INF 552: Assignment 4
Supervised Learning

Priyambada Jain (priyambj@usc.edu)
Sai Sree Kamineni (skaminen@usc.edu)
Varad Kulkarni (vdkulkar@usc.edu)

Part 1: Implementation

Please note all the result files and plot is in the Results folder to avoid confusion.

Language used – Python 2.7

1. Perceptron Learning

Command to run – python Perceptron.py
Input File -classification.txt (In the same directory as the python file)

Data Structures Used: Numpy array and Lists

Result: Output file "Perceptron_Result.txt" has the data, labels and the predictions.
Output file " Perceptron_Weights.txt " has the weights.
We computed and printed the accuracy of the model as well.

Code Level Optimizations: For faster computations used the Numpy library.

Notations: X_train : Feature vector
Y_train : Actual Classes
predictions: Predicted Classed using the implementation

Algorithm:

1) Initialize the weights and the threshold. Weights may be initialized to 0 or to a small random value.
2) Now, make a prediction for each data point in X_train :
   prediction = np.dot(weights.T,X_train[data_points_iterator])
3) Check for violated constrains and update the weights using the gradient descent
   weights-= le * X_train[data_points_iterator]
   where le is the learning rate.
4) Iterate through Step 2 and Step 3 until the convergence i.e since this is a promise problem the convergence implies no more violated constraints.
5) Now make the predictions for the data points using the weights updated until convergence.

Challenges Faced: Deciding the optimal value of the learning rate.

2. Pocket Algorithm

Command to run – python pocket.py

Input File -classification.txt (In the same directory as the python file)

Data Structures Used: Numpy array and Lists

Result: Plot is created after the program reaches the max_epoch at 7000 iterations.
Named as **Pocket_Result.png**.

Code Level Optimizations: For faster computations used the Numpy library.

Notations: X_train : Feature vector
Y_train : Actual Classes
predictions: Predicted Classed using the implementation

Algorithm:

1) Initialize the weights and the threshold. Weights may be initialized to 0 or to a small random value.
2) Now, make a prediction for each data point in X_train :
   prediction = np.dot(weights.T,X_train[data_points_iterator])
3) Check for violated constrains and update the weights using the gradient descent
   weights-= le * X_train[data_points_iterator]
   where le is the learning rate.
4) Also in the epoch each a track of violated constraints for each
   violations+=1
5) Assign the violations to the epoch:
   constraints[i] = violations
6) Iterate through Step 2, Step 3 , Step 4 and Step 5 until the epochs reaches 7000 i.e since this is a not promise problem the convergence implies until the max epochs are reached.

Challenges Faced:
Deciding the optimal value of the learning rate and also the running time for the 7000 iterations.

