CMPS242 Machine Learning
Fall 17 Warmuth/Amid
handed out Th 10-12-17 **Homework Assignment 3** due Tu 10-24-17 by beginning of class

Programming homework. Work in groups of up to three.
No overlap to people you worked with in homework 1!

You are provided with a ham/spam data set (train.csv, test.csv)

Your task is to develop a spam detector using logistic regression. Implement all in Python using Jupyter Notebooks, which is a powerful way to document you work. You might want to prototype various parts in R, Matlab, or Octave.

- Tokenize you texts and apply the tf-idf transform (described in Section 4.2.3).
- **Implement in Python** (do not use the libraries) batch gradient descent logistic regression with regularizer $\lambda\|\mathbf{w}\|^2$ and learning rate $\eta = \eta_0 \times t^{-\alpha}$ where η_0 is a constant, t is the iteration number and $\alpha = .9$.
- Choose λ with 10-fold cross validation based on the classification accuracy. Report your results as in solutions to Hw 1: cross validation curve (with error bars if you can) plus table of results with best choices/results bolded.
- Output your Jupyter notebook as an html file and submit it to Canvas by beginning of class on the due date. You can overwrite your previous submits. So do a trial submit early on.

Tips:

- Partition your work. Some of you focus on getting the tf-idf tranform of your data. The second group should implement logistic regression on a single split and then implement cross validation.
- You can use the `nltk.corpus` library to remove the English stopwords. Always report all your steps in the notebook.
- Use the `decode_error='ignore'` option if you are using `CountVectorizer`. It ignores jibberish introduced by the detaction of the texts.
- Remember to normalize your data after applying tf-idf (there is an option for that). Always report all your steps in the notebook.

Extra Credit:

- Add a bias term but don't include the bias term in the regularization.
- Try different regularizers such as $\lambda\|w\|_1$.
- Use EG$^{\pm}$ instead of gradient descent.
- Implement Weighted Linear Least Squares and compare convergence speed and total computation time against Batch Gradient Discent
- Ditto for Stochastic Gradient descent