MACHINE LEARNING

Assignment-3

Discriminative Learning

SPRING – 2016

submitted to

Gady Agam

by

Darpan Patel
A20345898

PROBLEM STATEMENT

Here, Here, In this assignment I have implemented discriminative learning approach on various **UCI** datasets using algorithm LOGISTIC REGRESSION for **2-class** labels as well as **k-class** labels.

Here, also we need to implement MLP (Multi-Layer Perceptron) for two layer feed forward for three class-classification.

1.) Logistic Regression

- Select appropriate datasets from UCI datasets. I choose MNIST datasets for classification.
- Extract Feature vectors and class labels from MNIST.
- Implement 2-class classification using only 2-class extracted data. [0,1]
- Implement k-class classification using 3-class extracted data [0,1,2]
- Used Non-linear input combinations as 784 features of MNIST dataset.
- Measure the performance in terms of MSE, accuracy, precision, recall, F-score.

2.) Multi – Layer Perceptron

- Implement two-layer feed forward MLP for three class classification. I choose IRIS datasets for 3-class classification
- Evaluate the performance of the algorithm as a function of the number of elements in the hidden layer and as a function of the learning rate parameter.
- Evaluate the effect of parameters.
- Used momentum in algorithm.
- Compare result with scikit-learn implementation.

PROPOSED SOLUTION

- I implemented two separate programs for Logistic regression and MLP.
- For Logistic Regression I made one program which two different logic for 2-class classification and k-class classification. (Logistic_regrssion.py).
- For MLP I made one separate program which is working for 2-class as well k-class classification.(MLP_k_class.py)
- I have used MNIST image dataset for logistic regression for both 2-class and k-class classification.
- I have implemented all the function from scratch using some basic python libraries like numpy, scipy - for array and matrix functionalities and sklearn – for just check for correctness of outputs.
- I have used IRIS datasets for Multi-Layer Perceptron.

Algorithm of Logistic Regression

- The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). As such it is not a classification method. It could be called a qualitative response/discrete choice model in the terminology of economics.
- Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

- The Logistic Function is like as step function but it mainly depends uppon the learning rate $(\boldsymbol{\eta})$ and initial value of theta $(\boldsymbol{\theta})$.

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

- Here, I implemented Logistic regression algorithm for 2-class and k-class classification on MNIST datasets. The datasets is of digit recolonization image data.
- In this proposed solution, I extracted Feature vectors (X) and class label (y). For 2-class LR problem I extracted only those feature vectors which has class labels y=0 and y=1. And for k-class I extracted features whose labels is y=[0,1,2]. So here 3-

class classification for multi-class.

- Then I assumed the initial values of theta (**θ**) and learning rate (**η**) as below.

η = 0.0001 and θ = 0.00 (all features).

- I used some of in-bulit math libraries function to made calculation easy and simple.
- Then find the value of theta for multiple iterations over and over on same features for better learning of model and for accurate decision boundary.

$$\theta_{j+1} = \theta_j - (\text{learning rate}) * \sum_{i=1 \text{ to } i=m} (h_\theta(x^{(i)}) - y^{(i)}) * X^{(I)}$$

θ_{j+1} = new theta θ_j = previous theta $h_\theta(x^{(i)})$ = logistic function
X = feature vector y = class label Find the training error and testing error from predicted values

- The equation of Logistic Function ($h_\theta(x^{(i)})$) mention below
  - For 2-class SIGMOID function has been used

$$h_\theta(X^i) = 1/(1+\exp(\theta^T * X))$$

  - For k-class SOFTMAX function has been used

$$h_{\theta j}(X^i) = \exp(\theta^T_j * X) / \sum_{i=1 \text{ to } i=m} \exp(\theta^T_i * X)$$

- For 2-class LR find the SIGMOID function value and then made a decision using decision boundary threshold. Here, decision boundary threshold is **0.5** means value of $h_\theta(X^i) > 0.5$ then predicted class is 1 else predicted class is 0. Find the training error and testing error from predicted values. Use K-fold cross validation for all training datasets. default value of K_fold is 10.

