Project 2

CLASSIFICATION

CSE 574 – Introduction To Machine Learning

UNIVERSITY AT BUFFALO

Authors:

KEYUR SANJEEV JOSHI

keyurjos@buffalo.edu

ABSTRACT

OBJECTIVE

1. Implement the Classification algorithm to identify/classify hand written digits in dataset. Report the error rate obtained for the given dataset.
2. Display the error rate of results obtained for:
    - Logistic Regression
    - Neural Networks
    - Naïve Bayes algorithms.

INTRODUCTION

What is classification?

Classification is the problem of identifying to which of a set of categories a new observation belongs. It is a type of supervised learning.

It basically involves training on a set of data and apply the learning to a test set.

The types of classification methods:

Naïve Bayes

Logistic Regression

Neural Networks

Support Vector Machines

Decision Tree

What is Logistic Regression?

Standard Logistic Regression is a binary classifier used to predict class labels from features. It is a soft classifier that gives the probability of test input belonging or not belonging to a class. The class label having maximum probability is assigned. It is a special case of linear regression where output values are discrete instead of continuous. It represents a discriminative approach where output probabilities are directly computed.

The general form of equation is given by .

In case of multi class logistic regression the error function becomes a cross entropy function and the activation function soft max function.

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{\exp(a_k(\mathbf{x}, \mathbf{w}))}{\sum_j \exp(a_j(\mathbf{x}, \mathbf{w}))}$$ Where, $$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i.$$

The default coding scheme is a 1-K coding scheme meaning that each column from K is set to 1 if the input belongs to that particular class. This ensures consistency in having probabilistic output

Iterative gradient descent method is used to achieve the optimum weight matrix.

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

What are Neural Networks?

Neural networks (ANNs) are statistical learning algorithms used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. The basic neural network can be described as a series of functional transformation called activations. Here the node xi are neurons.

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

These are further changed by applying non-linear transformation h(.) called activation functions which are hidden units

$$z_j = h(a_j).$$

Network refers to the inter–connections between the neurons in the different layers of each system. Activation values are again linearly combined to give output unit activations values. This is further transformed using another non-linear function like sigmoid.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

Where $k = 1, \ldots, K$, and K is the total number of outputs.

In case of multi class. We can use soft max. The number of hidden layers can be adjusted.

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

This is similar to a multi-layer perceptron. Since this does not have closed form solution iterative weight update method is used to update each of the weights. The error is calculates using back-propagation technique.

Error back propagation

The neural network used is a feed forward type of network and weights are adjusted in two stages. First assuming weights error is found and derivatives of error function at each layer are calculated. These are used to update weights where we consider network flow backwards called as back propagation.

Applying chain rule to derivatives of error function

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j.$$

&

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

This gives

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i.$$

Where,

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

What is Naïve Byes Classification?

The Naïve Bayes classifier is probabilistic approach for classification based on Bayes theorem assuming independence of features.

The algorithm relies on simple logic that posterior probability of event occurring is proportional to the product of likelihood and prior. It is suited for categorical and independent features.

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}.$$

The classifier calculates the posterior probability of the input vector containing features F1 to Fn belongs to class C1, C2 etc. The classifier labels the input as the class which has the maximum probability this is known as the MAP (Maximum Aposteriori Probability) approach.

$$\mathrm{classify}(f_1,\dots,f_n) = \underset{c}{\operatorname{argmax}}\ p(C=c)\prod_{i=1}^{n} p(F_i=f_i|C=c).$$

APPROACH

1. Preprocessing
   The data from given input files residing in ‘features_train’ was loaded into Matlab appended with 1 for bias saved as ‘train.mat’ and test data as ‘test.mat’. These are loaded by the ‘load_data’ function. It returns the data and class labels. Optionally, PCA can also be called for dimension reduction if needed.Project2.m is main file which handles all calls.1-K coding scheme followed.

2. Logistic Regression
   Data is fetched from load function and passed to ‘train_lr’ to perform training. Iterative gradient descent method was used to update weight vectors. The soft max activation function was used. First all weight vectors were initialized to zero and with eta set as 0.0001 and the following update formula was used.

