Machine Learning (CS60050)
Spring 2020-2021
Instructor: Dr. Aritra Hazra

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture Date: Friday, 12^{th} March 2021
Scribed by: Vishal Gourav(20CS60R21)

March 13, 2021

Introduction

In general most of the problems we solve using machine learning comprises of data sets which has the classes along with the data i.e., the model we create is trained under the **supervised learning** paradigm. We have already seen that given a data set how we can create a *good learner* that has a high accuracy and a low in sample error.

But every learner we create may not be good. For example, a learner which is a 2-class classifier has an in sample error of 0.4 is a weak learner.

Now we will try to learn how given a set of learners which are each individually not good(high error close to 50%), we can combine them to make a good learner which is termed as **Ensemble Learning**.

Multiple Learners in Ensemble Learning

Now the next thing one would imagine is how for a given data set different learners can be created. It can be done by changing different aspects of the learning process which are as follows:-

- **Different Algorithms.** Different algorithms like *Support Vector Machine(SVM), Decision Tree Learner(DTL), Artificial Neural Network(ANN), etc.*might be used on the same data set to create different learners.
- **Different Hyper-parameters.** On a given data set, given the same algorithm we play around with the hyper-parameter values to obtain different learners. For Example, In ANN, by varying number of hidden layers, varying the Learning rate, varying the number of neurons in the hidden layer,etc.

- **Different Training Set.** Break the given data set D into smaller chunks such that $D_1 \bigcup D_2 \bigcup D_3 \bigcup .....D_n = D$ and create separate learners using the same algorithm on these smaller chunks.
- **Different Representation.** One can also use different mathematical representations of final output inside the learner to create different learners. For example, In ANN we can use sigmoid, ReLu, Step functions or various other functions in the learning phase.

Eventually by the above ways we achieve more then one learner. Now ideally we want the hypothesis for each learner and their respective errors to be independent of one another. For example, in the Figure 1 for a given data set enclosed by the orange colored boundary each hypothesis gives error for the part of the data set inside the smaller green boundaries i.e., h_1 has error e_1, h_2 has error e_2, h_3 has error e_3, h_4 has error e_4 and h_5 has error e_5.

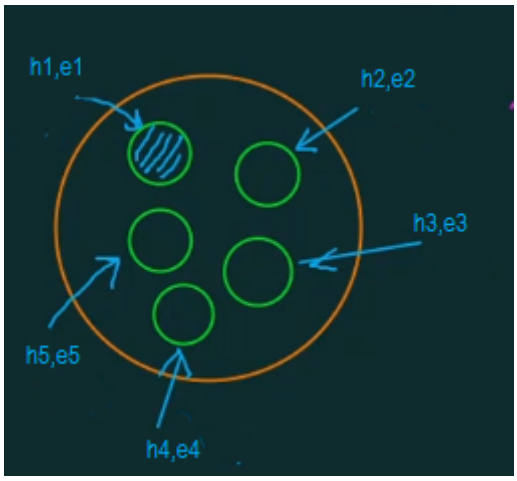

Figure 1: Ideal Ensemble Learner

What is to note here is that the data points where h_1 has error e_1 all others are accurate therefore we can take majority of all outputs by all learners. This can be done for all the other 4 hypothesis therefore resulting in the *Ensemble Learner(H)* having 100% accuracy.

But this is generally not the case. The multiple learners that are produced have some overlapping error error. For example, 3 learners on a given data set as given in Figure 2 have hypothesis h_1, h_2 and h_3 which have errors e_1, e_2 and e_3 respectively.

