UNIVERSITY
OF MIAMI

ECE548
Group Project 1
Perceptron Learning Algorithm

Khaled Qurainis
Lia Murphy
Julio Ojalvo

Introduction:

Our goal with this project is to experiment with and further understand the perceptron learning algorithm. We want to take what we learned in class and apply it in a hands-on manner that gives us insight into the details of this algorithm including its advantages and disadvantages. Through constructing this algorithm and testing it we hope to further improve our understanding of all the elements that make up the PLA algorithm. This includes the impact of non-linearly separable data, the optimal epochs and learning rates, and the cases that lead to an overfit model.

The linear separability of the datasets is an important note in this experiment as it determines whether our algorithm will ever reach one hundred percent accuracy. We want to find out how long it will take for our algorithm to reach perfect accuracy given that the data is linearly separable. We also intend to experiment with different numbers of epochs to figure out when it is that we could be certain that a dataset is not linearly separable and there is no combination of epochs and learning rates that could ever get the algorithm to one hundred percent accuracy.

Another important topic in our experiment is how the PLA algorithm works on different kinds of datasets. We will be testing the performance of the algorithm on datasets with different numbers of attributes, kinds of attributes, and number of instances. We strongly suspect that the lower the number of attributes a dataset has, the quicker this algorithm will reach perfect accuracy since there are less weights that need to be adjusted to their perfect values before the algorithm will provide perfect results. It also seems that the fewer variables there are to account for, the more likely that the data points are linearly separable, giving us a desirable result. We also suspect that the more instances a dataset has the better the algorithm will perform but are curious to find out if there are problems with datasets that might have too many data points and how it could affect our results.

Experimental Design:

Our version of the perceptron learning algorithm was implemented in C++. We created a class to hold all model data and functions. There are three main functions that implement the core of the algorithm.

These functions include:

1. hypothesis() -> returns a positive or negative value for the class distinction.
2. updateWeights() -> iterates through the epochs and updates the weights' values depending on if the example is positive or negative.
3. classifyData() -> iterates through all data points and determines class based on the hypothesis value.

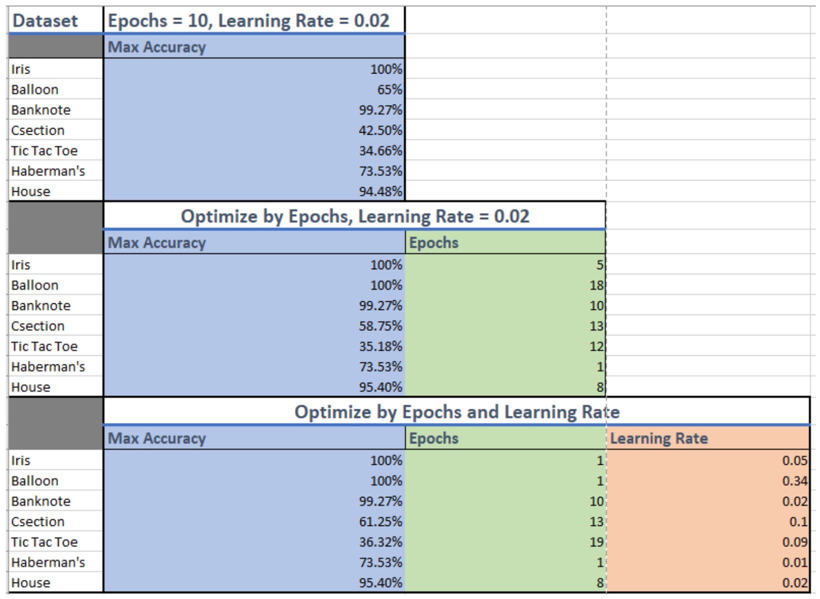
We used vectors as the primary data structure to hold all attributes, weights, and class data. As many of our datasets contained more than one attribute, we used a vector of vectors of doubles to hold attribute data. A difficulty we encountered was matching the size of the attributes to weights. This was overcome by the use of vectors, as we were able to dynamically run the model using datasets of various sizes and attributes.

