Project 1 : Single Layer Linear Neural Network
Report

Sajal Kumar

Datasets

All statistics being reported were computed on the following 4 data sets.

Dataset	**# attributes**	**# samples**	**# class-labels**
Iris	4	150	3
Glass	9	214	7
Indian-Liver-Patient	10	583	2
Wine-Quality	11	1599	6

Performance

a) Accuracy

All accuracy being reported were computed on separate test cases that uniformly represented (stratified partitioning) 30% of the data-set. The software being submitted automatically activates 'one versus all' strategy for any data-set that has more than two class-labels. The models were trained on standardized data, the transformations were applied on each of the 'm' columns of X :

$$\forall i \in [0, m), X[, i] = \frac{X[, i] - \mu_i}{\sigma_i} \tag{1}$$

where μ_i and σ_i represents the mean and standard deviation of $X[, i]$.

Accuracy of Perceptron :

The Perceptron algorithm reported an accuracy of 93% on 'Iris', 52% on 'Glass', 21% on 'Wine-Quality' and 63% on 'Indian-Liver-Patient' data-sets.

Accuracy of Adeline :

The Adeline algorithm reported an accuracy of 84% on 'Iris', 62% on 'Glass', 57% on 'Wine-Quality' and 70% on 'Indian-Liver-Patient' data-sets.

Accuracy of Stochastic Gradient Descent :

The Stochastic Gradient Descent algorithm reported an accuracy of 84% on 'Iris', 62% on 'Glass', 57% on 'Wine-Quality' and 70% on 'Indian-Liver-Patient' data-sets.

b) Cost reduction during training

Here we show the cost reduction plots for 'Iris' and 'Indian-Liver-Patient' data-sets. Other plots have been uploaded to github for reference.

Mis-classification rate for Perceptron:

Perceptron saw a steady decrease in cost for all class labels of Iris. However, for Indian-Liver-Patient it can be seen that a lower mis-classification rate was achievable.

SSE rate for Adeline:

Adeline saw a steady decrease in cost for all class labels of Iris and Indian-Liver-Patient.

SSE rate for Stochastic Gradient Descent:

SGD saw a steady decrease in cost for all class labels of Iris and Indian-Liver-Patient.

c) Performance evaluation on other data-sets

The answers for (a) and (b) already include performance evaluation for atleast one more data-set other than Iris.

