Virginia Tech
Invent the Future
The Bradley Department Electrical & Computer Engineering

ECE 6524 – Deep Learning

Homework Assignment #3: Logistic Regression

(1) Logistic regression for two-class classification:

a. N pairs of data points: ($\mathbf{x}_i$, C_i), i = 1, …, N, where $\mathbf{x}_i$ is the feature vector, and C_i is the binary class label (i.e., C_i is either 0 or 1;).

b. Logistic regression is a (special) linear classifier: $y_i = \boldsymbol{\beta}_1^T\mathbf{x}_i + \beta_0$, but y_i is interpreted as log-odds, i.e., $y_i = \log(p / (1 - p))$, where the probability of class 1 ($C_i = 1$) is p, and the probability of class 0 ($C_i = 0$) is 1 - p. The prediction is based on sigmoid(y_i) = $1/[1+\exp(-y_i)]$; if sigmoid(y_i) >= 0.5, $\mathbf{x}_i$ is predicted as a data sample from class 1; otherwise, class 0.

c. The likelihood function is: $\Pi_{i=1,\ldots,N} (p(\mathbf{x}_i))^{C_i} (1- p(\mathbf{x}_i))^{(1 - C_i)}$, where $p(\mathbf{x}_i)$ is the probability of $\mathbf{x}_i$ from class 1.

Prove that the log-likelihood function can be rewritten/formulated as $\sum_{i=1,\ldots,N} \{C_i (\boldsymbol{\beta}_1^T\mathbf{x}_i + \beta_0) – \log[1 + \exp(\boldsymbol{\beta}_1^T\mathbf{x}_i + \beta_0)]\}$, and **compute the gradient** of the log-likelihood function.

(2) As instructed in Homework Assignment #0, we can build a simple Python program to construct a logistic regression classifier to test on simulate data.

a. Generating simulated data (two-dimensional data of two classes with some overlap; as illustrated in Fig. 1) by sampling from multivariate normal distributions.

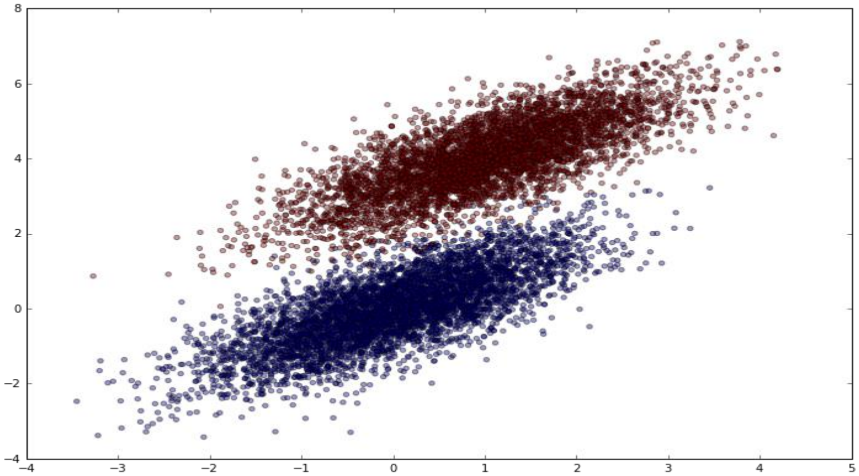

Fig. 1. An example of simulated data: for example, data points in brown are in class 1, and data points in blue are in class 0.

b. Build a logistic regression model using the scikit-learn library. Below is an example for building a logistic regression model:

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(fit_intercept=True, C = 1e15)
clf.fit(simulated_features, simulated_labels)

print clf.intercept_, clf.coef_
```

(3) Implement a two-class logistic regression classifier with the **stochastic gradient descent** (SGD) algorithm, where the **loss function** is the log-likelihood function. Specifically, you are asked to 'replace' the `clf.`fit() method with your own implementation of the method such as the following:

```
def logistic_regression(features, class_label, num_steps, learning_rate)

     # TO DO: YOUR IMPLEMENTATION

     return weights
```

Verify that the weights estimated from SGD are in consistent with the weights obtained from the clf.fit() method.

(4) Investigate the impact of learning rate on the convergence of the algorithm. (Hint: you may plot the log-likelihood over time to show the convergence.)

(5) Report the prediction performance on the simulated data.

(6) Apply your logistic regression classifier to some real data of your choice (like the Iris data and MNIST data (http://yann.lecun.com/exdb/mnist/)) in predicting any two classes. [Note that you need to extend your implementation to handle n-dimensional data.] Experiment with different choices of two classes as many as possible if you use a multiclass data set, and report the overall performance of your logistic regression classifier.

(7) Discuss about how you can extend your implementation to classify multiclass data (i.e., the number of class > 2).

You need to prepare **a written report** (in the pdf format) including the following sections: (1) Introduction or Problem Statement, (2) Method(s) or Approach(es), (3) Experimental Results, and (4) Discussion & Conclusion.

In addition, you need to attach your **implementation codes** as separate files to the report.