_2+....+w_dx_d <0$, we'll classify it as - class.

For any new point, $x^{new} = \langle x_1^{new}, x_2^{new}, ..., x_d^{new} \rangle$, we'll check that whether fitting these values into different positions of the linear equation $w_0+w_1x_1+....+w_dx_d$ gives the equation of the line to be ≥ 0 or <0, and accordingly we'll classify it + class or - class. This is the whole concept of linear separators.

3 Input representation of a real dataset

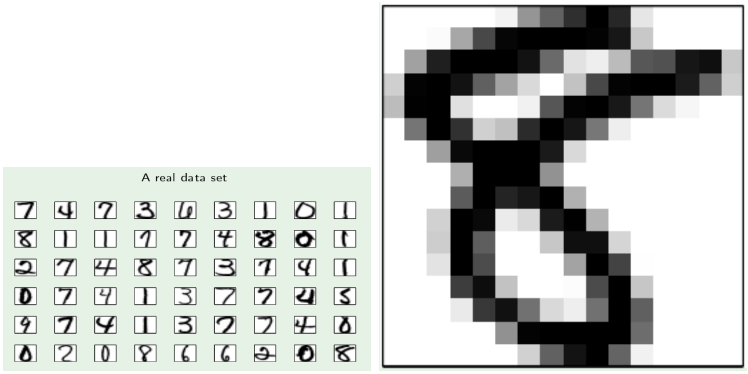


Figure 4: A real data set

In the figure, each pixel has a value in greyscale. if we take it into inputspace, then we'll get (x_1,x_2,...,x_{256}), because each one is a value in our input. If we

apply general linear separator $\sum_{i=0}^{256} w_i x_i$, then we've to learn 257 no. of weights in general to learn the line which is separating it. Then it'll cost too much in terms of computational efficiency.

So, what we'll do

Instead of taking raw data or pixels, we can notice certain patterns expressed by the handwritten digits.

Eg.: In case of '1', it is more symmetric than the handwritten digit '5', means it has more number of black pixels or intensity than '1'.

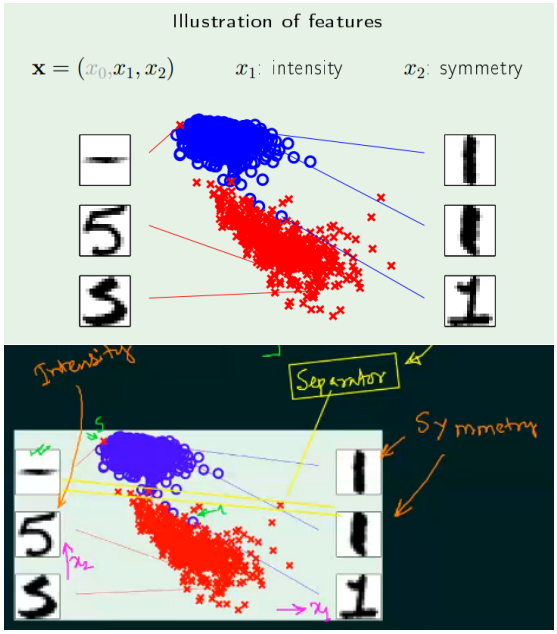


Figure 5: A real data set

Therefore, if we're only looking to the attributes symmetry and intensity, then we can reduce the 256 numbers of weight learnings to only 3 type of weight learnings (w_0,w_1,w_2) in order to find out the linear seperator. So, our equation of linear seperator will be

$w_0+w_1x_1+w_2x_2 = 0$, where x_0=1, $x_1 \rightarrow intensity$,$x_2 \rightarrow symmetry$

4 Perceptron

Perceptron tries to learn the sum formula $\sum_{i=0}^{d} w_i x_i$=0 by adjusting the weight looking at each training example TE_1,TE_2,...,TE_N and then after looking into each training example, it keeps on switching the weights in such a way that finally we result up in a sum line.

Our goal is to fit in a line, twist or change its slope or may be shift the line

in such a way that it fits the data well, depending on the training examples.So, there are two approaches in which we can adjust the weights,

- Take the whole training examples in a batch together and the weights one time.
- *Iterative approach:* Take each training example and adjust the weight and fit the line accordingly.