   W=W-(0.0001*(X’*err))

   The function returns calculated W that’s given along with class labels and test data to ‘test_lr’ which uses this W to predict y the classes and compare against class labels to give error rate. This function also writes output to ‘class_lr.txt’.

3. Neural Network
   Data fetched from load is passed to 'train_nn' function that process the class labels and train data to generate neural network with 1 hidden layer using tanh activation and softmax for output activation function. The number of hidden neurons i.e. M is also passed to this function. The two weight vectors are updated iteratively using back propagation. Stochastic approach was used here. These were then used by test_nn for testing.Output written to 'classes_nn.txt'.

```
W2=W2-eta*(deltaK*z');
W1=W1-eta*(deltaJ*X(r,:));
r is one of the randomly chosen inputs
```

4. Naïve Bayes implementation also divides the data into train and test sets. First the class priors are calculated for each class by the following using Naïve Bayes class from Matlab. It has two function NaiveBayes.fit that takes input/train data and class labels as input to form a model. Output labels are assigned using the predict function which takes model and test data as input. The 'nbc_model' function take both train and test data and performs error evaluation. The final classes are written to 'classes_nbc.txt'.

5. Error evaluation

   Error rate was evaluated by using the following formula

$$Error\ rate(\%) = \frac{misclassified}{total} * 100$$

RESULTS

The weight parameters were tuned to obtain result in case of neural networks and logistic regression and to improve speed PCA was performed before computing Naïve Bayes. A low learning rate was chosen to avoid over-fitting. Random initialization used to obtain better results and avoid entering a poor local minimum.

The following optimum results were obtained-

Algorithm	Learning rate (if any)	M (inner dimensions/neuron)	Error Rate in %
Logistic	0.0001	-NA-	2.1333
Neural Network	0.00043	260	5.6000
Naïve Bayes(Matlab)	-NA-	-NA-	5.3333 (PCA top 100)

RESULT TUNING

Logistic Regression

Error-Rate (%)	Learn rate (ɳ)
2.2667	0.00005
2.13	0.0001
3	0.001
3.8667	0.0017
25	0.0023

