MACHINE LEARNING-ASSIGNMENT 1 REPORT

CS519

JOANNA AUGUSTINE

800656114

Objective: The main objective of this project/assignment is that we Implement Perceptron, Adaline, Stochastic Gradient Descent and One Vs All strategy for multiclass Classifier

Description: For this we make use of the IRIS dataset and implement the three models. We further investigate the models by training and testing them with 70% and 30% of the data respectively. The data used here are training.txt and testing.txt.

With the help of all these information, we predict the errors and the accuracy of the data.

Procedure:

i)Implementation of Perceptron:

perceptron.py

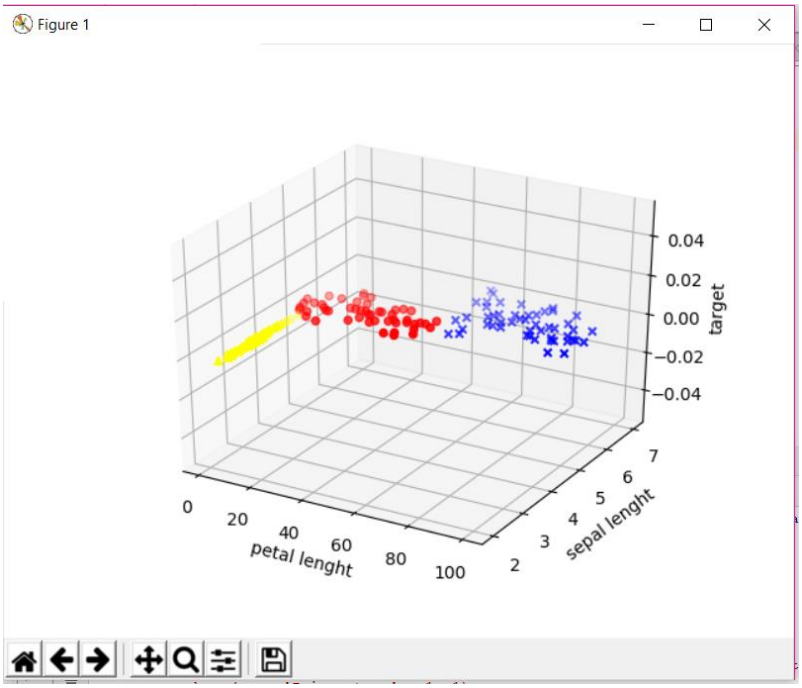


Fig 1.1 Shows the 3 Dimensional representation of the plot of a perceptron.

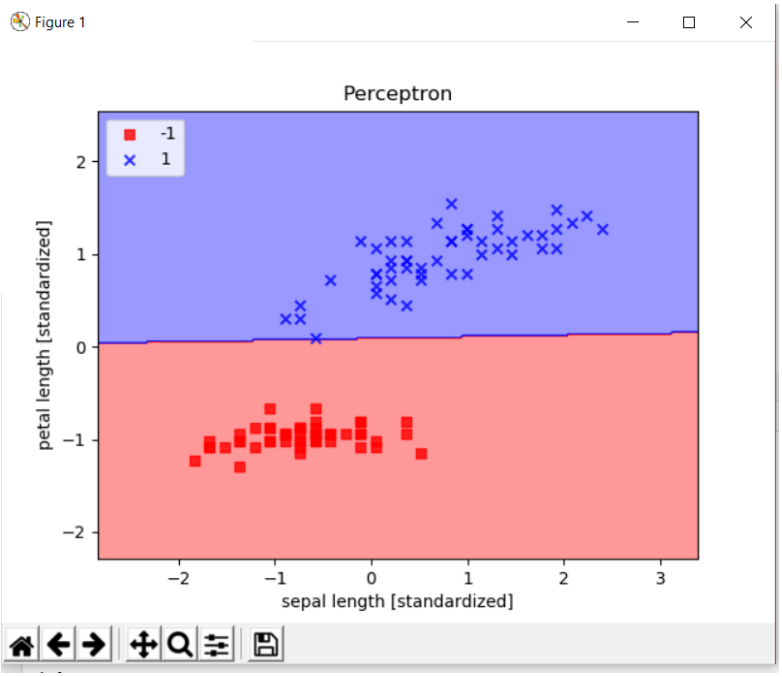


Fig 1.2 Shows that the data is linearly classified into two parts as perceptron is a binary classifier. The sepal length and the petal length are shown in the x and y axis respectively.