5 In Sample Error(E_{in}) and Out of Sample Error(E_{out})

Our basic learning assumption is E_{in}(The error rate we get on the same data set we used to build our predictor) tracks E_{out}(The error rate we get on the new data set). If we could reduce E_{in} i.e. in given sample if we could classify in such a way that E_{in} reduces, then E_{out} performance will also be good i.e. a new data will also be classified well.

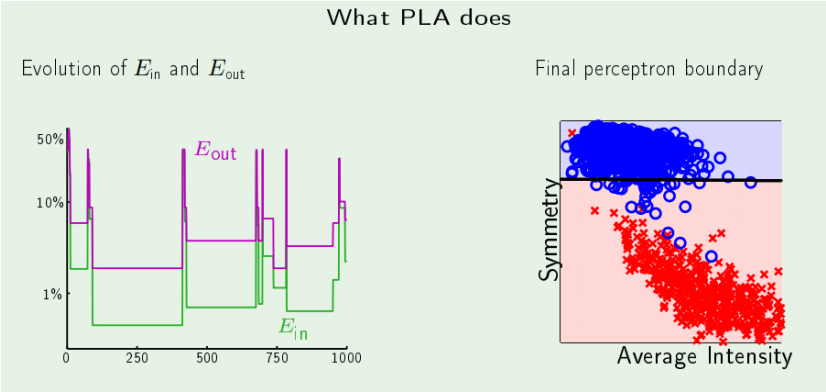


Figure 6: Perceptron Learning Algorithm

When we're classifying using perceptron learning algorithm(PLA), we're getting this kind of classification. Because in PLA, we're taking the example one by one and changing the weights accordingly. Ordering of the example effects on switching the separator line.

Whereas in Pocket Learning Algorithm ('Pocket' means the best solution/in sample performance used i.e. at each position we get w_0,w_1,w_2, we keep this in pocket until we get better one, in terms of E_{in}), we get a better performance.

