Machine Learning with Networks *ECEN765.600, Fall 2016*

Programming Assignment #2: Linear Models & Optimization

Due date: October 28th, 2016 (Friday) 4:10PM before class

"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 BRML Chapters 17 & Appendix

1. Implement the gradient descent/ascent algorithm for logistic regression (40pts):

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 in the course content.
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).

2. 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 **1**, 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

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

   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**.

3. Support Vector Machines (20pts):

In this part, we will experiment using other people's software. I suggest you use one of the SVM implementations available at `http://www.support-vector-machines.org/SVM_soft.html`. There are also MATLAB SVM implementations that you can use, including the toolboxes in BRMLToolBox (`http://web4.cs.ucl.ac.uk/staff/d.barber/pmwiki/pmwiki.php?n=Brml.Software`) and PMTK3 (`https://github.com/probml/pmtk3`) with discussions (`https://code.google.com/p/pmtk3/`). You might need to transform the data format.

Using the same binary data set, train the following SVMs (using just the training data): a linear SVM, and an RBF SVM (for (**extra credit** 10pts). For the linear SVM, try different values of C ranging in $0.25, 0.5, 1, 2, 4$. For the RBF SVM, try τ values (bandwidth) of $0.25, 0.5, 1, 2, 4$. Plot error rates on both the development data and test data for the different values of C. How many support vectors are used for each model? Should this increase or decrease with C (why?)?