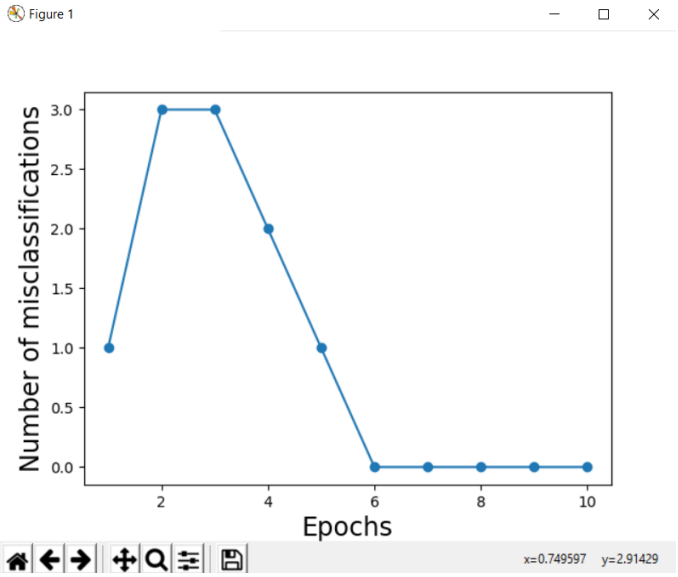


Fig 1.3 Shows that the number of misclassifications or the errors are at the maximum during the 2nd and 4th Epochs(iteration or time) and then they gradually descend to have 0.0 misclassifications at a constant rate.

As we can see, the perceptron converges after the 6th iteration and separates the two flower classes perfectly.

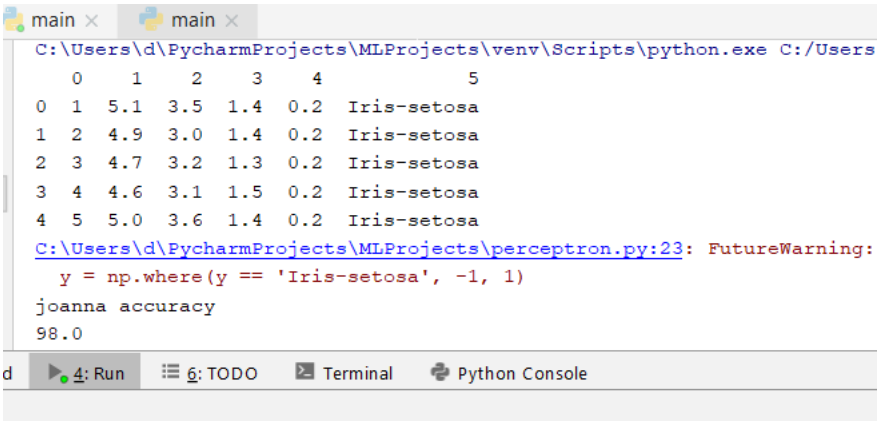
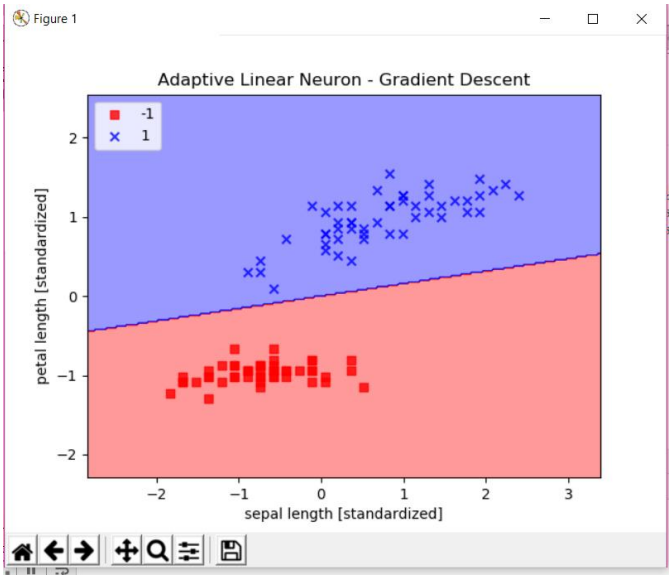


