Neural Networks MLP

1. Problem Statement

The data set should be classified using Neural Networks feed-forward MLP classification model. The data with K class should be trained and tested using the MLP model. Neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other. Here the neurons are nothing but logistic function unit. Many logistic functions are combined to form the neural network. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning. The 10-fold cross validation has to be done to analyze the performance of the model. The accuracy of the model is analyzed with the training and testing error calculated by Mean Square Error.

2. Problem Solution

Build a model with two layers, one hidden layer and one output layer. The number of parameters in the hidden layer is set dynamically. The solution is to find the parameters of V and W. The V is the parameter for the output layer and W is the parameter for the hidden layer. Based on the output layer's output we find the parameter of the given data and determine the class label of the data using discriminant function. The features and the labels are in the form of vectors. The predictor function is used for modeling.

3. Implementation Details

The initial parameters are assumed randomly and close to zero. This plays a more role in classification. If this is not set properly the data point doesn't converge well. Use the data X and the current guess of W to find Z using a logistic function. And using the Z and V find the prediction of Y. By Minimizing the error function update V and W using the stochastic gradient descent function. Using the find V and W find Z on the test data and classify the values of the test data. I have implemented the code in ipython notebook. The filename has to be mentioned and in the tool bar option “Cell” -> “Run All” will implemented the whole file and the results will be printed.

4. Results and Discussion

The data set used is ""**iris.data.txt**""

The data is split of training and testing using sklearn's train_test_split. The evaluation is done based on different values of learning rate, iteration count and the number of layers.

a) H=4, learning rate =0.00454,iteration count = 1000

```
Confusion Matrix
[[23  0  0]
 [ 0 18  1]
 [ 0  0 18]]
Accuracy 0.983333333333
Precision [1.0, 1.0, 0.94736842105263153]
Recall [1.0, 0.94736842105263153, 1.0]
F_score [1.0, 0.97297297297297303, 0.97297297297297303]
```

10 fold cross validation

```
Fold 0 Testing Error [ 0.4]
Fold 1 Testing Error [ 0.4]
Fold 2 Testing Error [ 0.2]
Fold 3 Testing Error [ 0.46666667]
Fold 4 Testing Error [ 0.]
Fold 5 Testing Error [ 0.33333333]
Fold 6 Testing Error [ 0.53333333]
Fold 7 Testing Error [ 0.4]
Fold 8 Testing Error [ 0.33333333]
Fold 9 Testing Error [ 0.4]
Average Mean Square Error
Training Error  Testing Error
0.317037037037 0.346666666667
```