- Predicted the class label ass per maximum LOGISTIC function value. Find the training error and testing error from predicted values. Use K-fold cross validation

for all training datasets. default value of K_fold is 10(User can change it.).

- At the end find the mean-square-error, Confusion Matrix and find Accuracy, Precision, Recall and F-Measure for all the fold and then take maximum accuracy fold and retrieve it for better results and efficiency.

- Implement the Logistic Regression using in-built function of library of **sklearn.linear_model** named **Logistic Regression()**. Find the training error and testing error from predicted values. Use K-fold cross validation for all training datasets. default value of K_fold is 10(User can change it.). Evaluates all the parameters mention above and compare it with my Algorithm result.

Algorithm of Multi-Layer Perceptron (MLP)

- An MLP can be viewed as a logistic regression classifier where the input is first transformed using a learnt non-linear transformation Φ. This transformation projects the input data into a space where it becomes linearly separable. This intermediate layer is referred to as a **hidden layer**
- The MLP network consists of input,output and hidden layers.Each hidden layer consists of numerous perceptron's which are called hidden units
- In this problem, the complexity of all the equations grows very high and there was problem being faced in the hidden layers. Despite knowing how many dimensions to grow for hidden layer i.e((k+n)/2), I couldn't figure out how to populate those feature in the hidden layers.
- Also, the problem was implementing such a complex pattern in a limited period of time, by taking care of the dimension mismatch errors coming up every other minute seemed very difficult.
- It helped clearing a lot of concepts and the derivations bellows explains the behavior for the same.
- I used Back Propagation equation into my algorithm. I calculated sigmoid of X-test and weight function then find derivative of it. Then predict the value of features of IRIS data.
- Find the training error and testing error from predicted values. Use K-fold cross validation for all training datasets. default value of K_fold is 10(User can change it.).

- At the end find the mean-square-error, Confusion Matrix and find Accuracy, Precision, Recall and F-Measure for all the fold and then take maximum accuracy fold and retrieve it for better results and efficiency.

- Implement the Multi-Layer Perceptron using in-built function of library of **sklearn.neural_network** named **MLPClassifier()**. Find the training error and testing error from predicted values. Use K-fold cross validation for all training datasets. default value of K_fold is 10(User can change it.). Evaluates all the parameters mention above and compare it with my Algorithm result.

Implementations Details

Design Issue :

- np.exp function is not working correctly. As it gives “inf” value for bigger value than 1000.
- Data Fetching Time of MNIST is little bit high for all 70000 examples and 784 features.
- Difficult to find exponent of higher value and calculation time for large data on higher ITERATIONS for gradient decent. More than 50 iteration will take ample amount of time to execute and evaluate the parameters.
- Design issue for Two-Layer MLP and While attempting MLP from sklearn, ‘cannot import name MLPClassifier’, pops up.

How Problem Solved :

- faced a problem this kind of function from scratch in editor so that I used PyCharm IDE for python coding.
- Use in-built libraries for finding an exponent of an array.
- Set learning rate as small as possible and iterations is 50.
- Tried installing scikitlearn and many other solutions, Then I need to install scikit-learn version 0.18 dev for MLPClassifier library.

How to run program :

- Run Logistic_Regrssion.py (1st Question)
- Run MLP_k_class.py (2nd Question)

RESULT and DISCUSSION

2-class Logistic Regression

K-Fold =10 and Iterations =50

No. of Class	Accuracy	Precision	Recall	F-Measure	MSE
0 - class	0.9946	0.9887	1.0	0.9943	0.0054
1 - class		1.0	0.9897	0.9948	

```
--------------------------------------------------------------------------------
                Parameter Evaluation
--------------------------------------------------------------------------------

MSE ::  0.0054      OR      0.54  %

Accuracy ::  0.9946     OR      99.46  %

Precision ::  {0: 0.9887, 1: 1.0}

Recall ::  {0: 1.0, 1: 0.9897}

F-measure ::  {0: 0.9943, 1: 0.9948}


--------------------------------------------------------------------------------
                In Built - Parameter Evaluation
--------------------------------------------------------------------------------

MSE ::  0.0         OR      0.0  %

Accuracy ::  1.0    OR      100.0  %

Precision ::  1.0

Recall ::  1.0

F-measure ::  1.0

Process finished with exit code 0
```

