Implementation assignment 3:

Behnam Saeedi, Kamilla Aslami, Mostafa Estaji
CS534: Machine Learning
Due Nov 30th 11:59pm
Fall 2018

CONTENTS

1 INTRODUCTION

In this assignment we are aiming to implement 3 different learning algorithms. The first methods is the decision trees and the next two build on top of this concept. In section 1 we come up with all data structures and learning methods and functions needed in order to successfully train a decision tree. Then we optimized the code to run fairly quickly. In section 2 we used the said tree structure and learning algorithm and modified it to produce a random forest learning model. Finally in section 3 we use the tree structure in order to create an Adaptive boosting model (AdaBoost). In this document we discuss different characteristics of such learning algorithms backed by data and plots. This write up is accompanied with the code for the 3 learning procedures and a full and shortened log of the output of the program. further more, we have included a readme file for instructions on how to execute the program for grading and testing purposes. For your convenience a Makefile is also included to assist in this procedure.

2 PART 1

2.1 A

The tree was produced. This was done through various forms of data structures and fast computation tricks that Numpy provides. Overall it takes about 230 seconds for our tree to be trained at a depth of 20 with a training accuracy of 1.0 and validation accuracy of 0.9202.

2.2 B

We can see the resulted graph in the follow figure:

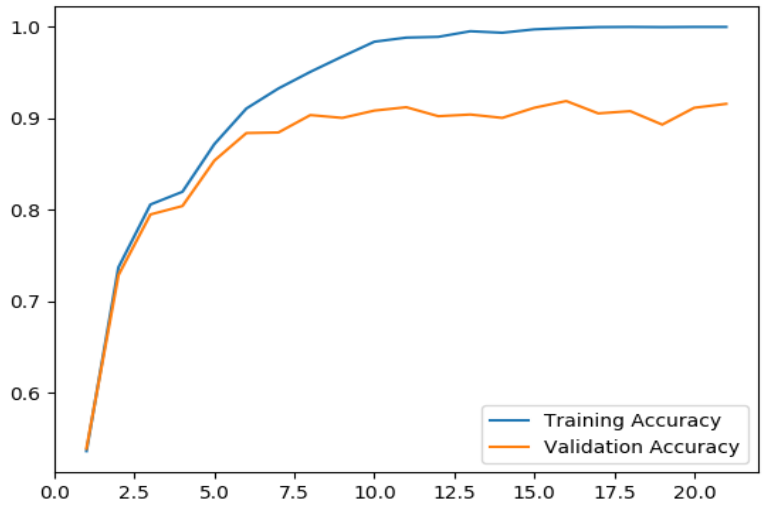

Figure 1. This is an extended version of the required graph with no logarithmic scaling. X-axis is the number of depth for the training and Y-Axis is the accuracy value. Please note that the process of over fitting, actually starts slightly before depth 16. Please note producing this graph took about 4 hours

2.3 C

There is a clear trend that as the depth of the tree increases, both train and validation accuracy start improving. This is clear because the more depth we add the more complex our model gets. For training accuracy we can keep training after we reach this 100% accuracy value, However, further training after that actually decrease our validation accuracy. Our tree manages to reach the 100% training accuracy at around depth of 17. (At depth of 16 we have a training accuracy of 0.99959 which is very close, but we truly achieve the 1.0 accuracy at depth 17).

2.4 D

Based on our results depth 16 (2^3) gives us the best value for validation accuracy.

3 PART 2

3.1 A

One of the challenges with this assignment was creating a very efficient program that is capable of handling large data sets and learn them in a very fast manner. In order to do so we had to incorporate many mathematical tricks in order to simplify the computation. For this section we decided to modify our already existing tree structure in order to maintain a high performance. The tree was developed to accommodate for the n, m and d values. These values could be set globally or passed directly to the training algorithm. The advantage of such approach is that the training algorithm does not care whether we are training a single large tree or a random forest. It just simply works.