b) H=8,iteration count = 500

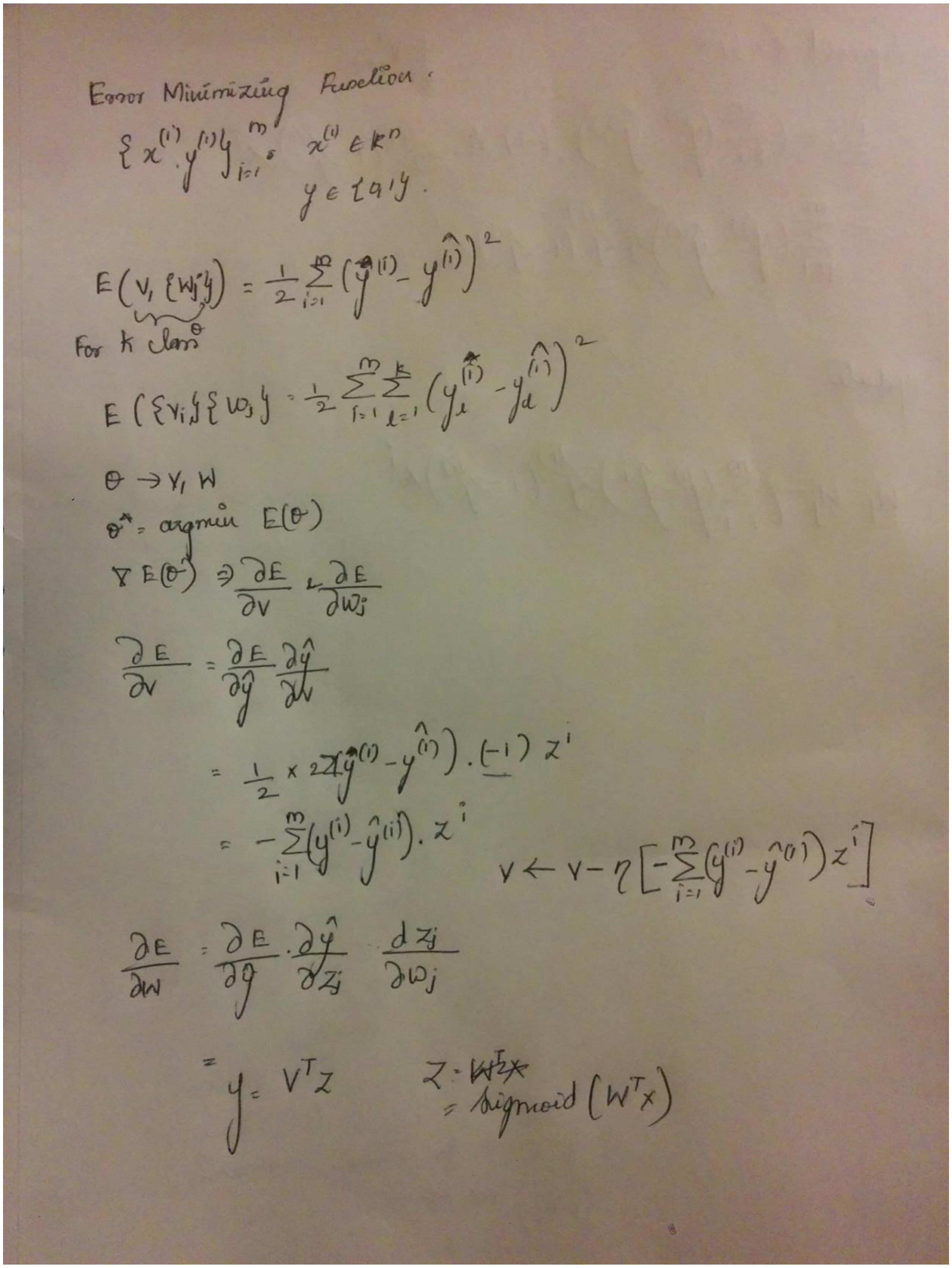
```
Confusion Matrix
[[23  0  0]
 [ 0 12  7]
 [ 0  0 18]]
Accuracy 0.883333333333
Precision [1.0, 1.0, 0.71999999999999997]
Recall [1.0, 0.63157894736842102, 1.0]
```

```
F_score [1.0, 0.77419354838709675, 0.83720930232558133]
```

H = 5, Iteration count = 1000

```
Confusion Matrix
[[23  0  0]
 [ 0  5 14]
 [ 0  0 18]]
Accuracy 0.766666666667
Precision [1.0, 1.0, 0.5625]
Recall [1.0, 0.26315789473684209, 1.0]
F_score [1.0, 0.41666666666666669, 0.71999999999999997]
```

5. Error Minimizing Function



Error Minimizing Function :

$\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$: $x^{(i)} \in R^{n}$

$y \in \{0,1\}$.

$$E(\underbrace{v, \{w_j\}}_{\theta}) = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

For k class

$$E(\{v_i\}\{w_j\}) = \frac{1}{2}\sum_{i=1}^{m}\sum_{l=1}^{k}\left(y_l^{(i)} - \hat{y}_l^{(i)}\right)^2$$

$\theta \rightarrow V, W$

$\theta^* = \arg\min E(\theta)$

$\nabla E(\theta^*) \Rightarrow \frac{\partial E}{\partial v}$ & $\frac{\partial E}{\partial w_j}$

$$\frac{\partial E}{\partial v} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial v}$$

$$= \frac{1}{2} \times 2\sum\left(y^{(i)} - \hat{y}^{(i)}\right) . \underline{(-1)}\, z^{i}$$