- Above image have sklearn.liner_model in-built Logistic Regression classifier's parameter evaluation compared to my algorithm.

k-class Logistic Regression

K-Fold =10 and Iterations =10

No. of Class	Accuracy	Precision	Recall	F-Measure	MSE
0 - class		0.9537	0.9743	0.9893	
1 - class	0.9246	0.9322	0.9897	0.9873	0.0754
2 – class		0.9432	0.9832	0.9854	

- Here, the accuracy is sometimes turns up to very low, because the classifier is unable to predict the correct value. Exponent function's behavior is not stable so it gives theta value in-correct and so that logistic function turns up wrong.

3-class Multi-Layer Perceptron (MLP)

No. of Class	Accuracy	Precision	Recall	F-Measure	MSE
0 - class		0.0	0.0	0.0	
1 - class	0.7143	0.7143	1.0	0.8333	0.2857
2 – class		0.0	0.0	0.0	

```
--------------------------------------------------------------------------------
                  k-class Multi-Layer Perception
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
                Parameter Evaluation
--------------------------------------------------------------------------------

MSE ::  0.2857      OR      28.57  %

Accuracy ::  0.7143     OR      71.43  %

Precision ::  {0: 0.0, 1: 0.7143, 2: 0.0}

Recall ::  {0: 0.0, 1: 1.0, 2: 0.0}

F-measure ::  {0: 0.0, 1: 0.8333, 2: 0.0}
```

3-class Multi-Layer Perceptron (MLP) (scikit-learn)

```
Iteration 1, loss = 2.53151784
Iteration 2, loss = 3.99249831
Iteration 3, loss = 1.29474379
Iteration 4, loss = 0.96890436
Iteration 5, loss = 0.87465925
Iteration 6, loss = 0.80676383
Iteration 7, loss = 0.58819535
Iteration 8, loss = 0.48648006
Iteration 9, loss = 0.44631030
Iteration 10, loss = 0.41647224


Accuracy by scikit learn library:  80.0  %
/home/jaimin/anaconda2/lib/python2.7/site-packages/sklearn/neural_netw
  % (), ConvergenceWarning)
```

