CS60050 - Machine Learning (Spring 2021)
Instructor Dr. Aritra Hazra
ML Scribe of March 17, 2021

Sreyasree Mandal (20CS60R23)
Somi Vishnoi(20CS60R25)

March 19, 2021

1 Previous class overview

The discussion started with Ensemble Technique: Simplify the problem with a binary classifier. Binary Classifier is a hypothesis function from the input space to +1 or -1. Binary Classifier, h : x $\rightarrow$ [+1, -1]. It is either 'yes' or 'no'. The whole goal of learning is to minimize in-sample error. Considering the probability of error lies within 0.0 and 1.0. For example, if a flip a coin, the error will be 0.5. If it is a strong classifier, then the error lies near to 0.0. If it is a weak classifier, then the error will be nearer to on the left side of 0.5

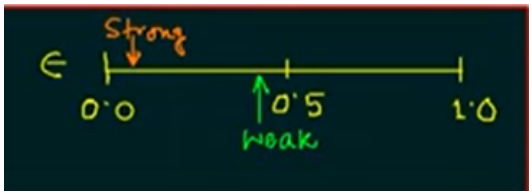

Figure 1: Type of Classifier

2 Goal of Ensemble Learning

The goal is to first create a set of weak classifiers and then among all the weak classifiers, we take a vote and create a hypothesis, H. We call H as the strong

classifier. This entire process is called Ensemble learning. We thought of such a process because though our classifier is weak, suppose a space contains 3 classifiers which go wrong.

2.1 Examples

Let say we have a space containing 3 weak classifiers that go wrong Scenario 1 with three classifiers as shown in the image. There is a first space in which first classifier is wrong, a space 2 in which second classifier is wrong, then we have a third space in which our third classifier is wrong. If we get such an independence, then our ensembling will be strong.

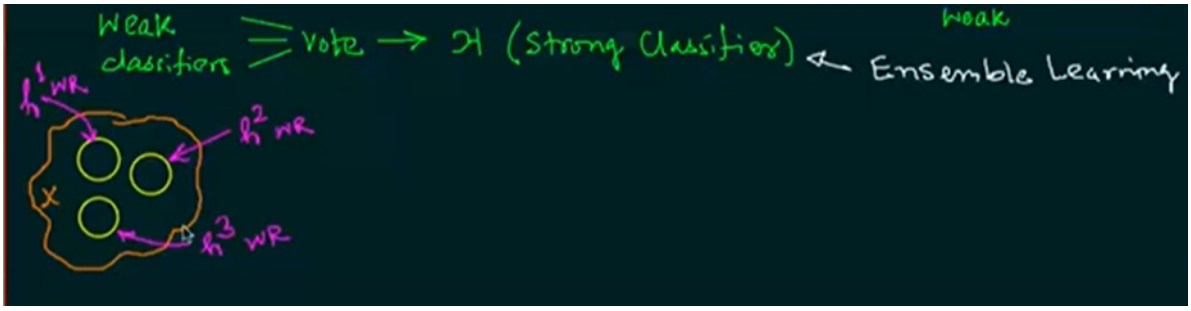

Figure 2: Hypothesis space

Now consider scenario 2: overlapping of spaces. We take a vote on the sign of the performance of the first classifier, performance of second classifier, and performance of third classifier. Each of them can take a plus (+) or a minus (-) because they are binary classifiers. It means result will be the sign of the two classifiers agreeing with each other. The shaded orange regions are the bad regions. Hence, problem will be created during voting because if two are wrong, it will give wrong result. Hence, we need to reduce such dependence among the classifiers. This will help to solve ensembling problem. The more the orange portion areas, worse is the ensembling procedure. Thus, area covered by the orange portion should not be more than any of the classifiers