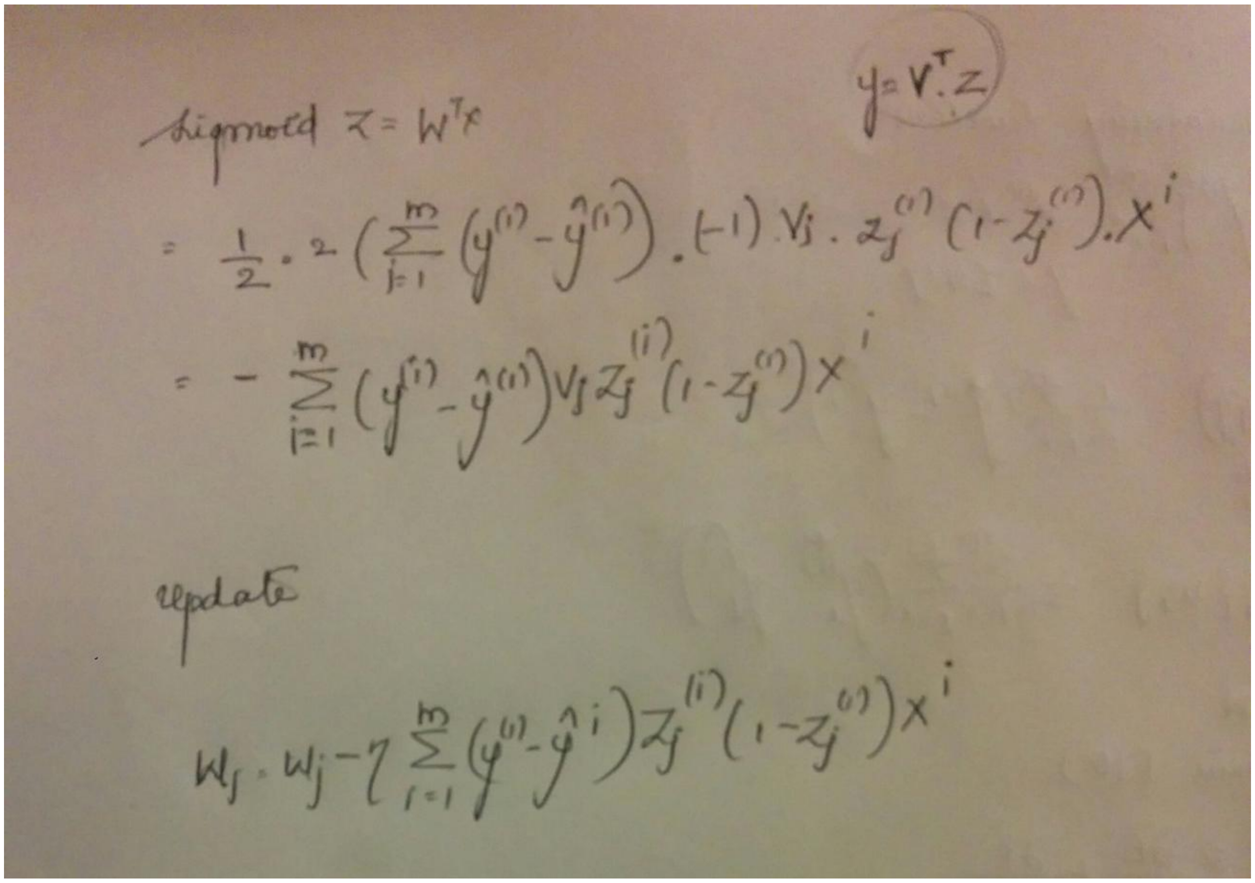
$$= -\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right) . z^{i}$$

$$v \leftarrow v - \eta\left[-\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right) z^{i}\right]$$

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial z_j} \frac{d z_j}{\partial w_j}$$

$= \hat{y} = V^T z$ $z = \cancel{W^T x}$

$= \text{sigmoid}(W^T x)$



sigmoid $z = W^T x$

$y = V^T . z$

$$= \frac{1}{2} \cdot 2 \Big(\sum_{i=1}^{m} \big(y^{(i)} - \hat{y}^{(i)}\big) . (-1) . V_j . z_j^{(i)} \big(1 - z_j^{(i)}\big) . x^i$$

$$= - \sum_{i=1}^{m} \big(y^{(i)} - \hat{y}^{(i)}\big) V_j z_j^{(i)} \big(1 - z_j^{(i)}\big) x^i$$

update

$$W_j = W_j - \eta \sum_{i=1}^{m} \big(y^{(i)} - \hat{y}^{i}\big) z_j^{(i)} \big(1 - z_j^{(i)}\big) x^i$$

6. References

https://en.wikipedia.org/wiki/Logistic_regression