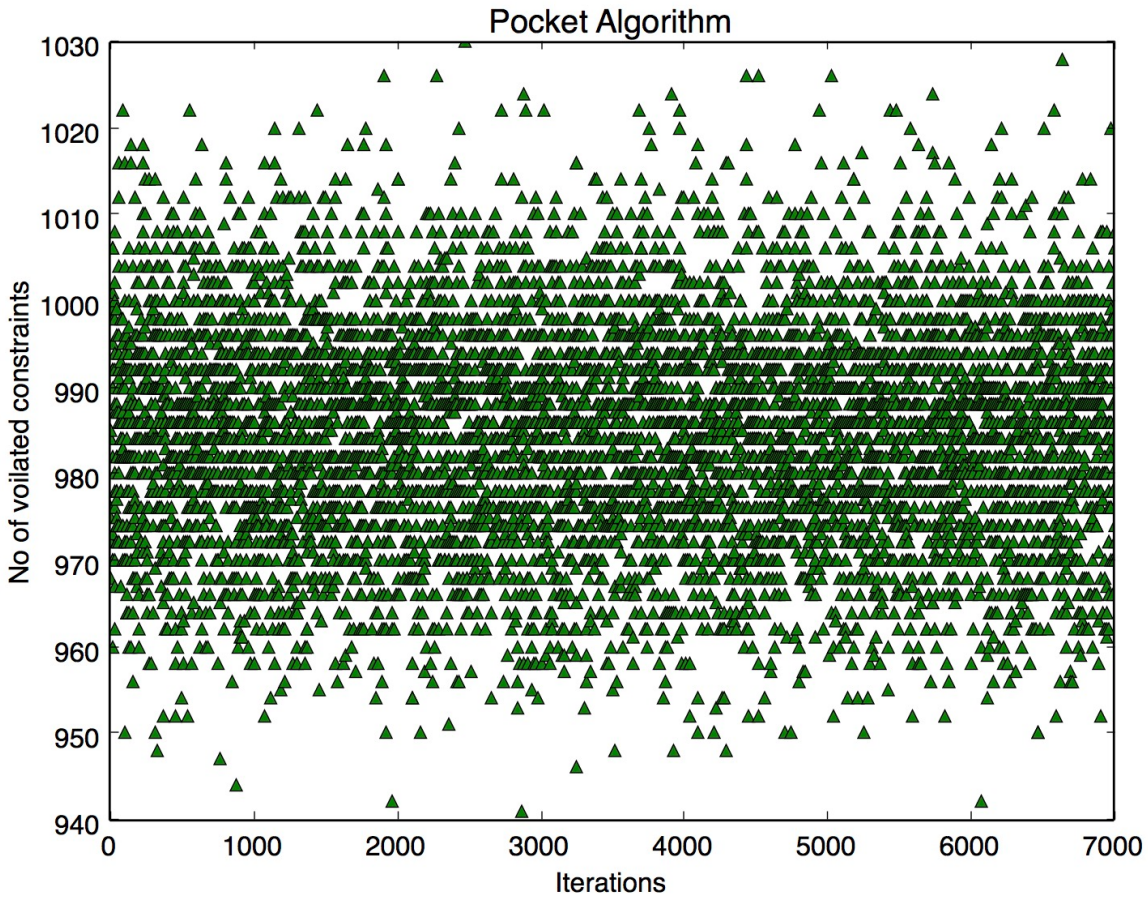

Figure 1

3. Linear Regression

Command to run – python pocket.py
Input File - linear-regression.txt (In the same directory as the python file)

Data Structures Used: Numpy array and Lists

Result: Output file "Linear_Regression_Output" has the data, labels and the predictions.
Output file "Linear Regression_Weights" has the weights.
Also printed the RMSE for the model
RMSE for the Linear_Regression model: 0.00724122625789 (on the terminal)

Code Level Optimizations:
numpy.linalg.inv() to calculate the inverse of the product of the vectors.

Algorithm:

1. Read data points and add one more column of ones to the data as X0.
2. Calculate the optimal weights using the formula: W = (X.X.T)^-1.X.T.Y
3. Using the new weights calculate the predictions using W.T.X.

Challenges Faced:
Deciding the optimal value of the learning rate.

4. Logistic Regession

Command to run – python logistic.py
Input File – classification.txt (should be in the same directory as code)
Output File – Prints weights and Accuracy to screen.

Weights [-0.02574055 -0.18078949 0.10880617 0.07188262]
Accuracy 0.528

Data Structures used – Numpy Arrays and lists.

Code Level Optimizations – Used Numpy library functions like sign, exp, dot to perform operations on the arrays.

Challenges Faced:
Initializing random weights was a problem and choosing a value for 'eta'.

Algorithm
Used Gradient Descent

1) Extracted x & y coordinates form file and appended 1 as 1st point in x coordinate
2) Choose random weight vector and eta
3) Calculated Delta E_{in} and updated weights
4) Repeated step 3, 7000 times
5) Calculated Y values using sigmoid function and computed the accuracy

Part 2: Software Familiarization

We have used **Python sklearn** library to compare against our implementation for Perceptron, Linear_Regression and Logistic Regression.