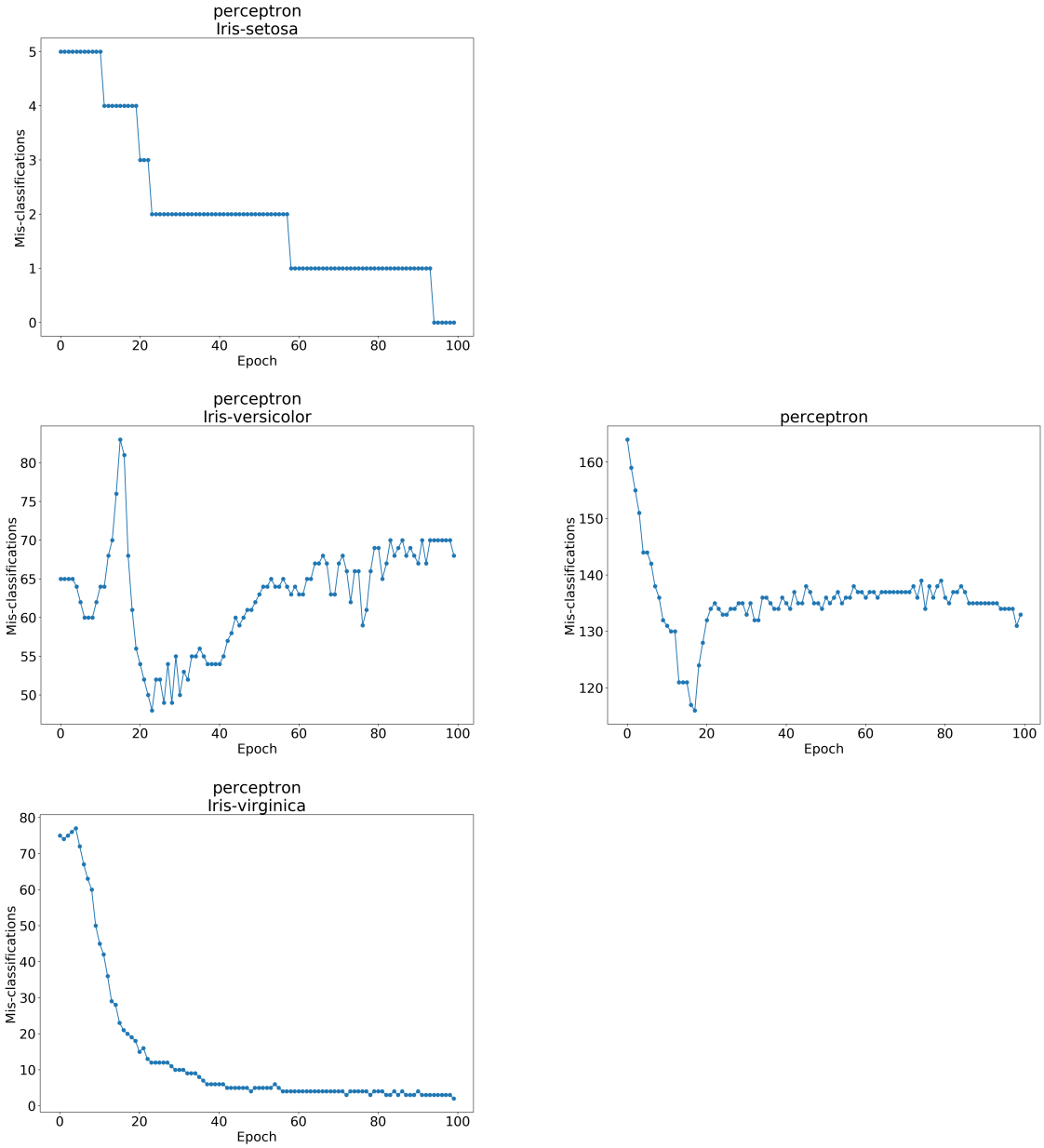

Figure 1: Cost reduction per iteration for perceptron – Iris data-set on the left and Indian-Liver-Patient dataset on the right