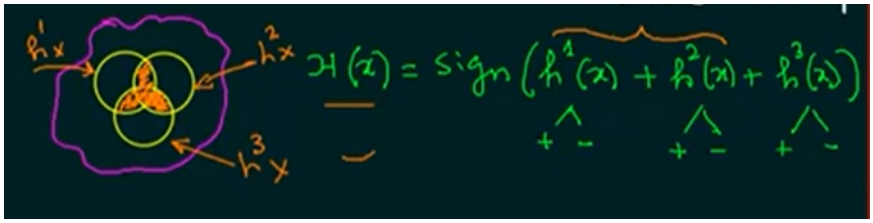

Figure 3: Overlapped space

3 Informal Description

Suppose we have labelled N data points in the supervised learning. Create a first sample dataset, D1, subset of the N datapoints. From this we get the first classifier, h1. It is a weak classifier; hence some data are wrongly classified. We make some data and some exaggeration of it, that is we will try to increase the probability of wrong points and decrease the probability of right points. Thus, with the new probability we created another dataset, D2. Third case is we start another exaggeration: h1 and h2 will differ. Since, they differ h3 choice will utmost matter and then we sample and again create a new dataset, D3. Then a voting is done on the final hypothesis, that is we take sign of the summation of h1, h2, and h3 and so on of the binary classifier. First, we assign weight α to the classifiers, called the weighted wisdom of expert crowds and then do the summation of the classifiers. Thus, the final hypothesis is the average of the majority of the weighted wisdom of expert crowds. Exaggeration basically means changing the weight of every point.

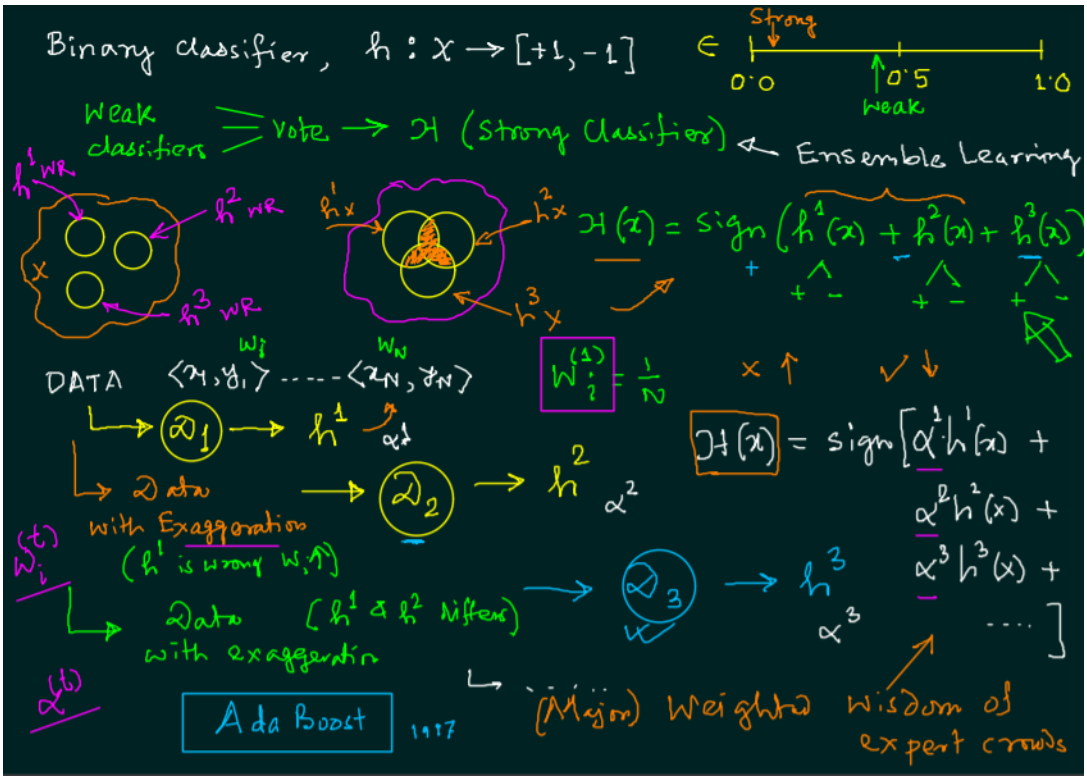

Figure 4: Final hypothesis

4 ADA Boost

We take a decision tree model. We call our model decision tree stumps: classify with one attribute, that is one side is plus (+) and another side is minus (-). Stumps can be places at different positions. Total there are 2n number of decision stumps. Each stump can create 2 classifiers.

Figure 5: Decsion Tree Stumps

Final goal is to get a Classifier C in which we can aggregate the wisdom of experts. We can build the classifiers in a recursive manner.
When we started of each point with a weight, that is, for all i from 1 to N. We start with time step 1. So, the error at time step 1 is sum of all the weights at time 1 which are wrong because binary classifier is +1 or -1. if we go generic way, at any step the error we make is wrong, that is the hypothesis is not matching with our label. Due to normalization, average of errors is taken. We assume the sum of all weights at any particular time is 1.