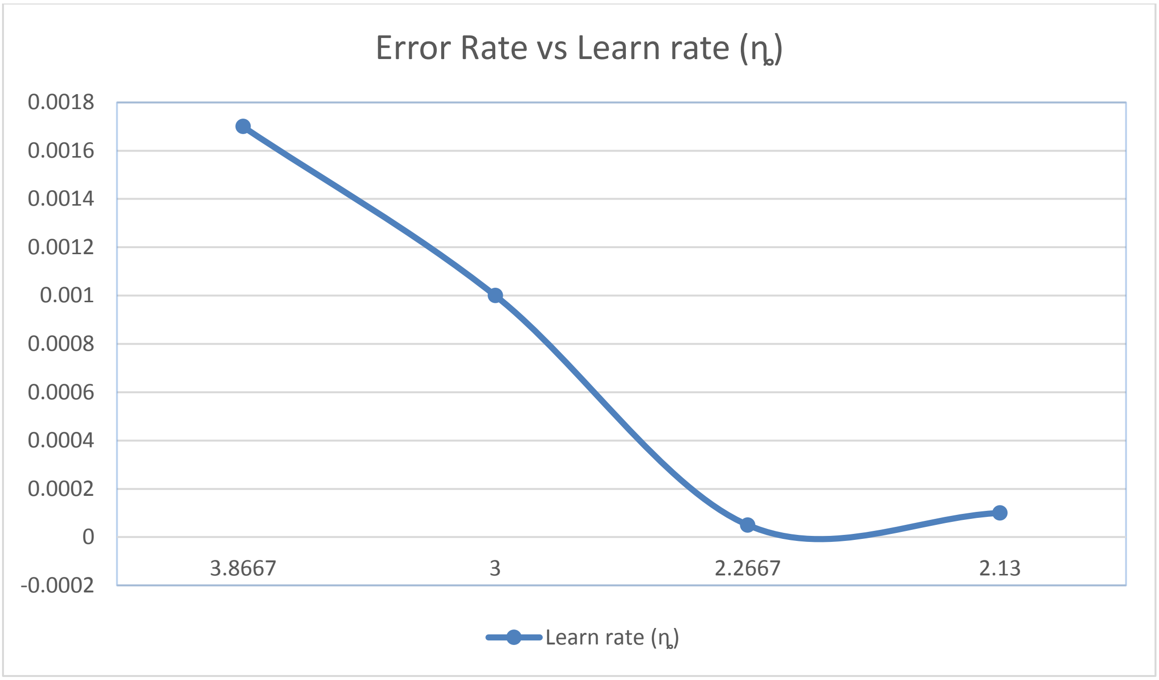

Figure 1 Error Rate (%) against eta

Neural Network

Learn rate (ɳ) for both layers	M-no of Neurons	Error-Rate (%)
0.00043	260	5.3333
0.00043	100	19.9149
0.00043	20	35.5774
0.0002	20	45.6713
0.0001	20	57.4427