We tested eight different datasets on our model (all from the UCI machine learning repository): cesarean section, banknote, iris, balloons, tic-tac-toe, haberman's survival, and political house. The model was run using a set learning rate of 0.02 and 10 epochs to determine the accuracy of the classification. We also ran an optimization function with a set learning rate of 0.02 to find the minimum epochs to reach the maximum accuracy. Finally we ran the optimization function on both the epochs and learning rate to find the optimal combination of the two.

Our code is available on github at:
https://github.com/lia-murphy564/ECE548-Perceptron-Learning-Algorithm

While ambitious, we learnt our lesson: C++ was a difficult language to implement this model in. Python would have been better suited for this project due to the ease of reading in data. At least we tested our C++ skills!

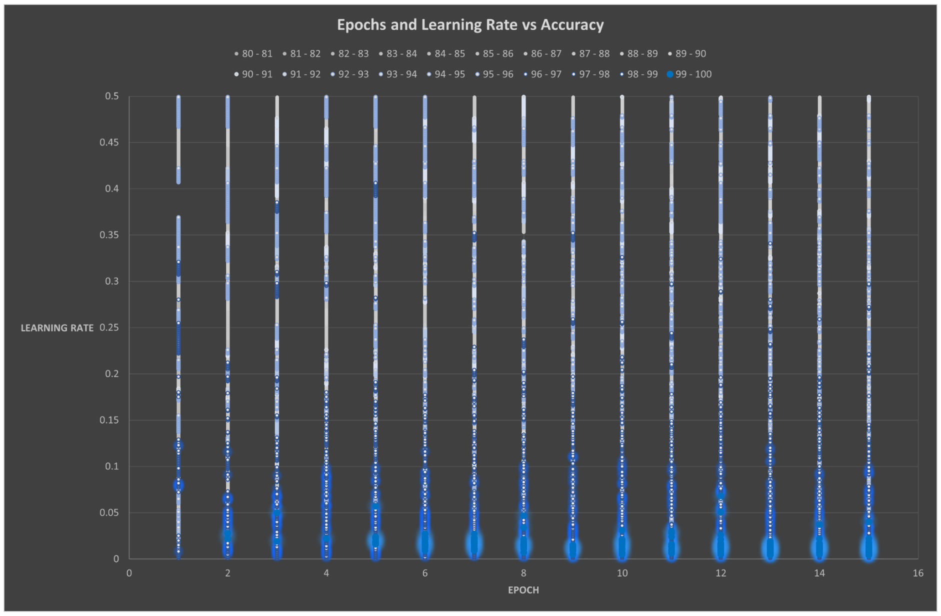
Presentation of The Results:



Dataset	Epochs = 10, Learning Rate = 0.02		
	Max Accuracy		
Iris	100%		
Balloon	65%		
Banknote	99.27%		
Csection	42.50%		
Tic Tac Toe	34.66%		
Haberman's	73.53%		
House	94.48%		
	Optimize by Epochs, Learning Rate = 0.02		
	Max Accuracy	Epochs	
Iris	100%	5	
Balloon	100%	18	
Banknote	99.27%	10	
Csection	58.75%	13	
Tic Tac Toe	35.18%	12	
Haberman's	73.53%	1	
House	95.40%	8	
	Optimize by Epochs and Learning Rate		
	Max Accuracy	Epochs	Learning Rate
Iris	100%	1	0.05
Balloon	100%	1	0.34
Banknote	99.27%	10	0.02
Csection	61.25%	13	0.1
Tic Tac Toe	36.32%	19	0.09
Haberman's	73.53%	1	0.01
House	95.40%	8	0.02

