Project 1: Perceptron Learning

Steven M. Hernandez
Department of Computer Science
Virginia Commonwealth University
Richmond, VA USA
hernandezsm@vcu.edu

***Abstract*— Perceptron learning provide a method of exacting classification within a finite amount of time provided the data is linear separable. On the other hand, in cases of non-linear separable datasets, learning occurs infinitely by the standard process of perceptron learning. This report provides details into this case as well as effects of different learning rates, outliers and finally the application of cross-validation is considered.**

I. INTRODUCTION

Perceptron learning provides a simple method of classification for linearly separable datasets in some finite time. In non-linearly separable datasets, a perceptron is said to learn infinitely, never converging on good enough weights. This report explores the use of perceptron learning as well as the effects of different parameters on learning such as learning rates, outliers as well as cross-validation.

II. BATCH TRAINING

To begin this exploration, we develop a random set of data with 20 normally distributed positive class data points centered at (0, 0) with standard deviation 2, and 10 normally distributed negative class data points centered at (5, 5) again with standard deviation 2. Figure 1 presents these data points created with MATLAB seed 1. Training begins with a weight vector $w = [0\ 0\ 0]^T$ following the algorithm defined in [1] Box 3.1. After 12 epochs of learning with learning rate $\eta = 0.1$, the perceptron lands on the weight vector $w = [-0.5954\ \ -0.7314\ 3.1000]^T$ for the linearly-separable dataset.

A. *Effects of Learning Rate*

Learning rate η effects the ability of the perceptron to find accurate weights. Figure 2 shows the effect of number of epochs required with this dataset when η is set to the following values: [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4] and initial weight $w = [1\ 1\ 1]^T$. As we can see, very small values for η such as 1e-4, the number of epochs increases very high. In this case, the learning rate is too small, the perceptron does not learn enough to reach its optimal placement. On the other hand, $\eta = 1e-2$ takes the least amount of time only a few epochs in this case and thus is the best learning rate for this dataset for the d values of η used. Large values of η take longer than 1e-2 because each point of data changes the given w by too much thus skipping over the optimal minimum error.

III. OUTLIERS

To review the effects of outliers on perceptron learning, an outlier negative-class data point was added to (20, 20). Again, in this case with $\eta = 1$ and $w = [1\ 1\ 1]^T$, learning takes 12 epochs to reach the final weight vector for both cases with and without the outlier. These cases do however result in slightly different final weights. For perceptron learning without outlier, $w = [-5.9543\ \ -7.3137\ 31.0000]^T$ while with outlier $w = [-3.8577\ -6.1313\ 25.0000]^T$. Graphically, the separation provided by these weights as can be seen in fig. 1.

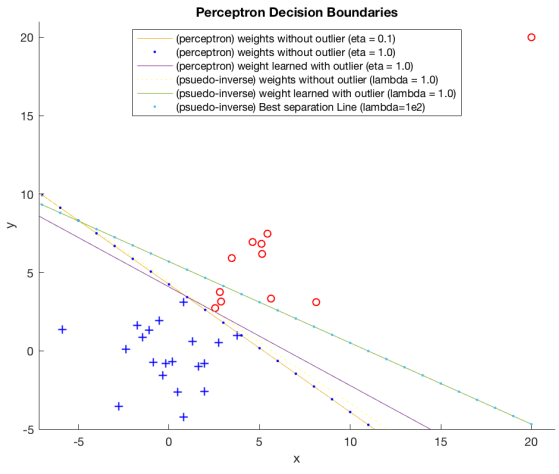

Fig. 1. Graph of data set with perceptron learned separation line.