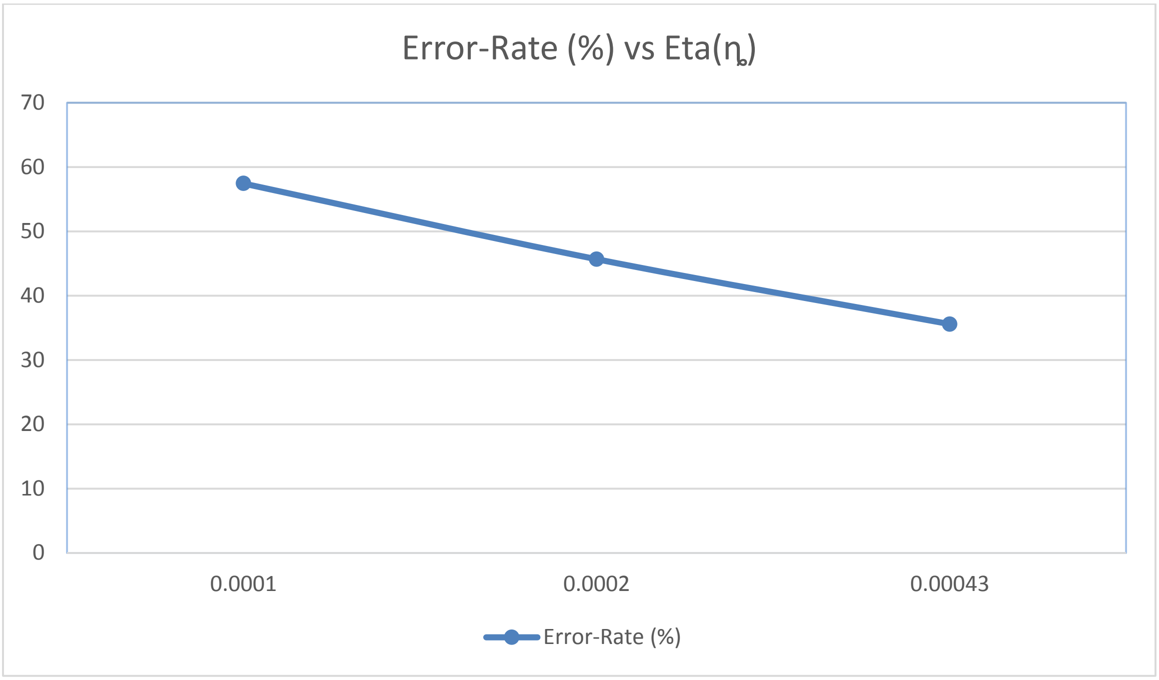

Figure 2 Error Rate (%) against eta

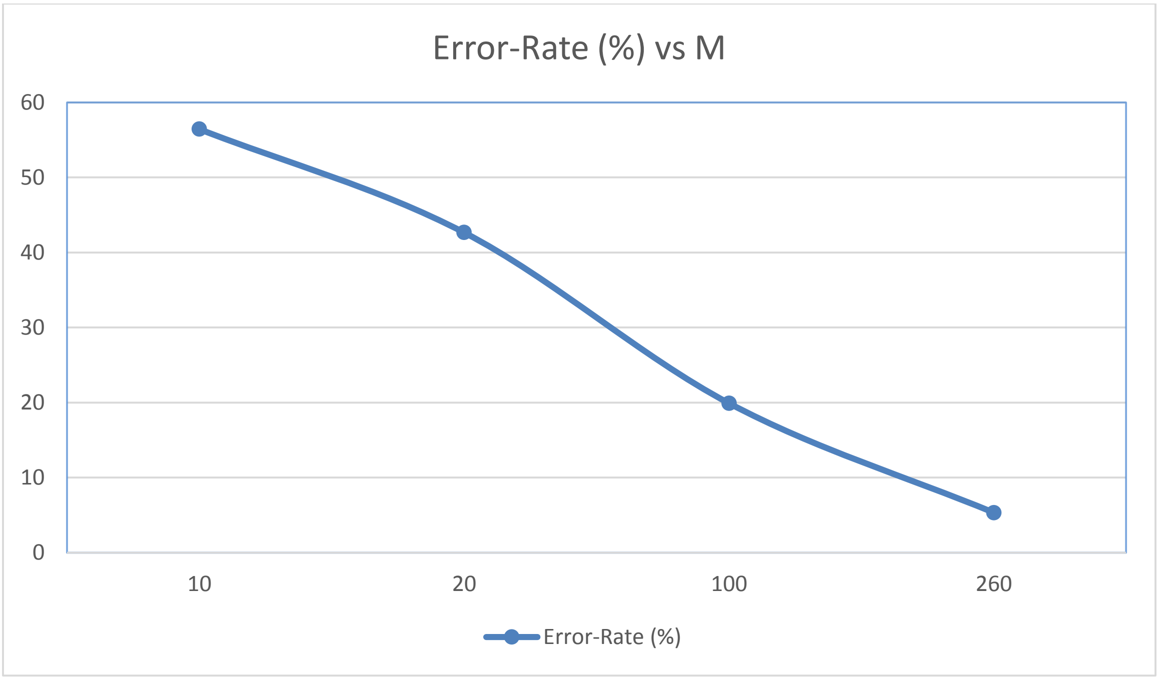

Figure 3 Error Rate (%) against M number of neurons

ANALYSIS

In this section, we discuss the results obtained, observations drawn

Results

1) Neural Network took more time for training due to two W matrices being used. In fact too many neurons can model better but considerably slow down performance.
2) Logistic regression was found to give better results under given conditions than other two.
3) In case of logistic regression reducing learning rate and using batch processing was found to give lower error rate than stochastic. This means large iterations are needed under stochastic to achieve low error-rate.
4) Random initialization is necessary for weights in neural networks else it may converge to some global minimum resulting in huge misclassification rate. Also initialization should be to a very low value to avoid tanh=1 and hence deltaJ=0.
5) Adding a momentum component may help moving around away from local minima.

DISCUSSION

In this section, we discuss the advantages, challenges and possible improvements of the Classification algorithms used.

ADVANTAGES

1) Naïve Bayes is easier to implement.
2) Logistic Regression in general can handle a variety of data and tuning parameters good results can be obtained
3) Neural network has better control over training due to a number of tuning parameters. Changing the number of layers and adjusting weights it possible to work with almost any type of data.

CHALLENGES OF CLASSIFICATION ALGORITHMS

1) Tuning multiple parameters to get desired accuracy
2) Deciding number of layers in neural network
3) Reducing time taken

IMPROVING

1) Hessian Matrix could have been used to give a one-step accurate solution.
2) Better feature extraction to reduce dimensionality of data.
3) Adding Momentum in case of neural networks.
4) Distributed approach to tackle very large datasets.
5) Better performance evaluation measures like f-measure, kappa etc. should be used to describe model performance.

CONCLUSION

REFERENCES

[1] http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

[2] http://www.willamette.edu/~gorr/classes/cs449/Momentum/momentum.html

[3]Christopher Bishop – Pattern Recognition and Machine learning