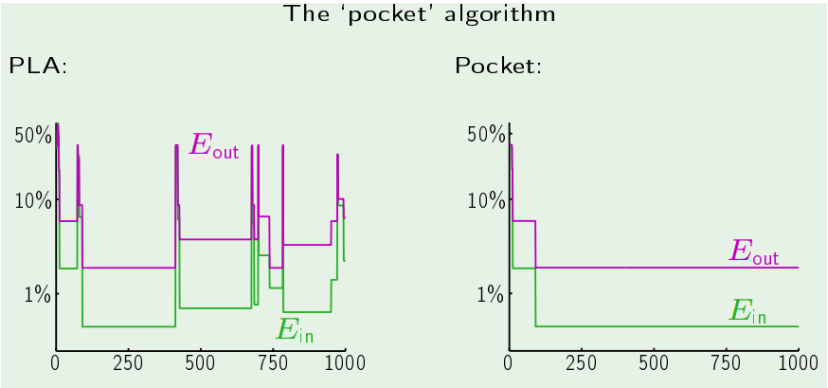


Figure 7: Pocket Learning Algorithm

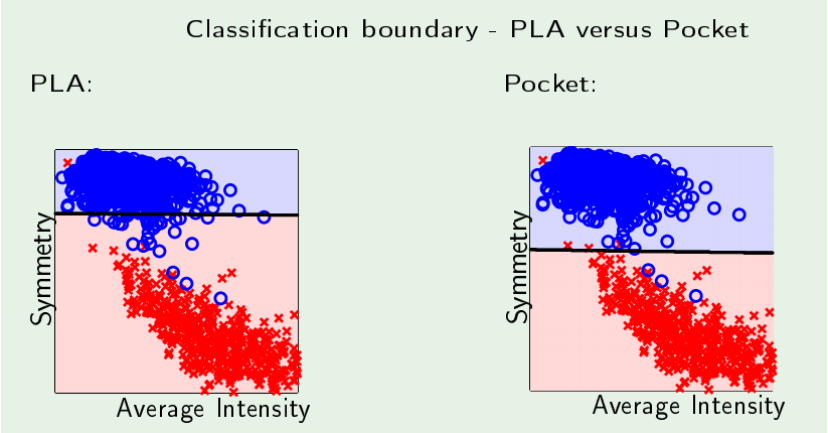


Figure 8: Comparison of Perceptron and Pocket Learning Algorithm

6 Regression

In a credit approval problem, generally we've only 'yes' and 'no' class which leads it to a classification problem. Suppose, now we've credit limit instead of 'yes' and 'no' class, and depending upon the attributes, credit will be approved. So, now this 'credit approval problem' is having 'real valued' output.This type of problem is called **Regression**.

6.1 Linear Regression

Linear regression is used for finding linear relationship between target and one or more predictors.

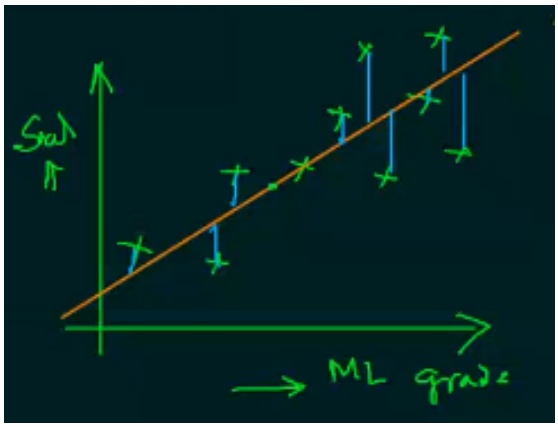


Figure 9: Simple Linear Regression for salary on the basis of ML Grade

Eg.: Suppose, there is a graph for salary given according to your ML grade, and it is linear. We've to predict a line that minimizes the error among the given training data.
The core idea is to obtain a line that best fits the data. The best fit line is the one for which total prediction error (existing training points) are as small as possible. Error is the distance between the point to the regression line.

Calculate the Error/Deviation
Suppose, there are N training examples(TE). Where each TE is
$X_i = [(X_{i1},X_{i2},...,X_{id}),y_i]$
$X_{i1},X_{i2},...,X_{id}$ are characterized in a d-dimensional attribute space, and y_i is a real number and output value.
When we try to fit a linear line in the attribute space, we can say that it is $\sum_{i=0}^{d} w_i x_i = y_i$ and in the vector form it is $W^T \cdot X = Y$,

$$where, W = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ . \\ w_d \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_d \end{bmatrix}$$

The error margin with respect to each point in this linear line is, e_i = (y_i - $\sum_{i=0}^{d} w_i x_i$). If the data point is exactly over this line, then e_i=0. To minimize the error for all TEs, we're using the **Sum Squared Error**, Now, our In-sample error becomes $E_{in} = \frac{1}{N}\sum_{i=0}^{N} e_i^2$

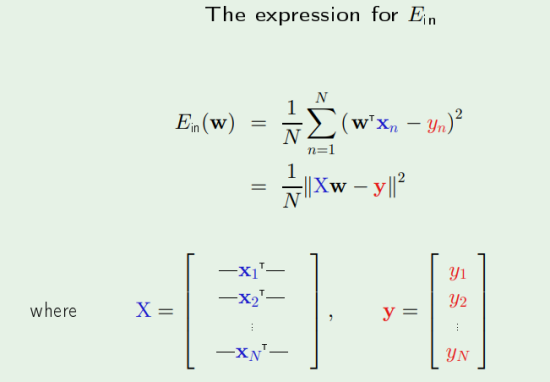


Figure 10: Illustration of In- Sample Error(E_{in})