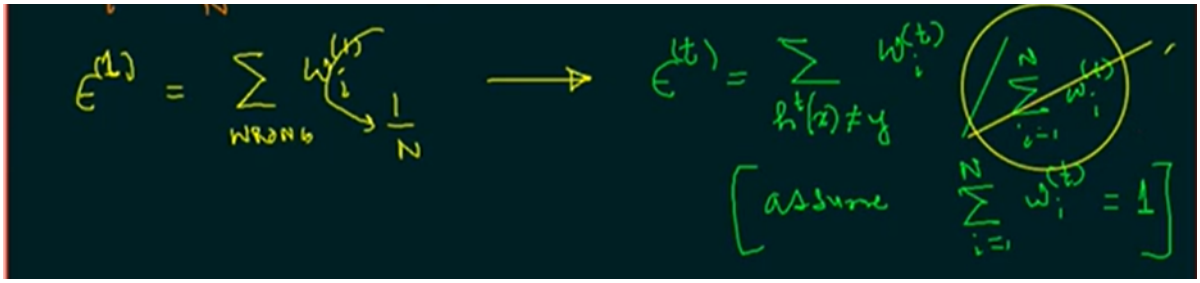

Figure 6: Average Error

4.1 ADA Boost Algorithm

Step1: Initialize all weights to 1/N. Then we have iterative version of updating w(t). We pick a hypothesis, ht which minimizes error ϵ^t. Then based on that we pick weight α^t and we update weight at t+1, that is w^{t+1} then we repeat the process of selecting a hypothesis that minimizes error. This is a repetitive process.

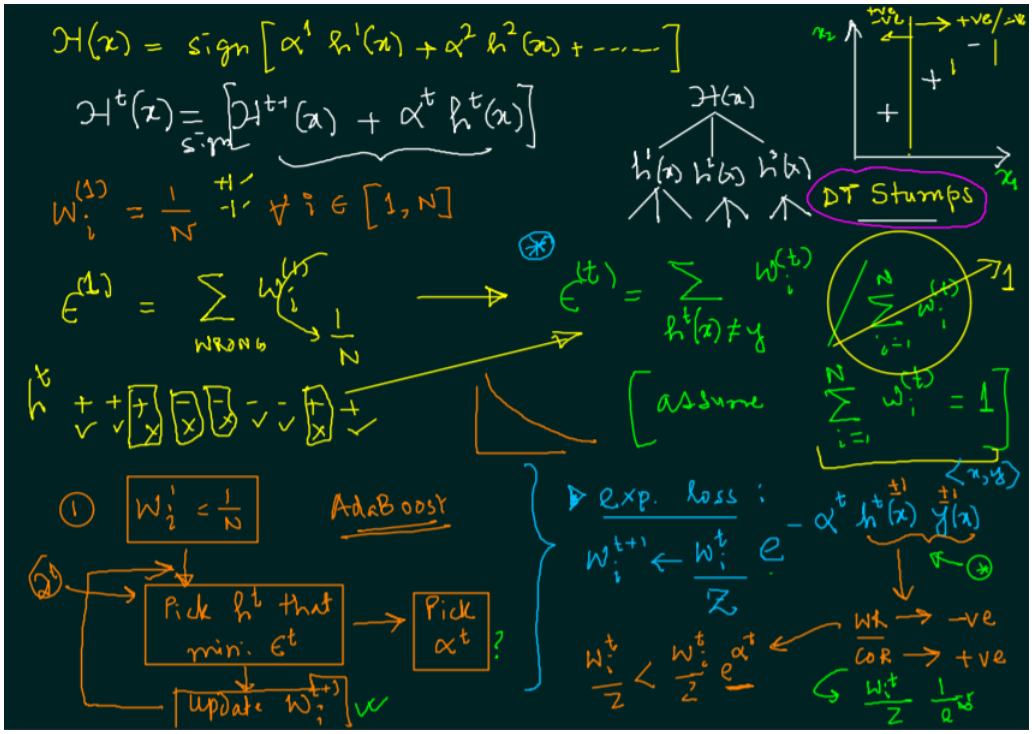

Figure 7: ADA Boost Algorithm

4.2 Mathematical Basis

We formulate the error in the form of an exponential loss. Sum of the weights should go to one. For the correct case, the sign is always plus (+). $h^t(x)$ and y(x) both will be +1. For the wrong case, sign will be always negative (-). We want to exaggerate those points which are wrongly classified. If we have a wrong classification, it will be e to the power positive α. For the correct classification it will be e to power negative. This is update or weight exaggeration.