Fig 1.4 Shows the accuracy which is displayed for the various iterations of the Perceptron

ii)Implementation of Adaline:

adaline.py



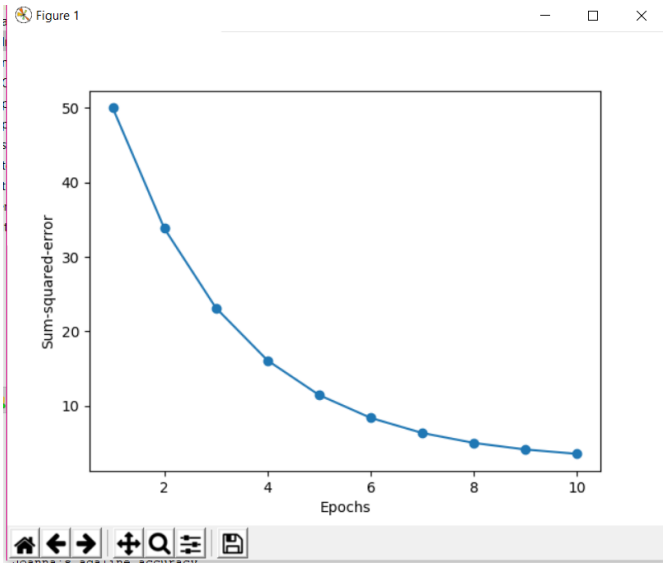


Fig 2.2 shows that If the learning rate is too large, gradient descent will overshoot the minima and diverge. The learning rate here is 0.01.

If the learning rate is too small, the algorithm will require too many epochs to converge and can become trapped in local minima more easily.

```
Joanna's adaline accuracy
95.85257298539285 %
3.565297671844979
[1.09873374 1.09745786 1.17149665 1.02022937 1.0980958  0.87470355
 1.09554404 1.02278113 1.09426816 1.02214319 1.02533289 0.94619058
 1.09681992 1.31957424 1.25382866 1.02724671 1.17596223 1.09873374
 0.87661737 1.02341907 0.87470355 1.02341907 1.39680273 0.87278973
 0.72024656 0.94746646 0.94746646 1.02405701 1.09937168 0.94555264
 0.94619058 1.02533289 1.02405701 1.1012855  1.02214319 1.24872514
 1.17660017 1.02214319 1.16958283 1.02341907 1.17341047 1.17022077
 1.16958283 0.94746646 0.72216038 1.09681992 0.9481044  1.09554404
 1.02469495 1.0980958  0.62547044 0.77227214 0.47420315 1.14310404
 0.69759541 0.76780656 0.62100486 1.6664791  0.69823335 1.21650489
 1.5164877  0.99502646 1.14629374 0.61972898 1.44500067 0.84950063
 0.76716862 1.06970319 0.77099626 1.21905665 0.54313843 1.14693168
 0.47037551 0.61972898 0.92290148 0.84886269 0.54887989 0.3976126
 0.76972038 1.52095328 1.29373338 1.36904805 1.22033253 0.31783235
 0.76589274 0.76972038 0.62355662 0.84694887 1.06842731 1.14310404
 0.84184535 0.69504365 1.14501786 1.66711704 0.99311264 0.99375058
 0.99375058 0.9216256  1.893699   1.06906525]
Joanna's adaline accuracy
96.43470232815503 %
```