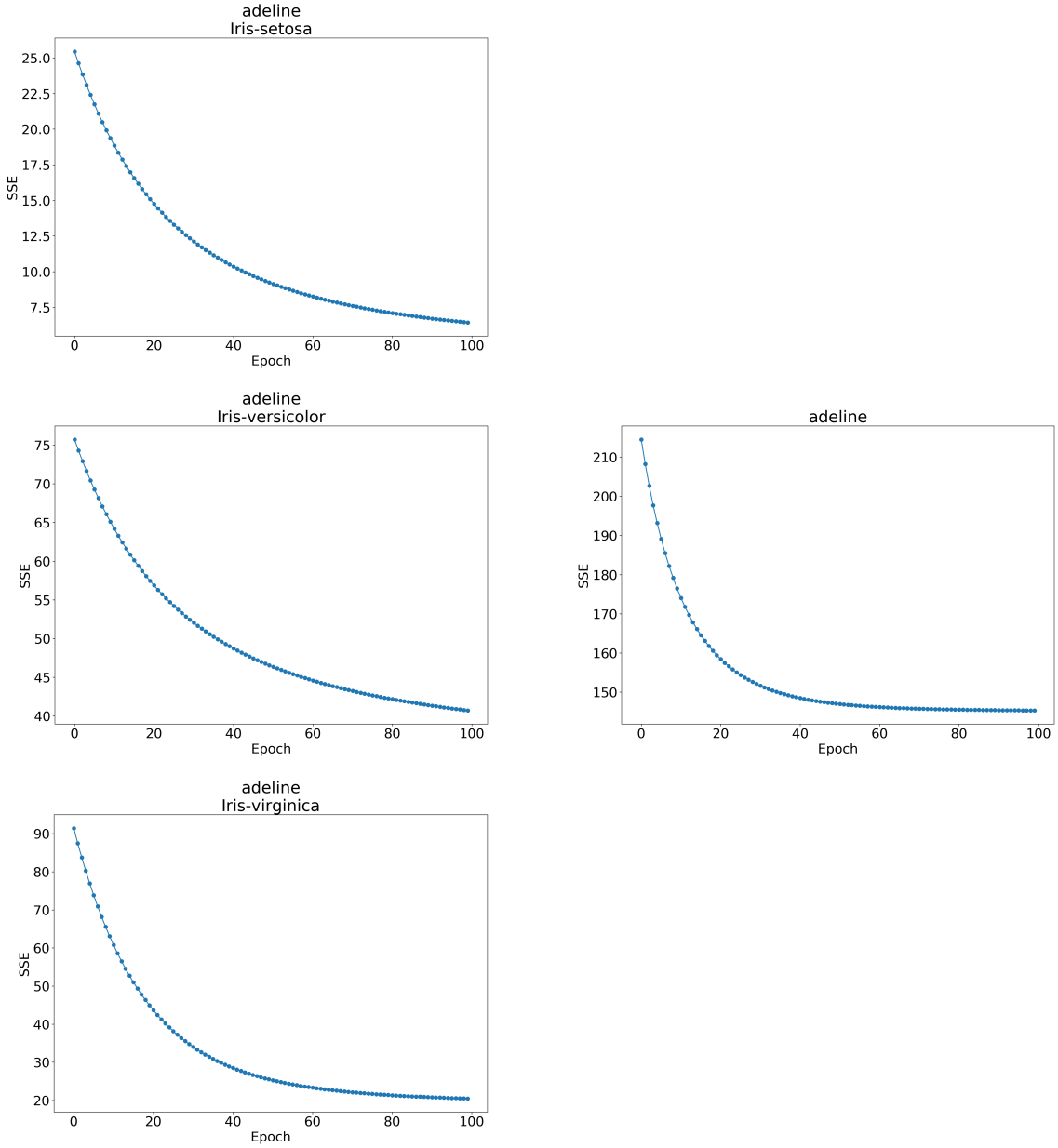

Figure 2: Cost reduction per iteration for Adeline – Iris data-set on the left and Indian-Liver-Patient dataset on the right