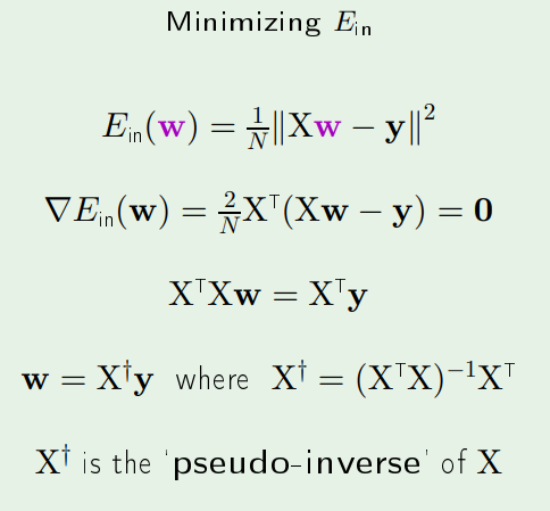


Figure 11: Minimizing the expression E_{in}

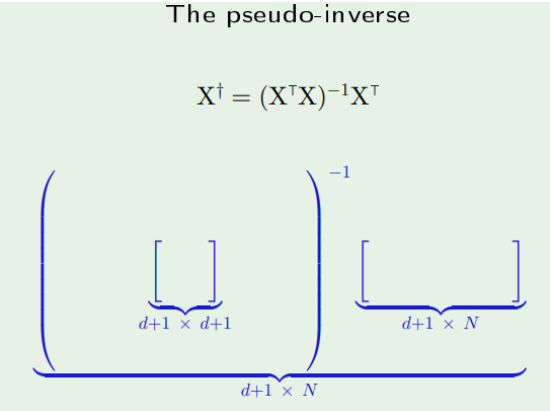


Figure 12: Pseudo-inverse

Working of Linear-regression algorithm

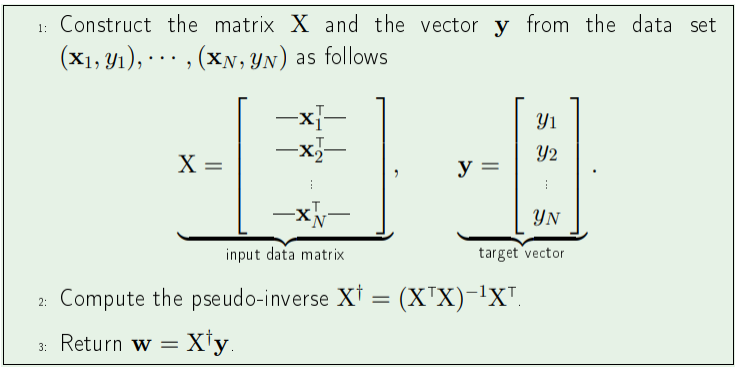


Figure 13: Linear-regression algorithm

7 Gradient Descent Algorithm

A gradient measures how much the output of a function changes if the inputs change a little bit. Gradient descent is an optimization algorithm that's used when training a machine learning model. It's based on a convex function and tweaks its parameters iteratively to minimize a given function to its local minimum.

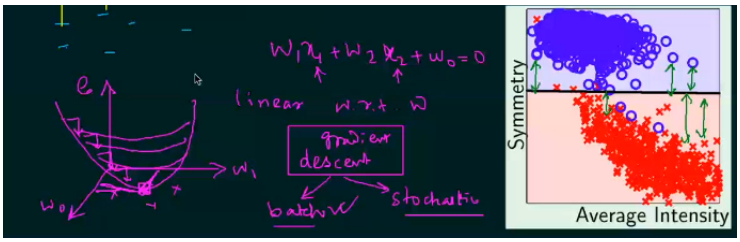


Figure 14: Gradient Descent algorithm

A gradient simply measures the change in all weights with regard to the change in error. We can also think of a gradient as the slope of a function. The higher the gradient, the steeper the slope and the faster a model can learn. But if the slope is zero, the model stops learning. In mathematical terms, a gradient is a partial derivative with respect to its inputs.
There are two ways how gradient descent works, they are

- **Batch Gradient Descent :** In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters.
- **Stochastic Gradient Descent :** 'Stochastic' means 'random'. In this algorithm, while selecting data points at each step to calculate the derivatives, it randomly picks one data point from the whole data set at each iteration to reduce the computations enormously.

[1]

[1]Note 1. This scribe is based on the lecture taught by Prof. Aritra Hazra on 03-feb-2021 in Machine Learning(CS60050) course.
2. All the figures in this document are taken from either handout-06a or Slide-06a uploaded on Machine Learning(CS60050) course website.