Fig 2.3 shows the accuracy displayed for the testing data

ii)Implementation of Stochastic Gradient Descent:

sgd.py

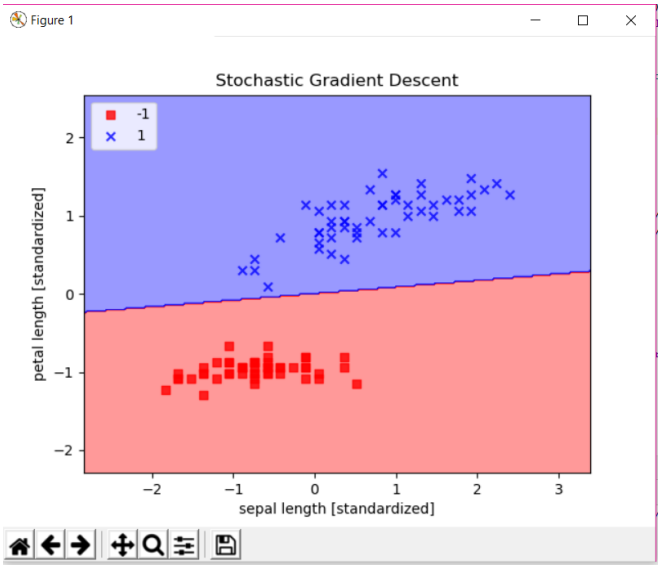


Fig 2.4 The values are plotted considering both the petal length and the sepal length

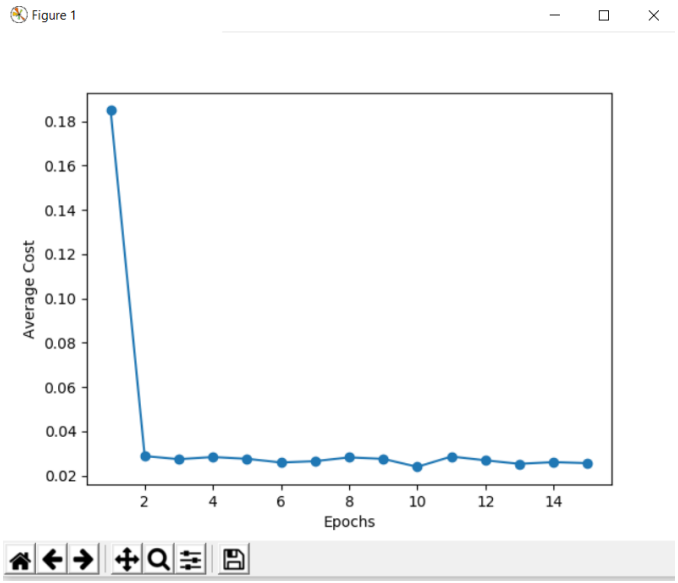


Fig 2.5 shows that If gradient descent is working properly, the cost function should decrease after every iteration.When Gradient Descent can't decrease the cost-function anymore and remains more or less on the same level, we say it has converged