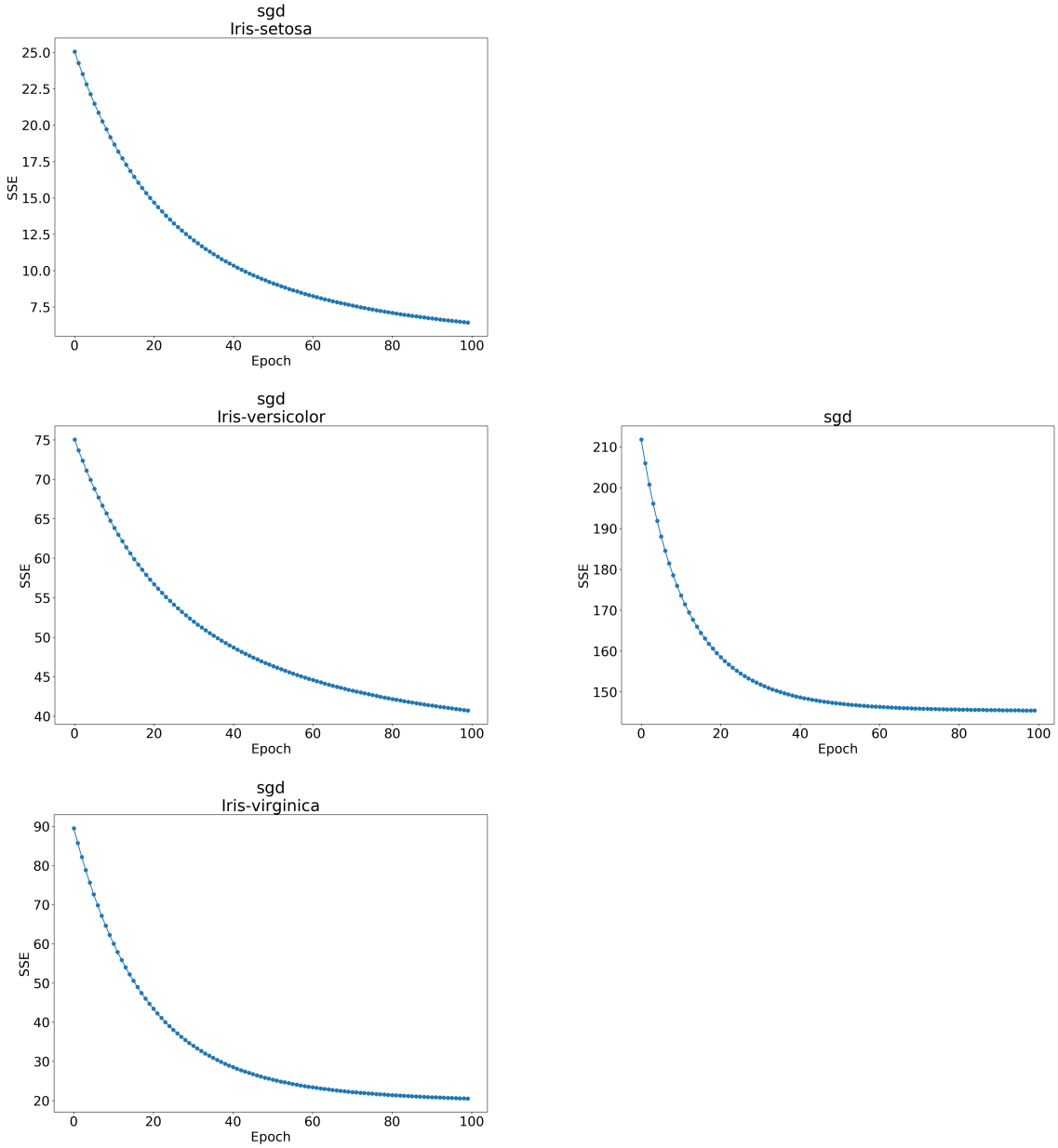

Figure 3: Cost reduction per iteration for sgd – Iris data-set on the left and Indian-Liver-Patient dataset on the right

d) Behavior of algorithms

Perceptron :

The effect of learning rate on Perceptron was found to be relatively weaker than that compared to Adeline or SGD. We tested the accuracy of Perceptron for both standardized and raw data-set and it seemed that the raw data always produced worse result (accuracy and cost). As can be seen in the cost-reduction (mis-classification in this case) for Perceptron, it tends to be more unstable when compared to Adeline and SGD, that is, one can achieve a decently low cost but could be thrown back up, as can be seen on the Indian-Liver-Patient data-set.

Adeline :

The effect of learning rate on Adeline was very drastic, a high learning rate can completely blow up the cost function, η (learning rate) of 0.0001 seemed like a good number after trying various others. Scaling the data was necessary for Adeline as the raw data usually produced worse results or in some cases (e.g., in Glass and Wine-Quality) produced significantly high cost and low accuracy. The cost reduction is smooth for Adeline and increasing the number of epochs can result in the improvement of accuracy, however, the improvement becomes subtle as the number of iterations are increased.

Stochastic Gradient Descent :

This algorithm behaved exactly same as Adeline.

e) Other peculiarities :

The most interesting behavior came with 'one versus all' prediction algorithm. The ideal way of making a decision is to call the 'predict' function and then choose '*argmax*' of the n prediction (n being the number of class labels). However, it can be an issue because of how 'step function' works. Two negative numbers $[-0.001, -0.99]$ will both be converted to -1 event though -0.001 is much closer to 0 than to -1 and thus changing just for 'one versus all' predictions, not applying the step function before choosing the '*argmax*' improved the accuracy by atleast 10% for each method on the Iris data-set.

7) One versus all

As stated earlier, the submitted software automatically activates one-versus-all if the number of class labels is more than two. The results reported above include results from one-versus-all.