Machine Learning with Networks	*ECEN765.600, Fall 2018*

Assignment #2: Linear Models & Optimization

Due date: October 19th, 2018 (Friday)

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

Signature:

Name:

1. Programming assignment (You are encouraged to program the specific machine learning algorithms by yourself. In case you are using the ones in existing packages, you would at least need to read the code!)

A) Implement the perceptron algorithm and logistic regression for classification: 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.

1. (10 pts) Run your logistic regression algorithm (by either gradient descent or more advanced local search methods) using the raw data as well as the data that has been normalized to have unit l_2 norm. Which seems to work best? Plot error rates for the training and test data as a function of iteration (both the raw predictions and the normalized predictions). Discuss the trends.
2. (10 pts) Repeat the previous tasks by implementing the perceptron algorithm.
3. (10 pts) Compare the logistic regression and perceptron algorithm. Discuss the differences.

B) Locally Weighted Logistic Regression: 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

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

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 **A)**, 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. (10 pts Math assignment) Compute the gradient $\nabla_\beta l(\beta)$ and the Hessian matrix $H = \nabla_\beta [\nabla_\beta l(\beta)]$;
2. (10 pts) 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 **A)**.

2. Math assignment

1. (10 pts) Suppose we modify the perceptron algorithm as follows: In the update step, instead of performing $\mathbf{w}_{t+1} = \mathbf{w}_t + y^i \mathbf{x}^i$ whenever we make a mistake, we perform $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y^i \mathbf{x}^i$ for some $\eta > 0$. Prove that this modified perceptron algorithm will perform the same number of iterations as the original perceptron algorithm and will converge to a vector that points to the same direction as the output of the original algorithm.

2. (10 bonus pts) Show that for least-square linear regression fits to data $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)\}$, the fitted lines go through the point $\frac{1}{N}\sum_{i=1}^{N}(x^i, y^i)$.