Linear Classification

1) **sklearn linear_model SGDClassifier model**:

Parameters Used : fit_intercept =False

Applied the model on the data points.
Checked the accuracy using ‘score’ attribute.

Accuracy from the Linear SGDClassifier sklearn : 0.9335
Accuracy from the Perceptron Implementation: 1 (printed on the terminal)

Linear_Regression

2) **sklearn linear_model LinearRegression model**:

Parameters Used : fit_intercept =False
Applied the model on the data points.
Checked the RMSE using mean_squared_error.

('RMSE of the sklearn Linear Regresssion ', 0.013869591053288739)
Variance score: 0.97225

RMSE for the LinearRegression Implementation: 0.00724122625789

Logistic Regression

3) **sklearn linear_model LogisticRegression model**:

Parameters Used : fit_intercept =False
Applied the model on the data points.
Checked the accuracy using ‘score’ attribute.

Accuracy from sk-learn: 0.5295
Accuracy from Logistic Regression Implementation: 0.528 (printed on the terminal)

Part 3: Applications

Applications of linear classification

1. **Direct Marketing**: Reduce cost of mailing by targeting a set of consumers likely to buy a new cell-phone product.

2. **Fraud Detection**: Predict fraudulent cases in credit card transactions.

3. **Customer Attrition/Churn**: To predict whether a customer is likely to be lost to a competitor

4. **Sky Survey Cataloging**: To predict class (star or galaxy) of sky objects, especially visually faint ones, based on the telescopic survey images (from Palomar Observatory).

Applications of linear regression

1. **Trend line**
   A **trend line** represents a trend, the long-term movement in time series data after other components have been accounted for. It tells whether a particular data set (say GDP, oil prices or stock prices) have increased or decreased over the period of time. A trend line could simply be drawn by eye through a set of data points, but more properly their position and slope is calculated using statistical techniques like linear regression. Trend lines typically are straight lines, although some variations use higher degree polynomials depending on the degree of curvature desired in the line.
2. **Epidemiology**
   A regression model in which cigarette smoking is the independent variable of interest, and the dependent variable is lifespan measured in years. variants of regression analysis such as instrumental variables regression may be used to attempt to estimate causal relationships from observational data.
3. **Finance**
   The capital asset pricing model uses linear regression as well as the concept of beta for analyzing and quantifying the systematic risk of an investment. This comes directly from the beta coefficient of the linear regression model that relates the return on the investment to the return on all risky assets.
4. **Economics**
   Linear regression is the predominant empirical tool in economics. For example, it is used to predict consumption spending, fixed investment spending, inventory investment, purchases of a country's exports, spending on imports the demand to hold liquid assets, labor demand and labor supply.
5. **Environmental science**
   Linear regression finds application in a wide range of environmental science applications. In Canada, the Environmental Effects Monitoring Program uses statistical analyses on fish and benthic surveys to measure the effects of pulp mill or metal mine effluent on the aquatic ecosystem

Applications of logistic regression

1. Image Segmentation and Categorization
2. Geographic Image Processing
3. Handwriting recognition
4. Healthcare: Analyzing a group of over million people for myocardial infarction within a period of 10 years is an application area of logistic regression.
5. Prediction whether a person is depressed or not based on bag of words from the corpus seems to be conveniently solvable using logistic regression and SVM.
6. If you want to predict the probability that a certain person will vote for a democrat/republican or not.
7. Developing a forecasting model to find volume of houses on sales in a month given economic factors, seasonality and other dimensions
8. Finding out drivers of retail product sales as a function of spend across media channels, economic factors and competitor actions

References

1) https://www.stat.ubc.ca/~rollin/teach/536a/confEtc.html#(1)

2) http://dni-institute.in/blogs/scenarios-multiple-regression-applications/

3) http://www.tandfonline.com/doi/abs/10.1080/01431160412331331012

4) http://www.sciencedirect.com/science/article/pii/S0023643811001630

5) https://www.ismll.uni-hildesheim.de/lehre/ml-08w/skript/classification1.pdf

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

7) http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html