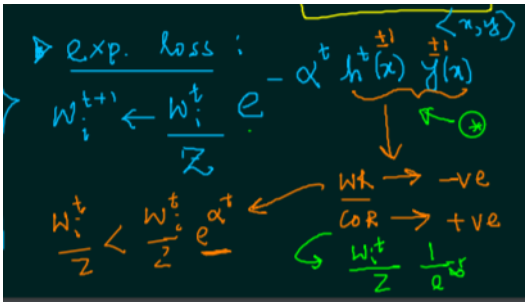

Figure 8: Exponential Loss

We need to find the error at time, t. Each of the points that are responsible for making wrong classification times whatever error we get. To get the minimum error, we differentiate the equation with respect to α^t and set it to zero. It is sum over wrong points and correct points.
Finally, we get α^t as half of natural logarithm of summation of correct points divided by summation of wrong points. Recall that summation of wrong points is ϵ^t and thus summation of correct points is 1-ϵ^t. Thus, now we are ensembling $h^t(x)$ with α^t.

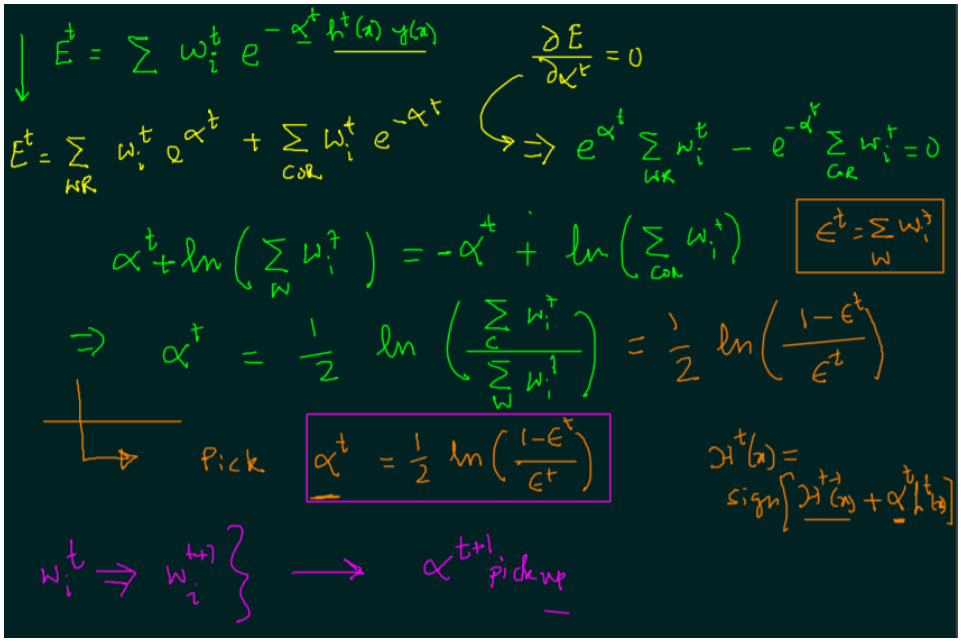

Figure 9: Weight at time step t

We have two things to consider:
1) Updating of weight from w^t to w^{t+1} which is basically with respect to exponential loss
2) What α^{t+1} to pick up
We combine the following two equations to get:

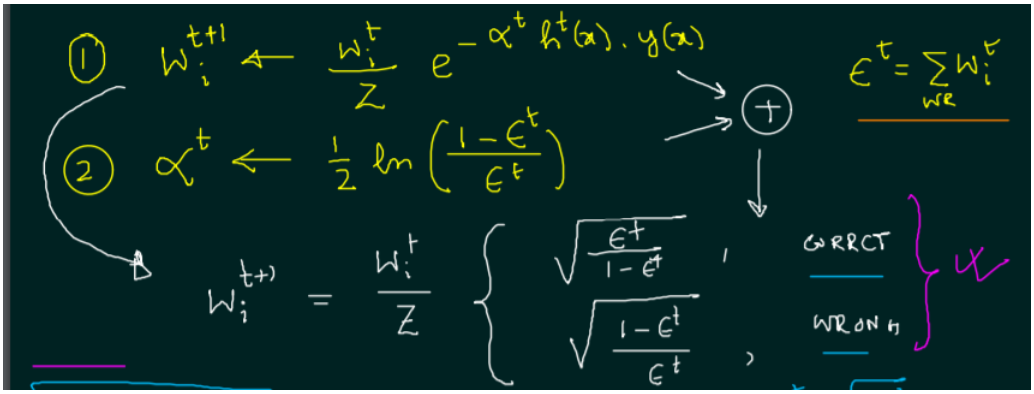

Figure 10: Combined result

4.3 Adaboost weight updation

So i^{th} weight updation at step t+1 is:

