CS584: Assignment 3

Pooja Mankani
Department of Computer Science
Illinois Institute of Technology
04/03/2016

Abstract

The report illustrates the implementation of Discriminative learning using Logistic Regression and Multilayer Perceptron on synthetic and real data sets. The 10 fold cross validation for all the implementations is considered and the training error and testing error is evaluated. A custom model is created and the corresponding python built in model is also created using sklearn package in python. The errors of each of the model is compared and analyzed.

1. **Problem Statement**

   The following problems are considered in the implementation:

   a. **Logistic Regression:**
      - Logistic regression measures the relationship between dependent variable and independent variable by estimating the probabilities using a logistic function.
      - The logistic function is the sigmoid function that calculates the probability of the sample present in a particular class.
      - The highest value of the sigmoid function across the classes helps to put the sample in that class.
      - Hence based on the sigmoid values we can define the decision boundary that clearly separates the classes from each other.
      - The gradient descent approach is applied to get the correct parameters in logistic regression updating the values based on the error obtained during each run.
      - Based on the gradient the optimal value of the parameter is calculated keep the cost function as minimizing the error.
      - This approach is considered while implementing the solution on the IRIS and MNIST dataset.

   b. **Multilayer Perceptron:**
      - The input layer is mapped to the output layer based on the hidden layer into consideration.
      - The hidden layer needs to be initialized with the weights and the bias as required.
      - The weights and bias are updated on every iteration to map the input to the output. The backward propagation is then applied and the outputs are mapped back to the inputs.
      - The error is calculated and then based on the error calculation the model is updated.
      - This problem is applied to the IRIS and MNIST dataset and the solution is implemented based on the back propagation technique.

2. **Proposed solution**

   a. **2-class Logistic Regression :**

   - The logistic regression is applied on IRIS and MNIST datasets considering only two classes.
   - First the sigmoid of the feature matrix is calculated using the formula

     Sigmoid= 1/ 1+exp(-theta.transpose , X) – h(theta(x))
   - The log likelihood is calculated using the sigmoid function for Y with theta.transpose and X.

     L(theta)=sum (i from 1 to m) (y(i) – h(theta(x)) x(i)
   - The gradient is calculated using perceptron error formula which will be used to update the theta on every iteration.

     Theta= theta – learning_rate (sum i from 1 to m) (h(theta(x(i)) – y(i)) x(i)
   - The values are predicted using the discriminant function. Once the optimal theta is obtained, the sigmoid of X_test is calculated with respect to theta. Based on the sigmoid value which forms the decision boundary and discriminant. In case the discriminant is greater than 0.5, X_test is classified in class 1 else it is classified in class -1
   - Cross validation is applied for 10 folds and fmin_bfgs is used to find the minimal theta based on the likelihood and gradient cost functions.
   - **Discriminant:**

     if discriminant > 0.5

     y_predicted = 1

     else

     y_predicted = -1
   - The accuracy, confusion matrix, precision, recall, F1 measure is calculated

   b. **2-class Logistic Regression with non-linear input features:**

   - The 2-class logistic regression with non-linear input is applied to IRIS and MNIST dataset.
   - The feature matrix is converted into non-linear input matrix using the poly_fit sklearn library.
   - Once we have multiple features in the dataset the same steps that were applied in logistic regression can be used.
   - The sigmoid along with the log likelihood and the gradient is calculated. The discriminant function will then decide the class based on the value.
   - The accuracy, confusion matrix, precision, recall and F1 measure are calculated as required.

   c. **K-class logistic Regression:**

   - The k-class logistic regression is applied on IRIS and MNIST dataset.
   - The sigmoid, likelihood and gradient are calculated as required.
   - However, in this case since there are k classes in the y matrix the Y matrix is first converted such that in case the sample belongs to a particular class the other values for that sample is left as 0. This helps in calculating the likelihood and the gradient
   - Moreover, the theta values for each of the classes are updated in every iteration based on the update that is calculated by the gradient function.

- The gradient function takes the Y and theta.transpose, X values into consideration while calculating the update to the previous X.
- Based on this function fmin_bfgs is used to calculate the optimal value of theta for all classes.
- The discriminant function in this case is the argmax value across each of the classes. For the classes, for the maximum value the index is calculated which is the class of the sample.

  y_predicted = argmax(theta)
- The theta values are calculated for each class. The only difference in this is that the predicted class is the argmax of theta(j) * X

  Yhat = argmax h(theta(j) X) = argmax (j) h(theta(j) (x))

d. Multilayer Perceptron:

- A hidden layer of size 3 is considered. The input and output dimensions based on the dataset. This is executed on both IRIS and MNIST dataset.
- A model is created based on the weights and the bias of the neurons. These are then updated on every iteration based on the error obtained in the previous iteration.
- The value of each neuron is calculated as the weight of the neuron and its value from the previous layer.
- The back propagation is used which calculates the local gradients and starting from the output neurons going till it reaches the input layer.
- The error is now the difference between the desired output vector and the actual output vector.
- The delta is now the derivative of the neuron multiplied by the error calculated.
- The weights are adjusted based on the error and the delta values.
- The algorithm details are present in the Solution for Q 2(A)

e. Comparison and Analysis of results:

- The accuracy of the custom and the built in model is compared and the analyzed.
- Also the accuracy across the models is compared with each other and the better model that fits the dataset concluded.