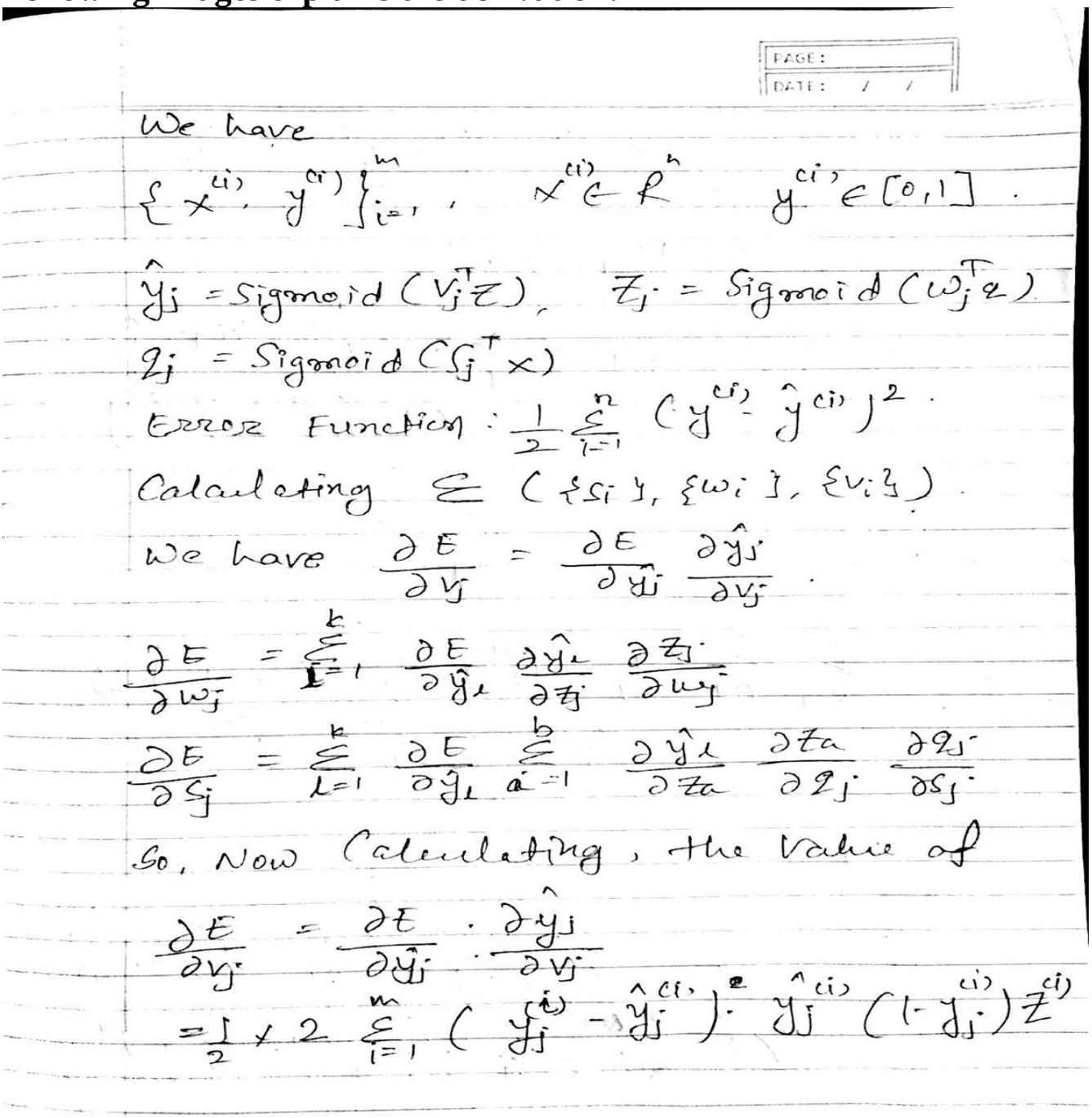
In-Built Accuracy = 80 %

Comments on Result

- Accuracy of 2-class Logistic Regression is near to 100 % because it is very accurate classification algorithm which uses gradient decent as learning part.
- LR has higher accuracy than all kind of Naive Bayes algorithm mentioned in previous assignment.
- LR k-class is also more powerful than all other classification algorithms.

Derivation(Problem-2(a)):

Following images explains the derivation:



PAGE :
DATE : / /

We have

$\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$, $x^{(i)} \in R^n$ $y^{(i)} \in [0,1]$.

$\hat{y}_j = \text{Sigmoid}(v_j^T z)$, $z_j = \text{Sigmoid}(w_j^T z)$

$z_j = \text{Sigmoid}(s_j^T x)$

Error Function : $\frac{1}{2}\sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$.

Calculating $\frac{\partial E}{\partial (\{s_i\}, \{w_i\}, \{v_i\})}$

We have $\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial v_j}$.