$$w_i^{t+1} \leftarrow (w_i^t/Z).exp(\alpha^{-t}.h^t(x).y(x)) \tag{1}$$

and α at time step t we calculated is:

$$\alpha^t \leftarrow 1/2 \ln(1 - \epsilon^t/\epsilon^t) \tag{2}$$

For all the points that are correctly classified,
If $y(x) = 1$ then $h(x) = 1$ or if $y(x) = -1$ then $h(x) = -1$, so

$$h^t(x).y(x) = 1 \tag{3}$$

For all the points that are incorrectly classified,
If $y(x) = 1$ then $h(x) = -1$ or if $y(x) = -1$ then $h(x) = 1$, so

$$h^t(x).y(x) = -1 \tag{4}$$

For correctly classified points, by combining the equation 1, 2, 4 and 3, we get -

$$w_i^{t+1} \leftarrow (w_i^t/Z) \begin{cases} \sqrt{\epsilon^t/1-\epsilon^t} & \text{for correctly classified points} \\ \sqrt{1-\epsilon^t/\epsilon^t} & \text{for incorrectly classified points} \end{cases} \tag{5}$$

Since the condition for weights at time t is,

$$\sum_{i=1}^{N} w_i^t = 1 \tag{6}$$

It will satisfy at time t+1 also, so combining weights for all correctly and incorrectly classified points at time t+1 from equation 5 and 6 we have-

$$\sum_{correct\_points} (w_i^t/Z)\sqrt{\epsilon^t/1-\epsilon^t} + \sum_{incorrect\_points} (w_i^t/Z)\sqrt{1-\epsilon^t/\epsilon^t} = 1 \tag{7}$$

Since $\sqrt{\epsilon^t/1-\epsilon^t}$ is not dependent on w_i^t so equation 7 reduces to-

$$\sqrt{\epsilon^t/1-\epsilon^t} \sum_{correct\_points} w_i^t + \sqrt{1-\epsilon^t/\epsilon^t} \sum_{incorrect\_points} w_i^t = Z \tag{8}$$

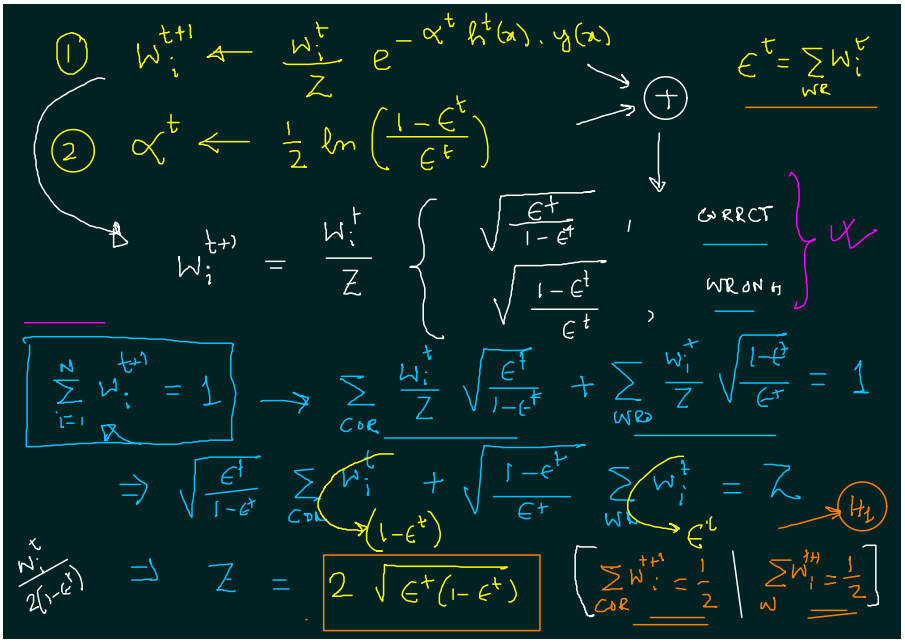

Figure 11: Handout for weight updation and Z value calculation

Since we know that :