3. Implementation Details:

a. **Design issues:** The following design issues were witnessed:

- The polynomial regression model for multiple feature is difficult to plot and cannot be plotted in a two dimensional space since there are multiple features associated to it with a quadratic function.
- The matrix for polynomial with respect to higher degree is difficult to obtain manually. This is complex since the combination of features is difficult to construct along with the order of the same. For e.g. if the degree is 2 and the number of features is 2 then the various feature combinations would be: **[1 X1 X2 X1^2 X2^2 X1X2].** Evaluating these combinations is difficult and complex.
- In case multiple records for large data sets are given as a input, for e.g. consider a data with billion records, in this case if we store all the data extracted from the records into variable then there could be a possibility of memory error within your

system. This is because it would take up lots of memory to store data within systems.

b. **Problems faced:** The following issues were faced:

- Cross validation: The implementation of cross validation to retrieve the training and testing indices was complex to obtain using python array and matrix manipulation. Hence in this case KFold cross validation of sklearn package was used to evaluate the same.
- Features in higher dimensional space: The multiple combination of features in the higher dimensional space is complex to evaluate using python functions. Since for different degrees and different number of features the combinations become complex. Hence in this case PolynomialFeatures of the sklearn.preprocessing package is used to evaluate the same.
- The model evaluated by gradient were resulting in high errors hence the data was required to be normalized. Thus the formula was updated such that the difference between the actual and the predicted value was divided by the number of samples.

c. **Implementation Details:**

Python Notebook	**Analysis**	**Command**
Assignment_3 .ipynb	2-class Logistic Regression	input_data=read_input('iris.dat') X=create_feature_matrix(input_data) Y=create_y_matrix(input_data) do_cross_validation(X, Y,[1,2],10, False, 0, verbose=True)
Assignment_3 .ipynb	2-class Logistic Regression with non linear input	input_data=read_input('iris.dat') X=create_feature_matrix(input_data) Y=create_y_matrix(input_data) do_cross_validation(X, Y,[1,2],10, True, 2, verbose=True)
Assignment_3 _MLR.ipynb	k-class Logistic Regression	input_data=read_input('iris.dat') X=create_feature_matrix(input_data) Y=create_y_matrix(input_data) do_cross_validation(X, Y,[1,2,3],10, verbose=True)
Assignment_3 _MLP.ipynb	Multilayer Perceptron for 3 class-classification	input_data=read_input('iris.dat') X=create_feature_matrix(input_data) Y=create_y_matrix(input_data) do_cross_validation(X, Y,[1,2,3],10, verbose=True)

4. **Results and Discussions:**
   a. **2-class Logistic Regression: (IRIS Dataset)**
      - **Initial Value of Theta: 0**
      - **Theta calculated after gradient descent:**

First Column	Feature 1	Feature 2	Feature 3	Feature 4
-5.45390546	-1.72640044	-11.03607063	12.40692912	16.55051744

      - **Looking at the theta values we can say that Feature 4 is dominant and plays a very important role to predict the analysis**
      - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[50 0]
[0 50]]

      - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

   b. **2-class Logistic Regression: (MNIST Dataset)**
      - **Initial Value of Theta: 0**
      - **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[980 0]
[0 3800]]

      - **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0.99581589958159
Precision	0.99730458221024254
Recall	1
F1 Measure	0.99746192893401009

c. 2-class Logistic Regression with non-linear features: (IRIS Dataset)

- **Initial Value of Theta: 0**
- **Theta calculated after gradient descent:**

[-3.2075114 -3.2075114 -3.1052183 -11.11058757 17.87403261 7.85228485 -3.2075114 -3.1052183 -11.11058757 17.87403261 7.85228485 -1.04978591 3.76090376 2.88489049 1.64431077 -1.44829682 0.47494692 0.6250598 -6.3425588 -4.3439567 -3.09739348]

- **Since there are multiple features due to the degree each of them plays an important role in predicting the analysis. For poly_degree=2 and number_of_features = 4 we get a total of 21 features.**