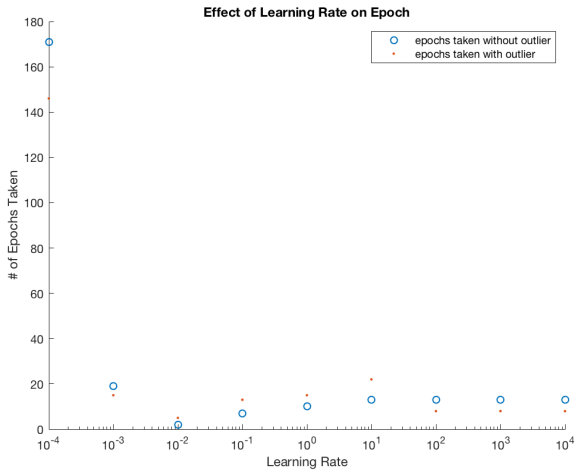

Fig. 2. Effect of different learning rates η on number of epochs.

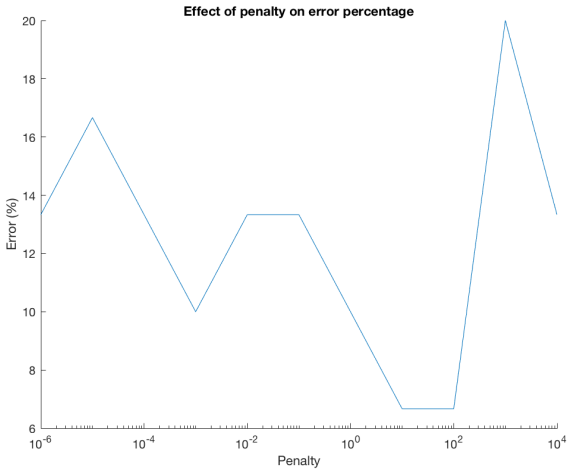

Fig. 3. Effect of different penalty parameters λ on percent error.

Now, we review the effect of outliers on learning after applying a penalty parameter $\lambda = 1$ using pseudoinverse. Pseudoinverse learning takes a single epoch to achieve its final result as opposed to the iterative learning algorithm presented previously within this report. As we will see, this prevents learning from infinitely searching for a linearly separable solution on non-linearly separable datasets simply because pseudoinverse takes a one-shot approach to learning. A result of this however is pseudoinverse may not ensure as accurate results when faced with outlier data.

Figure 1 presents the outcome of learning on both non-outlier datasets and dataset with outlier. Without outlier, the dashed line presents, a very good separation with 100% accuracy results. Pseudoinverse is not effective with the given outlier data, misclassifying 3 negative class elements.

IV. CROSS VALIDATION

One method to handle this inaccuracy with pseudoinverse is to apply a penalty λ. To determine the best value for λ, we must apply cross validation. Cross validation checks the average accuracy for different possible values of a variable on different parts of the dataset. For example, in this example, we ran 10-fold cross validation for λ as each of $[1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^{0}, 1e^{1}, 1e^{2}, 1e^{3}, 1e^{4}]$. This means, for each value of λ, the dataset is split into 10 different groups of training and test data to validate the generality of the learning for the different parameters of λ. Figure 3 shows $1e^1$ and $1e^2$ provide the lowest percent error. From this, we can choose $1e^2$ from this list because it is the largest value for λ. By choosing the largest value, we ensure that the mathematics involved for this process will not result in very small numbers a computer may round to 0. This is simply a small technicality which we must be aware of when working with computers on these sorts of problems.

Using cross validation, we build a final psuedoinverse model with $\lambda_{best} = 1e^2$ (fig. 1) which results in the weights $w = [-0.1199 \ \ -0.1623 \ \ 0.3887]^T$ when trained without outlier and $w = [-0.0435 \ \ -0.0997 \ \ 0.2475]^T$ with outlier.

V. CONCLUSION

In this report, we review perceptron learning through minimizing errors as well as through pseudoinverse. Through these methods, we looked at the effects of both learning rate η as well as regularization of data through different penalty parameters λ. Finally, cross validation was used to help determine the best parameters for a given learning method keeping the generality of the model in mind.

REFERENCES

[1] V. Kecman (2001). Learning and soft computing. Mit Press, p.2014.