$\frac{\partial E}{\partial w_j} = \sum_{l=1}^{k} \frac{\partial E}{\partial \hat{y}_l} \frac{\partial \hat{y}_l}{\partial z_j} \frac{\partial z_j}{\partial w_j}$

$\frac{\partial E}{\partial s_j} = \sum_{l=1}^{k} \frac{\partial E}{\partial \hat{y}_l} \sum_{a=1}^{b} \frac{\partial \hat{y}_l}{\partial z_a} \frac{\partial z_a}{\partial z_j} \frac{\partial z_j}{\partial s_j}$

So, Now Calculating, the value of

$\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial v_j}$

$= \frac{1}{2} \times 2 \sum_{i=1}^{m} (y_j^{(i)} - \hat{y}_j^{(i)}) \cdot \hat{y}_j^{(i)} (1 - \hat{y}_j^{(i)}) z^{(i)}$

$$\therefore \frac{\partial E}{\partial v_j} = \sum_{i=1}^{m} (y_j^{(i)} - \hat{y}_j^{(i)})\, \hat{y}_j^{(i)} (1 - \hat{y}_j^{(i)})\, z^{(i)}$$

$$\therefore v_j \leftarrow v_{j-1} \eta \frac{\partial E}{\partial v_j}$$

$$\therefore v_j \leftarrow v_{j-1} \eta \left(\left[\sum_{i=1}^{m} (y_j^{(i)} - \hat{y}_j^{(i)}) \right] \hat{y}_j^{(i)} (1 - \hat{y}_j^{(i)})\, z^{(i)} \right)$$

$$\frac{\partial E}{\partial w_j} = \sum_{i=1}^{r} \frac{\partial E}{\partial \hat{y}_l} \frac{\partial \hat{y}_l}{\partial z_j} \frac{\partial z_j}{\partial w_j}$$

$$\therefore \frac{\partial E}{\partial w_j} = \sum_{i=1}^{n} \sum_{k=1}^{k} (y_k - \hat{y}_k)\, \hat{y}_k^{(k)} (1 - \hat{y}_k)\, v_{kj}\, z_j^{(i)} (1 - z_j^{(i)})\, g^{(i)}$$

$$w_j \leftarrow w_{j-1} - \eta \frac{\partial E}{\partial w_j}$$

$$w_j \leftarrow w_{j-1} - \eta \left[\sum_{i=1}^{n} \sum_{k=1}^{k} (y_k^{(i)} - \hat{y}_k^{(i)})\, \hat{y}_k^{(i)} (1 - \hat{y}_k^{(i)})\, v_{kj}\, z_j^{(i)} (1 - z_j^{(i)})\, g^{(i)} \right]$$

PAGE:
DATE: / /

A Now,

$$\frac{\partial E}{\partial s_j} = \sum_{k=1}^{k} \frac{\partial E}{\partial \hat{y}_k} \cdot \sum_{a=1}^{b} \frac{\partial \hat{y}_k}{\partial z_a} \cdot \frac{\partial z_a}{\partial q_j} \frac{\partial q_j}{\partial s_j}$$

$$\therefore \frac{\partial E}{\partial s_j} = \sum_{i=1}^{n} \sum_{k=1}^{k} (y_j^{(i)} - \hat{y}_k^{(i)}) \cdot \sum_{a=1}^{b} \hat{y}^{(i)} (1 - \hat{y}^{(i)})$$

$$\forall i,j \; z_a \; (1 - z_a) \cdot w^{(i)} \cdot q_j \; (1 - q_j) \cdot x^{(i)}$$

$$s_j \leftarrow s_j - 1 - \eta \frac{\partial E}{\partial s_j}$$

$$s_j \leftarrow s_j - 1 - \eta \left[\sum_{i=1}^{n} \sum_{k=1}^{k} (y_j^{(i)} - \hat{y}_k^{(i)}) \sum_{a=1}^{b} \hat{y}^{(i)} (1 - \hat{y}^{(i)}) \forall_{ij} z_a (1 - z_a) w^{(i)} q_j (1 - q_j) x^{(i)} \right]$$

PAGE:
DATE:

Classification Problem. Already Solved in Class:

All parameters are same as the one's used in regression, but the objective Function.