- **Overall Parameters computed:**

Parameter	**Result**
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

Confusion Matrix: [[50 0]
[0 50]]

- **Average Parameters Computed after 10 folds:**

Parameter	**Result**
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

d. k-class Logistic Regression: (IRIS Dataset)

- **Initial Value of Theta: 0**
- **Theta calculated after gradient descent:**

Class	First Column	Feature 1	Feature 2	Feature 3	Feature 4
Class 1	-2.44482501	-20.18460599	22.629431	-4.6648963	1.1010832
Class 2	3.5638131	-11.88180094	2.60127544	9.28052551	17.11865031
Class 3	-3.83475439	-13.28389592	7.88103212	5.22011308	-13.1011452

- **For every class there would be a value of theta that would be updated on every iteration in the gradient descent. Moreover, in case of class 1, feature 2 dominated, in case of class 2, feature 4 dominates and in case of class 3 feature 2 dominates the classification.**
- **Overall Parameters computed:**

Parameter	Result
Accuracy	0.986667
Precision	0.986667
Recall	0.986667
F1 Measure	0.986667

- **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0.986667
Precision	0.984920
Recall	0.98
F1 Measure	0.979604

e. k-class Logistic Regression: (MNIST Dataset)

- **Initial Value of Theta: 0**
- **Overall Parameters computed:**

Parameter	Result
Accuracy	1
Precision	1
Recall	1
F1 Measure	1

- **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0. 98601455868971788
Precision	0. 98613362687732153
Recall	0. 98601455868971788
F1 Measure	0. 98600496677486082

f. Multi-Layer Perceptron: (IRIS Dataset)

- **Initial Value of W1 and W2 is set as random numbers**
- **Initial Value of bias is set as zero**
- **Number of passes = 200000, learning rate: 0.01**
- **Overall Parameters computed:**

Parameter	Result
Accuracy	0.666667
Precision	0.5
Recall	0.666667
F1 Measure	0.555556

Confusion Matrix: [[50 0 0]
[0 0 50]
[0 0 50]]

- **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0. 8666666666666667
Precision	0. 76000000000000001
Recall	0. 8666666666666667
F1 Measure	0. 80740740740740735

g. Multi-Layer Perceptron: (MNIST Dataset)

- **Initial Value of W1 and W2 is set as random numbers**
- **Initial Value of bias is set as zero**
- **Number of passes = 200000, learning rate: 0.01**
- **Overall Parameters computed:**

Parameter	Result
Accuracy	0. 177203
Precision	0. 141450
Recall	0. 177203
F1 Measure	0. 109950

Confusion Matrix: [[11 5 0 5399 0 488 0 0 0 0]
[3 6758 0 1110 0 6 0 0 0 0]
[61 504 0 6227 0 198 0 0 0 0]
[45 1733 0 5252 0 110 0 1 0 0]
[2593 33 0 662 0 3536 0 0 0 0]
[138 415 0 5555 0 205 0 0 0 0]
[25 48 0 6346 0 457 0 0 0 0]
[6063 324 0 295 0 610 0 1 0 0]
[28 687 0 5793 0 317 0 0 0 0]
[1732 192 0 601 0 4433 0 0 0 0]]

- **Average Parameters Computed after 10 folds:**

Parameter	Result
Accuracy	0. 21231884057971015
Precision	0. 13635509293144182
Recall	0. 21231884057971015
F1 Measure	0. 14532333917348036

Conclusion:

- **For IRIS dataset 2class logistic regression gives a good performance. Also the theta values show that which feature is the most dominant while predicting the results. The same pattern is seen for MNIST dataset. However, in this case there are multiple features.**
- **For 2 class non-linear inputs the logistic regression gives good performance for IRIS dataset. However, for the MNIST dataset it takes a large amount of time to predict the values since there are around 784 features in the original set and converting all these features to a non-linear space is very time consuming.**
- **K class logistic regression shows good performance on both IRIS and MNIST dataset. The model works well for both the datasets and evaluates it as required.**
- **The model created for multilayer perceptron works well for IRIS dataset however for the MNIST dataset it does not show good results. The accuracy is very poor for the MNIST dataset. This is because the weights of the neurons are taken as random values and these do not change in a desired manner in while we move from the input layer to the output layer using the hidden layer in question.**
- **Hence in this case it shows lower performance for MNIST dataset compared to the sklearn performance of 88% accuracy.**

Data Set:

GDA Analysis: IRIS dataset: http://archive.ics.uci.edu/ml/datasets/Iris

MNIST Dataset from sklearn

References:

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

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

http://blog.refu.co/?p=931

http://scikit-learn.org/dev/modules/neural_networks_supervised.html

http://www.di.ubi.pt/~lfbaa/pubs/NN2008.pdf