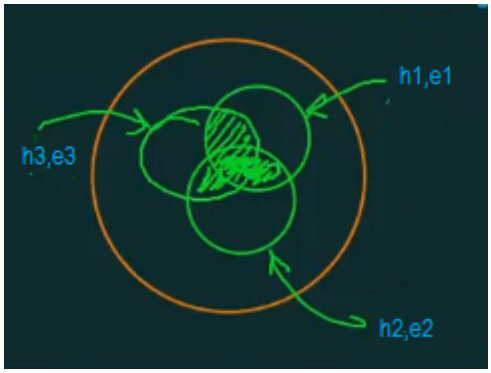

Figure 2: General Ensemble Learner

We can see there are overlapping regions in which majority voting will result in an inaccurate result. This brings down the accuracy. Our goal is to reduce this overlapping part as much as we can thus improving the performance of the ensemble learner. Thus our main goal when creating an ensemble learner is to get the constituent learners with error boundaries as **independent** from each other as possible.

Bias and Variance in Ensemble Learning

Before getting into the effect of of ensemble learning on bias and variance lets take a detour to what for a given learner bias and variance is. This can explained with the Figure 3 below.

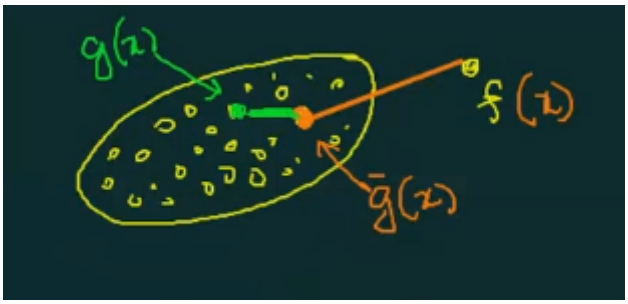

Figure 3: Bias and variance

As we see, the hypothesis(H) is yellow boundary, $f(x)$ is the target function, the orange point is the average hypothesis($g'(x)$), the difference between ($g'(x)$) and $f(x)$ is the bias, the green point $g(x)$ is what the learner has achieved and finally the difference between $g(x)$ and ($g'(x)$) is the variance.

In general more the bias, less is the variance. But we want to get to a state where the bias and variance both are as low as possible which we will try to achieve in ensemble learning.

An example, given a 2 class problem(Classes A and B), n independent learners each having accuracy=0.7 where n_1 learners classifies a test point to class A and n_2 learners classifies it to class B where $n_1 > n_2$ and $n_1 + n_2 = n$. Using binomial theorem,

$$Bin(n, r, p) = \binom{n}{k} p^r (1-p)^{n-r}$$

we get the probability of test data belonging to class A as,

$$Probability[Class(n_1) = A] = 1 - Bin(n_1, n_2, 0.7)$$

Sometimes instead of using majority as in the above example we also may use weighted majority where each learner is given a predefined weight $w_i \geq 0$ such that $\sum_{i=1}^{i=n} w_i = 1$. The weights w_i for example, may be proportional to accuracy or inversely proportional to variance for each learner. Finally the output is obtained of the ensemble learner is obtained as,

$$y = \sum_{i=1}^{i=k} w_i d_i \text{ where } w_i \geq 0$$

If all the weights w_i are the same then it effects the variance int he following ways:-

$$Var(y) = Var(\sum_j \frac{1}{k} \cdot d_j) = \frac{1}{k^2} \cdot k \cdot var(d_j) = \frac{var(d_j)}{k}$$

As we can see here the **variance of our ensemble learner decreases by a factor** k which is the number of learners we have. Thus, the bias also deceases.

Ensembling

Given the Learning model being used it may not be a straight forward task to ensemble the weak learners to make an ensemble learner. For example, in a Bayesian Learner with ensemble technique we produce classifiers $M_1, M_2, ...M_k$ then probability $P(C_i|x)$ of a particular data point x belonging to a class C_i is,

$$Prob(C_i|x) = \sum_{t=1}^{t=k} Prob(C_i|x_i, M_t) \cdot Prob(M_t) \cdot w_i$$

But getting the value of $Prob(M_t)$ as a *priori* which is the probability of how good the model M_t for the given data set. Further finding $Prob(C_i|x_iM_t)$ is computationally very heavy. To get the values faster something called the **Bayesian Model Average**(BMA) is used which uses Monte Carlo Approximation(out of scope).
Once the ensembling is done the final value can be found in the following way:-

$$y = \arg\max_{C_j \in C}(\sum_{h_i \in H} Prob(C_j|h_i) \cdot Prob(TE|h_i) \cdot P(h_i))$$

The value of y obtained above classifies given the test example, TE to a particular class.