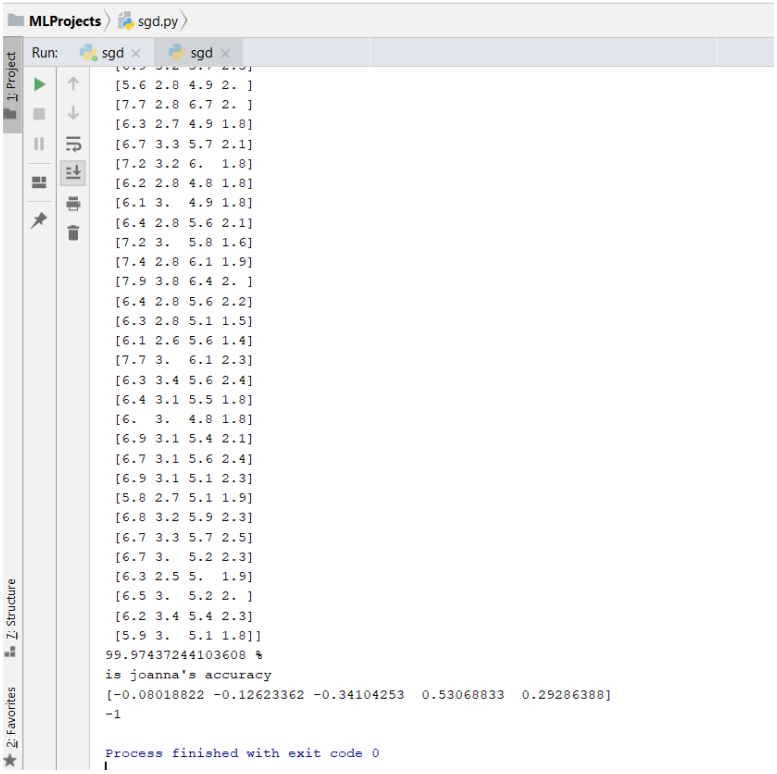


Fig 2.6 The figure above shows the accuracy of stochastic gradient descent which is 99.97%

One Vs Rest Classifier

iii)Implementation of Multiclass using SGD

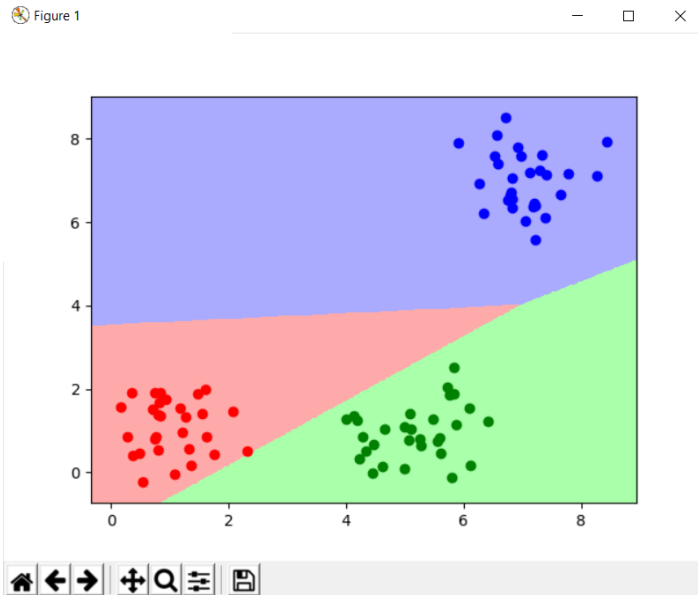


Fig 3.1 shows that the three datasets have been separated and plotted accordingly using One Vs Rest strategy

Findings:

(a)Perceptron runs with an accuracy of 98% for the iris dataset.

Adaline reports an accuracy of 96.43% and runs through for 10 iterations

SGD reports an accuracy of 99.97% when tested with the Iris dataset

(b)The errors are being reported and plotted in the above figures for each classifier.

(c),(d),(e) Gradient descent is also a good example why feature scaling is important for many machine learning algorithms. It is not only easier to find an appropriate learning rate if the features are on the same scale, but it also often leads to faster convergence and can prevent the weights from becoming too small (numerical stability).

We can track convergence in Perceptron using the errors attribute

The same is with Perceptron and Adaline linear classifiers too.

.

Result:

Various datasets were tested and trained and their accuracy was more or less the same as that of the accuracy of the Iris dataset.

1)Binary Perception Classifier, labels={setosa-versicolor}, eta=0.01, n iter=10, accuracy=100%

2)Adaline Classifier, labels={setosa-versicolor}, eta=0.01, n iter=100, accuracy=96.41%

3)Stochastic Gradient Descent Classifier, labels={setosa-versicolor}, eta=0.01, n iter=10, accuracy=98.2%

REFERENCES:

[1] https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/multi_layer_perceptron_mnist.html

[2] http://scikit-learn.org/stable/auto_examples/plot_multilabel.html#sphx-glr-auto-examples-plot-multilabel-py