3.2 B

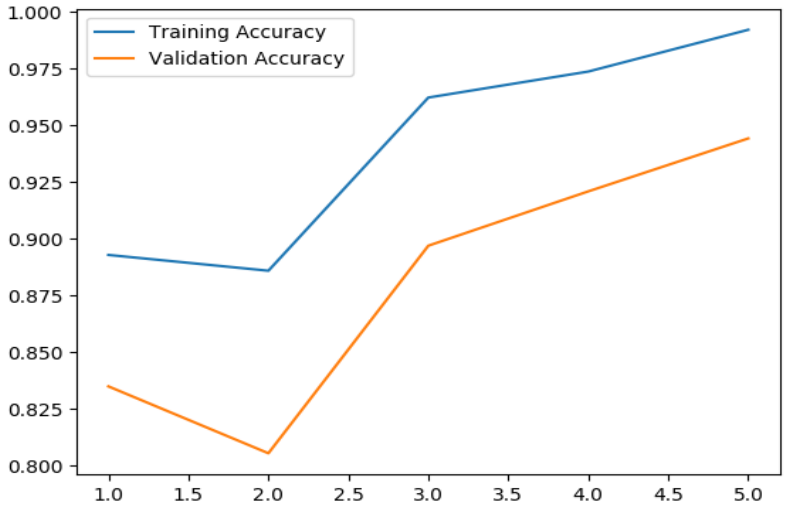

Figure 2. $d = 9$, $m = 10$ and $n \in [1, 2, 5, 10, 25]$. Please note matplotlib tends to break the x label with number of items due to formatting issues. Here is the corresponding of the x labels with number of trees: $1.0 \rightarrow 1$, $2.0 \rightarrow 2$, $3.0 \rightarrow 5$, $4.0 \rightarrow 10$, $5.0 \rightarrow 25$ trees respectively.

In this figure we computed and trained the random forest with corresponding number of trees, 10 features and depth of 9. As the number of trees in the forest increase, we can see a trend on both training and validation accuracy increasing.

3.3 C

As we mentioned earlier it improves the outcome of our accuracy. The reason to this is that random features end up learning similar trends from different variables. This generally leads to a larger majority when we are computing a prediction. The more trees we have the closer we get to a tree that has trained all of the features at a higher depth. When the number of trees are small there is a good chance that the most of our trees haven't learned a specific trend. Hence why our accuracy increases as the number of trees increase.

3.4 D

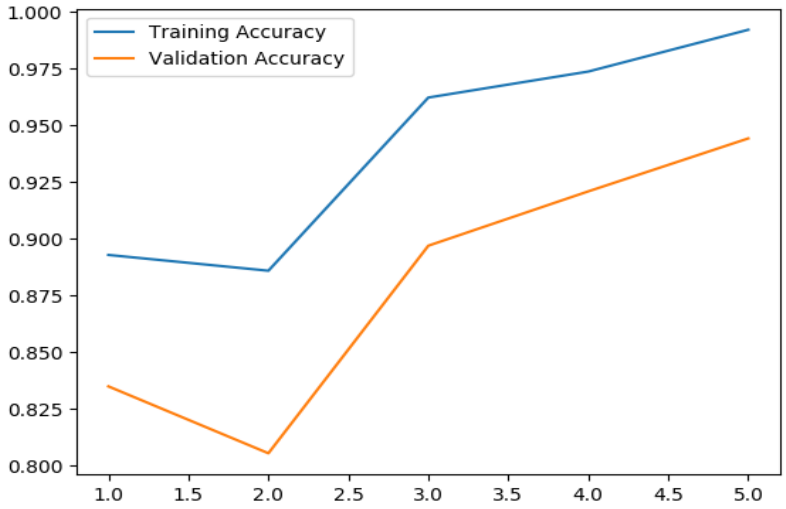

Figure 3. $d = 9$, $m = 10$ and $n \in [1, 2, 5, 10, 25]$.

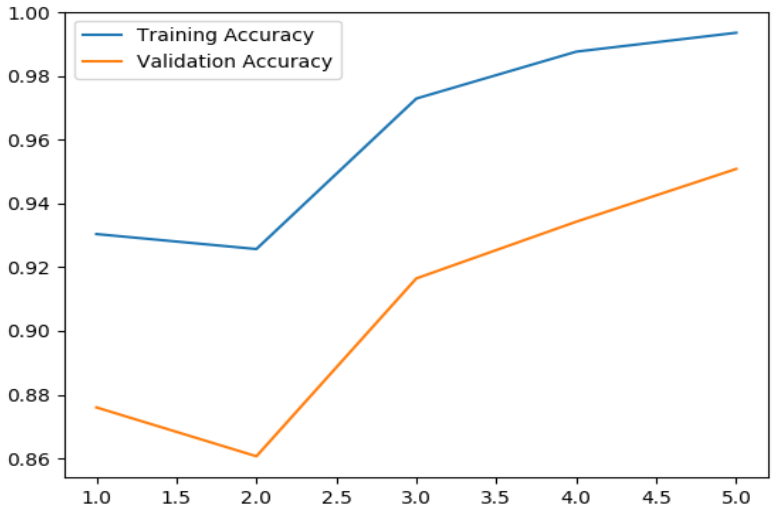

Figure 4. $d = 9$, $m = 20$ and $n \in [1, 2, 5, 10, 25]$.