MACHINE LEARNING (CS60050) Spring 2020-21

Instructor : Prof. Aritra Hazra

Scribe by : Suyash Tiwari (20CS60R18)

Date : 2021-03-12

Independent Learner :

Till now, the main bottleneck is the non-indepencence of outputs. For independent outputs we can write from earlier as :

$$Var(y) = \frac{Var(d_j)}{k}$$

If they're not independent and are correlated, then we can write the formula for covariance as :

$$Var(y) = \sum_j \frac{1}{K} d_j = \frac{1}{K^2}\left[\sum_j Var(d_j) + 2\sum_i \sum_j Cov(d_i, d_j)\right]$$

This formula is not as bad as it looks. The learners d_i and d_j will be independent for some i and j since, not all learners are dependent with each other. This will give a considerable amount of reduction since $Cov(d_i,d_j) = 0$ for those independent learners (or very less due to less dependency).

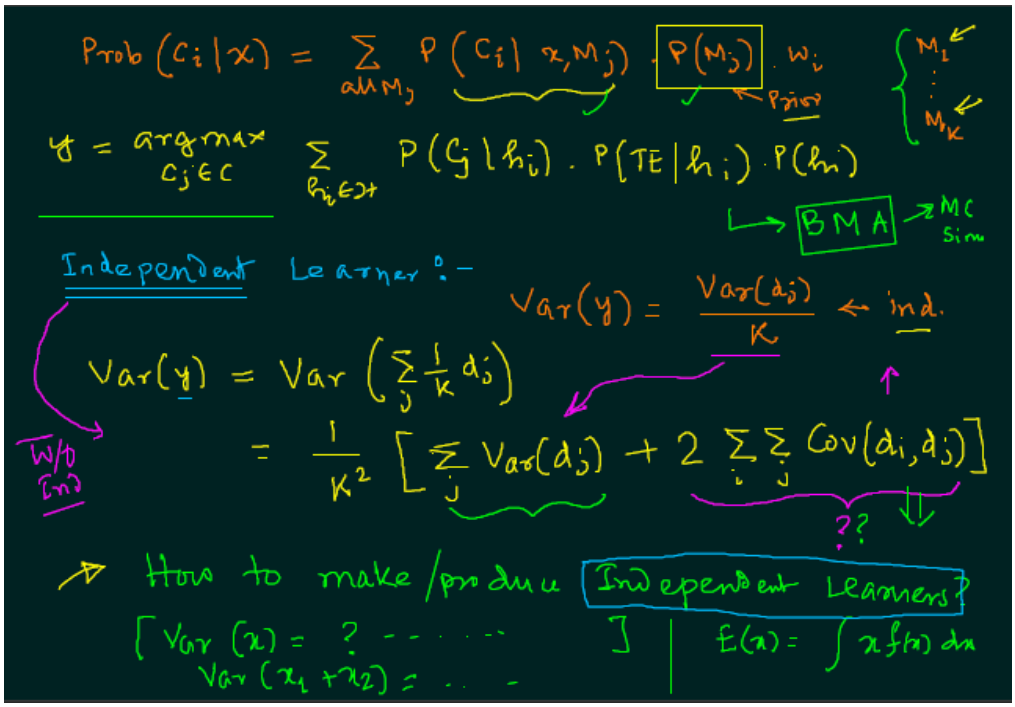

Fig : Independent Learners

How to make/produce Independent Learners?