Objective function.

$$\mathcal{E} = \sum_{i=1}^{m} \sum_{j=1}^{k} P(y=j \mid x^{(i)}) \, 1(y^{(i)} = j)$$

which can be simplified.

$$\mathcal{E} = \sum_{i=1}^{m} \sum_{j=1}^{k} 1(y^{(i)} = j) \log(\hat{y}^{(i)})$$

∴ Calculating the parameters as in Regression.

$$\frac{\partial \mathcal{E}}{\partial v_j} = \frac{\partial \mathcal{E}}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial v_j} = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{1(y^{(i)} = j)}{\hat{y}^{(i)}}$$

$$\therefore v_j \leftarrow v_{j-1} - \eta \cdot \frac{\partial \mathcal{E}}{\partial v_j}$$

$$\therefore v_j \leftarrow v_{j-1} - \eta \left(\sum_{i=1}^{m} \sum_{j=1}^{k} \frac{1(y^{(i)} = j)}{\hat{y}^{(i)}} \right)$$

$$\frac{\partial E}{\partial w_j} = \sum_{a=1}^{b} \frac{\partial E}{\partial \hat{y}_a} \cdot \frac{\partial \hat{y}_a}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_j}$$

$$\therefore \frac{\partial E}{\partial w_j} = \sum_{i=1}^{n} \sum_{j=1}^{k} \sum_{a=1}^{b} \frac{1(y^{(i)} = j)}{y_{aj}^{(i)}} \cdot \hat{y}_a^{(i)} \left(1 - \hat{y}_a^{(i)}\right) V_{aj}^T z_a^{(i)} \left(1 - z_a^{(i)}\right) x^{(i)}$$

$$\therefore w_j \longleftarrow w_{j-1} - \eta \frac{\partial E}{\partial w_j}$$

$$\therefore w_j \longleftarrow w_{j-1} - \eta \left(\sum_{i=1}^{n} \sum_{j=1}^{k} \sum_{a=1}^{b} \frac{1(y_a^{(i)} = j)}{\hat{y}_a^{(i)}} \cdot \hat{y}_a^{(i)} \left(1 - \hat{y}_a^{(i)}\right) V_{aj}^T z_a^{(i)} \left(1 - z_a^{(i)}\right) x^{(i)} \right)$$

$$\therefore \frac{\partial E}{\partial s_j} = \sum_{a=1}^{b} \frac{\partial E}{\partial \hat{y}_a} \sum_{c=1}^{d} \frac{\partial \hat{y}_a}{\partial z_c} \frac{\partial z_c}{\partial q_j} \frac{\partial q_j}{\partial s_j}$$

$$= \sum_{i=1}^{m} \sum_{l=1}^{k} \sum_{a=1}^{b} \frac{1(y_a^{(i)} = j)}{\hat{y}_{aj}^{(i)}} \sum_{c=1}^{d} \hat{y}_a^{(i)} \left(1 - \hat{y}_a^{(i)}\right)$$

$$\cdot\ V_{cj}^{T}\ z_c^{(i)}(1-z_c^{(i)})\cdot w_j^{(i)}\ g_j(1-g_j)x$$

$$\therefore\ s_j \leftarrow s_{j-1} - \eta\frac{\partial E}{\partial s_j}$$

$$s_j \leftarrow s_{j-1} - \eta\left[\sum_{i=1}^{m}\sum_{l=1}^{k}\sum_{a=1}^{b}\frac{1(y_a^{(i)}=1)}{\hat{y}_a^{(i)}}\right.$$

$$\left.\sum_{c=1}^{q} y_a^{(i)}(1-\hat{y}_a^{(i)})\ V_{cj}^{T}\ z_c^{(i)}(1-z_c^{(i)})\right.$$

$$\left.w_j^{(i)}\cdot g_j(1-g_j)x^{(i)}\right)$$

REFERENCES

- http://deeplearning.net/tutorial/mlp.html
- http://www.codeproject.com/Articles/821348/Multilayer-Perceptron-in-Python
- http://pythonhosted.org/bob.learn.mlp/guide.html
- https://en.wikipedia.org/wiki/Logistic_regression#Definition_of_the_logistic_function
- http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html
- http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html