$$\sum_{incorrect\_points} w_i^t = \epsilon^t \tag{9}$$

This gives -

$$\sum_{correct\_points} w_i^t = 1 - \epsilon^t \tag{10}$$

Combining equation 8, 9 and 10:

$$Z = 2\sqrt{\epsilon^t(1-\epsilon^t)} \tag{11}$$

In equation 5 put value of Z from equation 11 and on solving we get-

1. For correctly classified points, on putting value of Z the 5 and taking sum over all correct points, we get

$$\sum_{correct\_points} w_i^{t+1} = \sum_{correct\_points} w_i^t/(2\sqrt{\epsilon^t(1-\epsilon^t)})\sqrt{\epsilon^t/1-\epsilon^t} \tag{12}$$

   On solving,

$$\sum_{correct\_points} w_i^{t+1} = 1/(2(1-\epsilon^t)) \sum_{correct\_points} (w_i^t) \tag{13}$$

   Using the equation 10,

$$\sum_{correct\_points} w_i^{t+1} = 1/(2(1-\epsilon^t)) * (1-\epsilon^t) \tag{14}$$

   Thus for correct points we get

$$\sum_{correct\_points} w_i^{t+1} = 1/2 \tag{15}$$

2. Similarly, For incorrectly classified points, on putting value of Z the 5 and taking sum over all correct points, we get

$$\sum_{incorrect\_points} w_i^{t+1} = 1/2 \tag{16}$$

Observation: In whatever ways the weights are modified, exaggeration is such that weighted sum of incorrect points is also $1/2$.

5 Decision Stumps

Decision stumps uses single level decision tree, so it predicts on the basis of single feature at a time. If there are N points in d dimensional space, then total number of hypothesis require to capture them using decision stumps will be dN. So weights are updated dN times in Adaboost. Let the case of N points in 2-D space has the following distribution as in fig 13. The Decision stumps require $2N$ hypothesis. But here the classifier $h1$ correctly classifies point $p1$ or misclassify it. Similarly, for the classifier $h2$, $h3$, $h4$ going on right side of them we get one more misclassified point. So these all are not good classifiers, they form a combination of **bad redundant classifier**. These redundant classifier overlaps the region of bad space thus they are not independent. We do not choose all weak classifiers rather choose any one among these bad redundant classifiers. Thus the choice of classifier sharply reduces from $2N$.
We observe that number of independent hypothesis that is decision stumps practically possible $<<$ dN so weight updation is done $<<$ dN times because of redundant hypothesis.

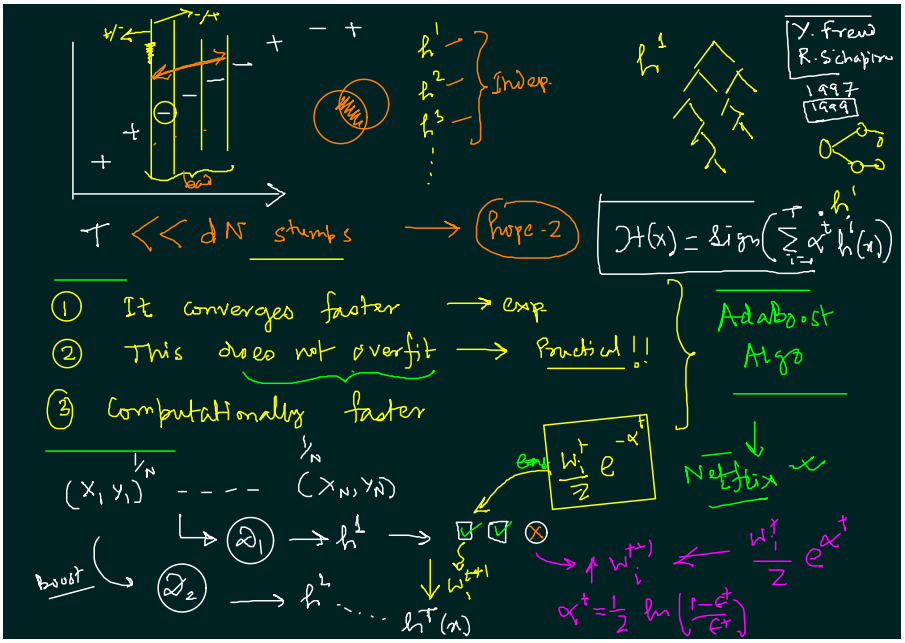

Figure 12: Handout for decision stumps

6 Adaboost advantages

The choice for exponential function makes Adaboost very practical algorithm because

1. It converges faster as it converges exponentially.
2. Practically it does not overfit.
3. It is computationally fast.

AdaBoost can be used to solve a variety of real-world problems, such as netflix price computation and it use adaboost to ensemble movie ratings, predicting customer churn and classifying the types of topics customers are talking/calling about.