One way is to produce datasets $\{D_1, D_2, D_3 \dots D_m\}$ from our existing dataset D. We can then use these different datasets to generate different learners $\{h_1, h_2, h_3 \dots h_m\}$ and then just combine them all and then finally produce a final h.

One approach to generate different datasets is called *Bagging*. Suppose there are m baskets, now we will take different samples randomly from D and place them in baskets with replacement. This will generate m datasets such that some of the samples may be repeated. The probability of picking up a sample in any of the basket with replacement is $(1-(1/n)^m)$. By this method, almost 63% of data will be not repeating and 37% data will be repeating. This particular method of sampling with replacement is called *Bootstrapping*.

Training Data

Data ID										
Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

Fig : Bagging Example

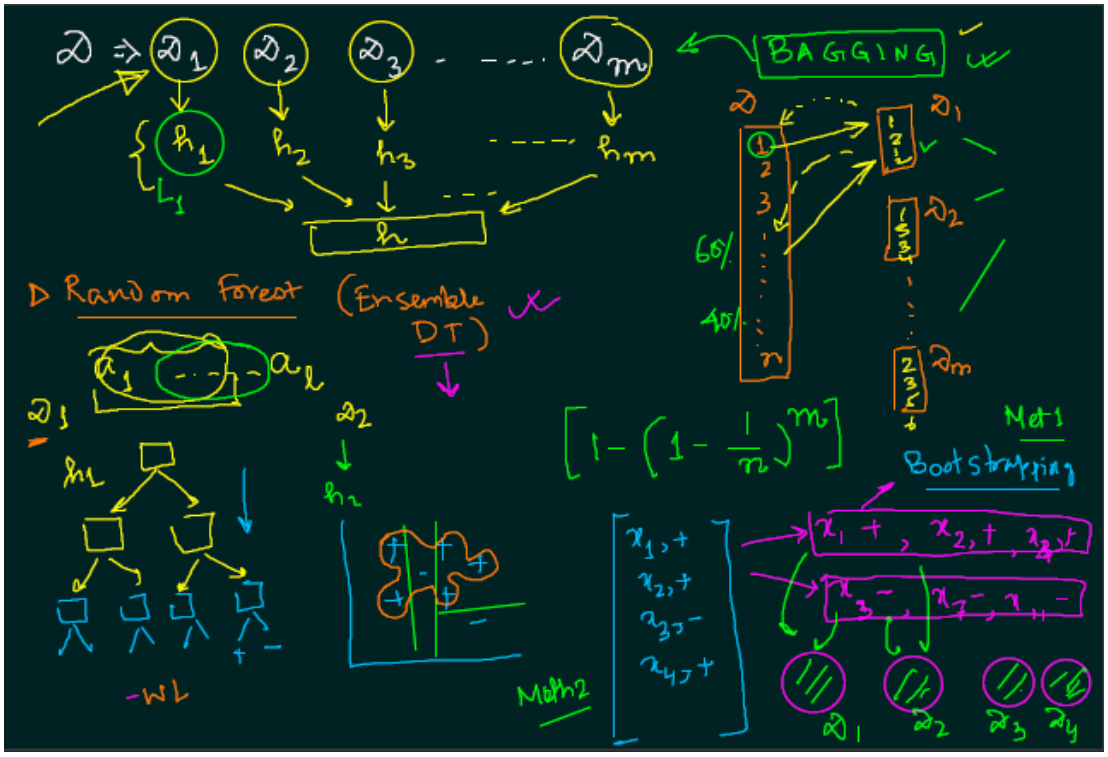

Fig : Bagging

Random Forest Classifier

The method of bootstrapping is very successfully used in Random Forest which is Ensembling of Decision Trees. It creates randomised datasets using bootstrapping then unpruned decision tree is grown for these datasets as $\{h_1, h_2, h_3 \ldots h_m\}$. However, one more kind of randomness is introduced in the datasets created. Suppose the attributes are $\{a_1, a_2, \ldots, a_l\}$, then when decision tree is created each node is split based on the random subset of these l attributes instead of all attributes. Refer again to the figure as before given below :