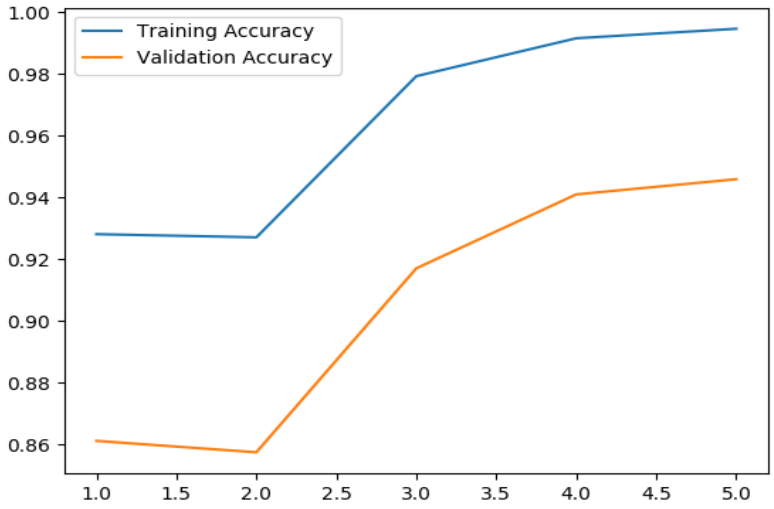

Figure 5. from top left to bottom right: $d = 9$, $m = 30$ and $n \in [1, 2, 5, 10, 25]$.

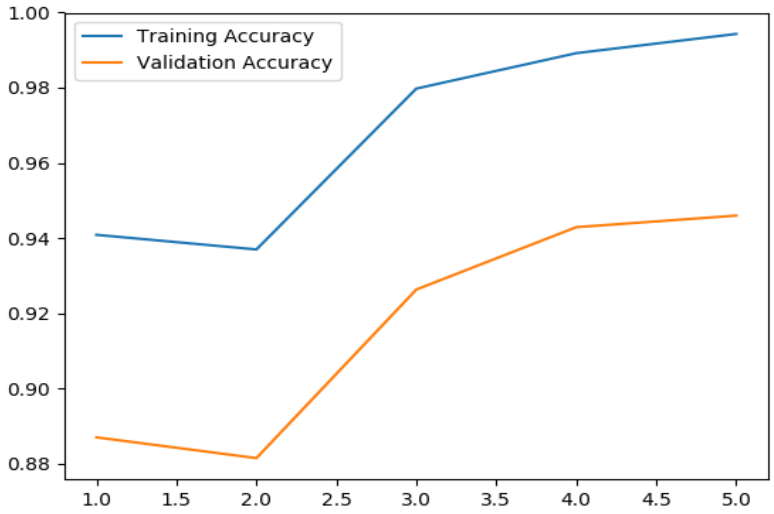

Figure 6. from top left to bottom right: $d = 9$, $m = 40$ and $n \in [1, 2, 5, 10, 25]$.

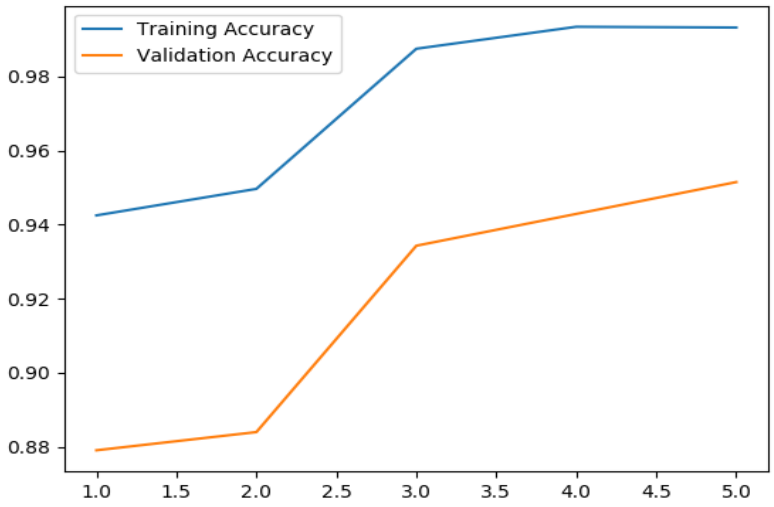

Figure 7. from top left to bottom right: $d = 9$, $m = 50$ and $n \in [1, 2, 5, 10, 25]$.