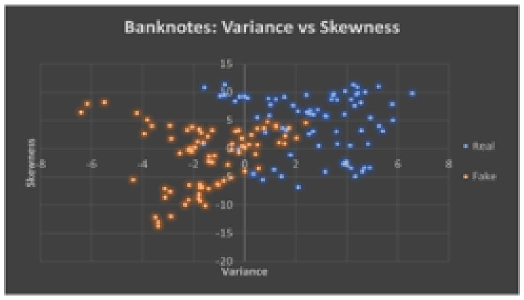
#1

Above are the results of the three tests of model runs: set epochs and learning rate, optimized epochs only, and optimized epochs and learning rate. We can see that optimizing both the epochs and the learning rate leads to better performance and accuracy of the classifier. There is a clear improvement with linearly separable datasets such as the balloon dataset, where we see the model does not reach 100% accuracy before its optimal point. We also see an improvement in the accuracy for datasets that are non-linearly separable, as is evident in the haberman's survival dataset.

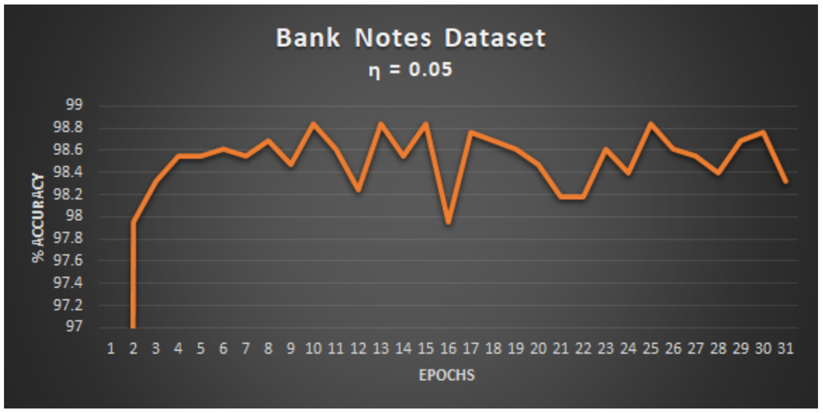


#2
A point of concern with the perceptron learning algorithm is the question of choosing a learning rate. We ran two experiments to answer this question; one with the accuracy at one particular epoch and one at every epoch.

When plotting the accuracy against both the learning rate and epochs, we notice that the most accurate combinations are below 0.1 learning rate and after 4-5 epochs. We can therefore say that with this particular dataset, learning rate has a more drastic effect on the algorithm's accuracy and error. Even after only two epochs, the accuracy is greatly improved compared to the same magnitude distance between learning rates. It is also worth noting that after the first appearance of high accuracy combinations (in this case at epochs=5, η=0.02), future epochs become redundant. This may lead to the model overfitting the data.



#3
The banknotes database is an example of non-linearly separable data. In the above graph, two attributes are plotted with the class distinction. We can see that no matter how one draws a line separating the two classes, there can never be a line that leads to 100% classification.



#4

The above graph shows the accuracy of our algorithm over the course of 30 epochs. The 1st epoch represents the accuracy of our "guesses" of the classification before any of the weights are updated. The value at 1 is 44% which makes sense since the weights get initialized to 0, meaning at first it will always predict the class to be positive and the dataset consists of 44% positive examples and 56% negative ones. The peak appears at the 9th epoch and everything after that is just overfitting.

Discussion:

When it came down to the result after all the steps we went through to get the perceptron algorithm work with our datasets, we had attempts to use a variety of data that didn't end up working. This was due to the fact that it is highly difficult to find linearly separable data. Which is why when we ended up choosing the dataset and testing it, most that we chose form the UCI repository never came to reach 100% accuracy. After trial and error, we came to the conclusion that only the Iris and Balloons datasets could be predicted with 100% accuracy. Additionally, we managed to get a high accuracy on the banknotes dataset with 98%, the house votes dataset with 95%, the wilt dataset with 98%, and the skin segmentation database with 98%. Although they're not accurate as the first two mentioned, they still had a very high level of accuracy. According to that, it was really important to choose the right linearly separable data to get to 100% accuracy working with the perceptron algorithm, otherwise if it's not it is impossible to reach a 100% no matter what we do and change.

