Machine Learning with Networks *ECEN765.600, Fall 2017*

Programming Assignment #2: Optimization & Linear Models

Due date: November 3rd, 2017 (Friday)

"On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work."

Signature:

Name:

0. Reading Assignments: The relevant content in UML Chapter 9&10, BRML Chapter 17

1. Implement the perceptron algorithm for logistic regression (30pts):

1. Use the data set with binary classes (`bclass-train` and `bclass-test`): Each row is one example with the first column as the class label $\{-1, 1\}$ and the rest as feature variables) You can replace the class label -1 by 0). You can find the data set from eCampus.
2. Run your logistic regression using the raw data, data that has been normalized to have unit l_1 norm, and data that has been normalized to have unit l_2 norm. Which seems to work best?
3. Plot error rates for the training, and test data as a function of iteration (both the raw predictions and the normalized predictions).
4. Repeat the previous tasks by modifying the perceptron algorithm to use the averaged weights instead of the original updated weights over the iterations. The averaging extension of the perceptron algorithm is to maintain all weight vectors during the iterations and the final weight vector is the accumulative average weight vector.
5. Compare two versions of the perceptron algorithms. Discuss the differences.

2. Implement the gradient descent/ascent algorithm for logistic regression (30pts):

1. Use the same data set as in **1**. Run your logistic regression using the raw data, data that has been normalized to have unit l_1 norm, and data that has been normalized to have unit l_2 norm. Which seems to work best?
2. Plot error rates for the training, and test data as a function of iteration (both the raw predictions and the normalized predictions).
3. Compare the results with the ones from the perceptron algorithm. Which one is better? Why?

3. Locally Weighted Logistic Regression (40pts):

Implement a locally-weighted version of logistic regression, where we weight different training examples differently according to the query point. The locally weighted logistic regression problem is to maximize

$$l(\beta) = \sum_{i=1}^{N} w^i \Big\{ y^i \log f_\beta(x^i) + (1 - y^i) \log \left[1 - f_\beta(x^i)\right] \Big\} - \lambda \beta^T \beta,$$

where the last term is the regularization term as we discussed for the linear regression in class. (You can also implement the regularized logistic regression for **2**, which often can give more stable results using either

gradient descent or Newton's method.) You can set $\lambda = 0.001$; Or you can use the development data included in the data set to pick the best performing λ:

1. Compute the gradient $\nabla_\beta l(\beta)$ and the Hessian matrix $H = \nabla_\beta[\nabla_\beta l(\beta)]$;
2. Given a new test data point x, we compute the weight by

$$w^i = \exp(-\frac{\| x - x^i \|_{l_2}^2}{2\tau^2}),$$

where τ is the bandwidth. Use the same data set with binary classes as above (`bclass-train` and `bclass-test`) to implement **Newton's** method for this locally weighted logistic regression. Vary $\tau = \{0.01, 0.05, 0.1, 0.5, 1.0, 5.0\}$ to: (a) compute w^i's for each development/test sample using the formula above, (b) maximize $l(\beta)$ to learn β, (c) predict y based on $f_\beta(x)$ ($y = 1$ when $f_\beta(x) \geq 0.5$), and finally (d) plot the error rates with respect to τ and compare them with the ones obtained in **1**.

4. Course Project

1. Please upload your midterm reports to your GitHub repository by Nov. 10th, 2017. Please check the course project guideline if needed.