7 Summary of Adaboost

1. Initialise the weights of N poitns $(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$ by $1/N$.
2. Create sample dataset D_1 and get the hypothesis h^1.

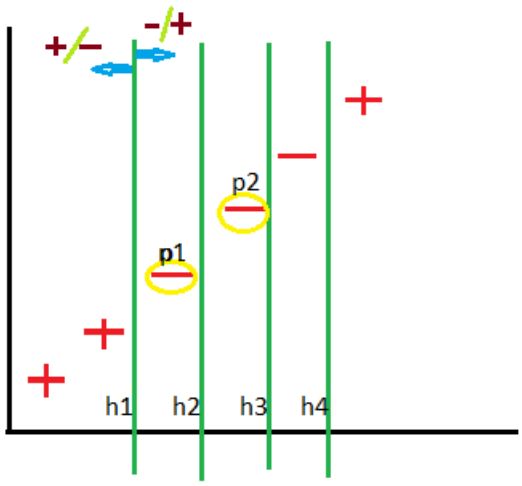

Figure 13: Decision Stumps hypothesis

3. Adaptively boost the weights of the data points. For the points that are correctly classified by the hypothesis h^1, reduce their weights by $w_i^{t+1} = (w_i^t/Z)exp(-\alpha^t)$ and increase the weights of correctly classified points by $w_i^{t+1} = (w_i^t/Z)exp(\alpha^t)$ where $\alpha^t = 1/2 \ln (1 - \epsilon^t/\epsilon^t)$.

4. Pick the next dataset D_2 and get hypothesis h^2 and repeat the steps 2 and 3 till we reach the final hypothesis $h^T(x)$ where T is the number of decision tree stumps or any weak learning algorithm.

5. Ensemble it using $H(x) = sign(\sum_{i=1}^{T} \alpha^i h^i(x))$

8 Introduction to Dimensionality Reduction

Dimensionality is a curse. Increase in single attribue (dimension) results in exponential rise in the computation, so there is need to reduce dimensionality. As shown in figure 14, the points are linearly separable in the 2-D plane. If they are projected in x_2 dimension then they are not separable but if they are projected on the x_1 dimension, they results into two separate clusters so they are also separable in x_1 dimension. Thus, at a time not all the attributes contribute in the separation method, so we can reduce dimensionality.
Kernel method enhances the dimensionality then separate the points in high dimension and then convert back to the original dimensions. Dimensionality reduction is the reverse process.
Goal of next class

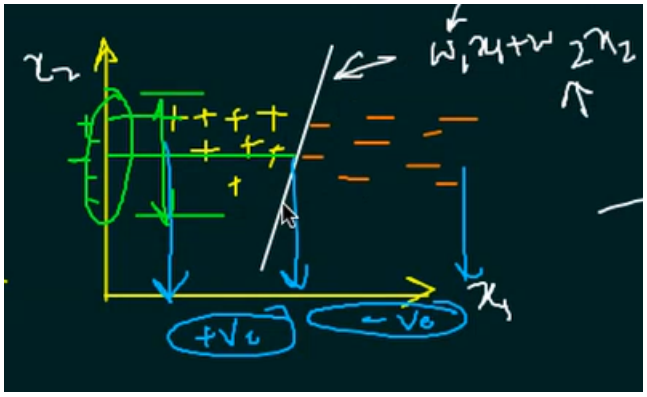

Figure 14: Diensionality reduction

1. How to choose the attributes to get reduced dimensions such that we can separate the dataset.
2. Kernel methods that enhances the dimensionality.
3. PCA (Principal Component Analysis) method that reduces the dimensionality to make machine learning algorithm easily implementable.

Contribution
Sreyasree Mandal: First 35 minutes of the lecture
Somi Vishnoi: Last 35 minutes of the lecture