Searching for the right datasets for this algorithm revealed a lot about its optimal function and its major shortcomings and strengths. For one, we realized very early on that it would be very obvious when a dataset is linearly separable since the algorithm was very good at getting to 100% accuracy when it was.On the two datasets that we achieved this, we reached 100% accuracy within 3 epochs, meaning that as long as the data is linearly separable, this algorithm works great and is very efficient. The major issue we had with it, on the other hand, was that linearly separable datasets are not as easy to come by as we imagined. We used 11 different datasets and in the end only two came out to be linearly separable. This brings up an interesting note that this algorithm may get close but will not reach perfect accuracy on a vast majority of the datasets available but on the few that are linearly separable, this algorithm is incredibly effective.

This algorithm also seems to be very susceptible to overfitting. In graph #4, we see that after the algorithm reaches peak efficiency at the 9th epoch and after that the values are only overfitting. We used an optimization function to run through a list of combinations of epochs and learning rates to be able to find the highest accuracy possible and found that peak efficiency was usually found very early on in the epochs, suggesting that it doesn't take long for the algorithm to be overfitted to the data. It appears that once the algorithm reaches peak efficiency it will take those outlier data points that are still misclassified and use those to readjust the current linear expression. This moves the threshold away from its optimal position until there are enough non-outliers outside of the threshold to shift the line back to the optimal position. This process repeats over and over again as demonstrated by graph #4.

Another shortcoming of this algorithm we noticed is that it is very sensitive to outliers. In any of these datasets, it only takes one outlier to throw off the algorithm. It could be highly effective but as long as the one outlier exists the algorithm will never achieve perfect accuracy. This is a similar theme with the issue of linear separability. It's either extremely effective or ineffective, as soon as one outlier exists the whole algorithm is thrown off but given that there are none, it performs exceedingly well.

Conclusion:

In conclusion, the ultimate aim was to understand the perceptron learning algorithm. What is important in all of this is applying what we learned and implementing it into our project. Doing this as a group gave us more learning of the algorithm itself and how it works with the dataset. Moreover, it is very important to know if the dataset that had been chosen was linearly separable or not, because that will affect the accuracy of the algorithm itself. Our implementation of the algorithm was used through C++ we gathered all data and graphs and had different functions for the center of the algorithm. All in all, we attempted several different types of data to be implemented into our project. Various data sets ended up not working. Throughout our trials of the various data sets, we came to the point that it is very hard to locate linearly separable data. After much trial and error overall we came to realize that both Iris and Balloons data set was in our favor with its trusted high accuracy. When it comes to coming to the right choice for algorithms, there comes a point that there are gonna be the pros and cons. Accuracy is highly important. Looking throughout the graphs, the changes in algorithm peaks can be seen as well as its susceptibility. Narrowing down choices in regards to the usage in the right datasets for the algorithm displayed much on its select function. As well as its great deal of shortcomings and power. Moreover, an optimization function had been utilized to run past different Combinations of epochs in order to locate the most most trusted accuracy and resulted that peak would typically occur in the beginning of these epochs. This indicates that not much time is utilized while the algorithm is overfitting data. Thus, it is highly crucial when you are choosing the dataset that you want to run it through the perceptron algorithm, you choose the linearly separable data that would work great on your algorithm. On the other hand, as the number of attributes increase, the data itself might be affected by being less linearly separable.

References:

fIsher. (1936). UCI machine Learning Repository: Iris data set. Retrieved September 27, 2021, from https://archive.ics.uci.edu/ml/datasets/Iris.

Pazzani, M. (n.d.). *Balloons Data Set*. UCI machine Learning REPOSITORY: BALLOONS data set. Retrieved September 27, 2021, from https://archive.ics.uci.edu/ml/datasets/Balloons.

banknote authentication Data Set. UCI machine Learning Repository: BANKNOTE authentication data set. (n.d.). Retrieved September 27, 2021, from https://archive.ics.uci.edu/ml/datasets/banknote+authentication.