Each one of these plots were produced for values $n \in [1, 2, 5, 10, 25]$ and took over 5 hours to train and produce. As the number of features for each tree increases, we can see that there is a general trend that our model becomes more accurate in both training and validation. The reason to this is that we end up with more overlapping features and the majority of our trees end up agreeing with each other more. If a trend is truly related to the true label, its effect becomes amplified through the forest.

Lets focus increase on impact of $m \in [10, 20, 30, 40, 50]$. As the number of features provided to the trees increase we can see that our accuracy increases as well. This is visible in the above figures as well (please pay attention to Y scale of each graph that gets narrower and narrower in each iteration, meaning our values are getting closer and closer to 1. This is an unfortunate artifact of matplotlib that automatically scales the axis and makes it hard to notice differences). The reason to this improvement could be due to the fact that more trees are going to be considering a certain feature now since with larger number of features available out of 100, they tend to overlap more. This means if feature x_l truly affects the outcome of our learning in a positive way, this will be present in more and more trees causing our final prediction to be moved more and more towards such prediction involving said specific feature (x_l).

4 PART 3

4.1 A

In this section we used our developed decision tree and made it compatible with the new specification for part 3. This required us to create a new file to avoid conflict with other sections.

4.2 B

The AdaBoost algorithm was implemented. This had a few challenges since we were originally unsure on certain steps such as computing ϵ_l since it was not explained in the notes very well. But eventually we managed to implement it and train batches of trees.

4.3 C

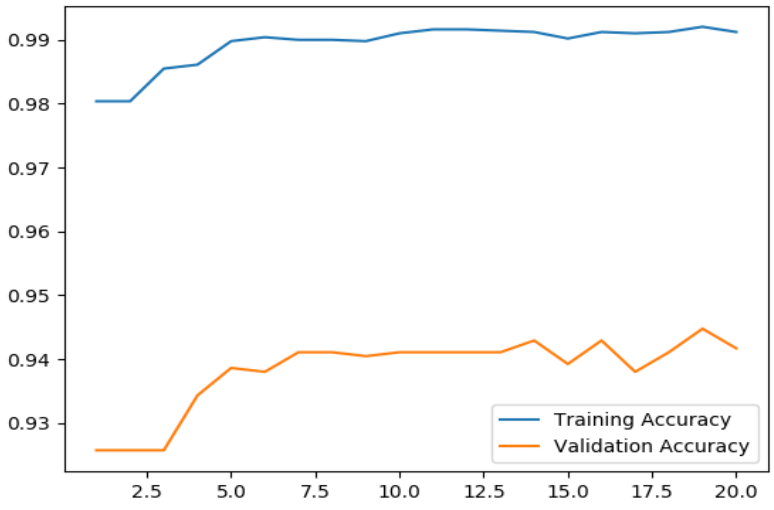

Figure 8. The figure above is representing our training and validation accuracy's for AdaBoost algorithm with 1 to 20 trees ($L \in [0, ..., 20]$).

The plot above covers the Training and Validation accuracy of models involving $L \in [1, ..., 20]$ trees. This was more time consuming but helped us to understand the behavior of AdaBoost better and closer.

4.4 D

First important thing to notice is that there is not a significant improvement in training and validation accuracy for the first batch of 2 trees. The reason to this is because the errors are not boosted (amplified) enough to make any significant difference. The distribution in first tree is very much meaningless since no other factor is contributing to the decision. With addition of the second tree we are using the distribution produced by first tree which might not have very significant differences from the first one. but after that we can observe a steady improvement in our trend of Training and Validation accuracy. Another interesting behavior we are observing is that the improvement in accuracy continues despite the fact that we are increasing the number of trees. in other words, we are not observing a serious over-fitting issue all the way to 19_{th} tree that is being trained. Even at 20_{th} tree added the disparity and inconsistency in behavior of validation accuracy could be due to number of reasons other than over fitting.

As it can be seen in Figure 8, the accuracy increases as the number of hypotheses grow. From a certain point on, further increase in the number of the hypotheses does not remarkably have effect on the value of the accuracy. TO elaborate, in Adaboost, as each h (i.e. hypothesis is learned) the labels of the training set get more meaningful distribution, D. The reason is that the distribution updates follow different exponential curves for wrong and correct predictions as outlined in the algorithm. At the same time, each hypothesis will be weighted based on the error value in the validation. The higher the error, the lower the contribution. Finally, the predictions get aggregated over all weighted hypothesis to predict the most accurate labels possible.

4.5 Conclusion

In conclusion, the Decision tree is an extremely simple, reliable and flexible model for learning. There are a few issues which we noticed through out this assignment. For example, decision trees do not scale well with number of features and have a high workload. Producing some of these graphs took a very long time.