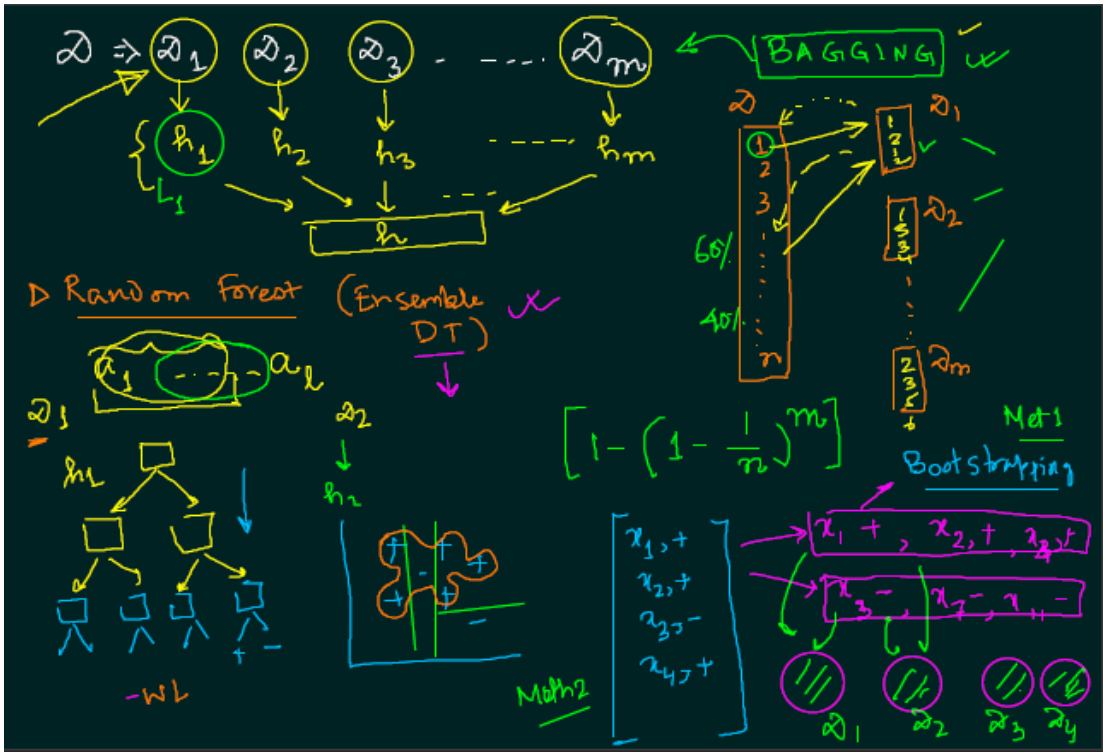

Fig : Random Forest Classifier

Boosting :

This is much more efficient than Bagging. Suppose we have a dataset of N points :

- Initially, each of the data points is given same probability (weights) of 1/N to be selected.
- Now, one dataset D_1 is created and a hypotheses h_1 is created using a learning algorithm.
- For all the data points misclassified by h_1 the weights are increased and for correctly classified points it's decreased such that $\sum w_i = 1$ where w_i is the weight of data point i.
- Again a new dataset D_2 is created but it will contain more misclassified points since probability (weights) of those points is increased.

- With D_2 again a learning algorithm is run to create hypotheses h_2. Again the weights of misclassified points is increased and correctly classified points is decreased.
- Now, D_3 is generated and this process goes on.

This produces very nice hypotheses and final classifier combines the votes of each individual classifier. The votes are combined by assigning weights to classifiers based on their accuracy.

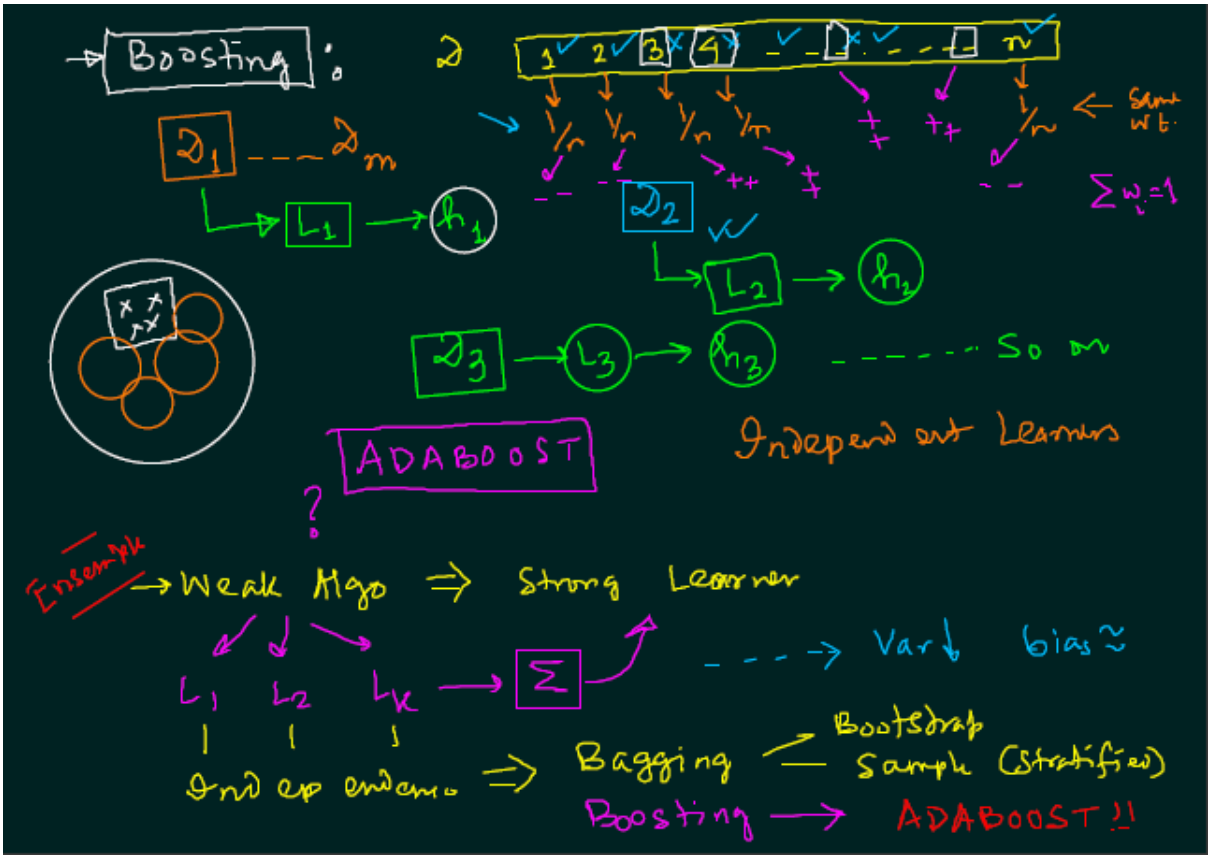

Fig : Boosting

In the next class, we will discuss *AdaBoost* which is the most popular ensembling technique.

Summary :

- We have weak algorithm and we need to produce strong learner.
- Different weak learners are produced from this weak algorithm and an ensemble is taken to produce a strong learner.
- In this process, *variance* decreases but *bias* remains almost same but the main challenge is independence of learners.
- For this, we tried to do *Bagging* which is either *Bootstrap* or *Stratified Sample* and learned philosophy behind *Boosting* whose main method is *AdaBoost.*