## Assignment 4.1: Linear Classification

For this assignment, you will conduct programming tasks in Jupyter Notebook and answer provided questions in a Google Doc or MS Word document. You will submit **2** documents to Blackboard:

1. A PDF file converted from your Jupyter Notebook.
2. A PDF file converted from your Assignment 4.1 Template Word Doc (document is linked in Blackboard assignment prompt).
   * This template has a copy of all short answer questions for this assignment.

As you go through the assignment, you will create several tables and figures. After you complete the programming section, use the tables and figures you generated to answer the following questions.

## Movie Data

For this assignment, you will work with the movie data. This data used here are a set of 25,000 movie reviews. For linear classification, this dataset gives you an example of using regularization on a dataset with many examples and many features (words).

The dataset for this assignment comes from this publication, Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). [Learning Word Vectors for Sentiment Analysis.](https://ai.stanford.edu/~amaas/papers/wvSent_acl2011.pdf) *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

## **Written Questions**

(short answer, 2-3 sentences):

In the simulated data section, you used the bootstrap to show you the distribution of the estimated accuracy of your classifier. You plotted this for three different classification methods. Based on this plot, do you think any of the three classification methods is significantly better than the other methods (on this particular dataset)?

**The log-loss classification had a much narrower distribution in the 88+% range compared to its counterparts. The hinge loss had an interval that was significantly larger than the log-loss, while the perceptron accuracy spanned an incredibly large interval. Therefore, log-loss was significantly better.**

When you had the simulated data and looked for the best regularization parameter, one method (l1 or l2) had a clear advantage over the other. When you used the same code and methods on the text data, was one method of regularization clearly better than the other? Why do you think this was the case? Elaborate.

**L2 Regularization did substantially better at each learning rate than L1. This is became L2 regularization shrinks feature weights while L1 removes them entirely. This method of regularization does not work for sparse data – most rows in the dataset contain zero values for a majority of the features, so an L1 would naturally remove them from the dataset, but this makes analysis difficult as the sparse features all still ultimately matter during classificaiton.**

## Programming Section

### Linear Classification

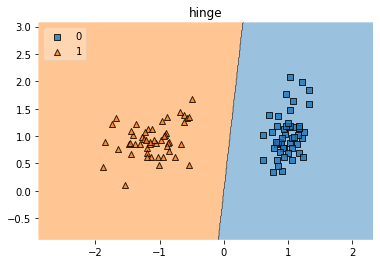
Install the package *mlextend*: <http://rasbt.github.io/mlxtend/installation/>

Next, you will do a few exercises to visualize the difference between the different linear classifiers.

Generate classification data using make\_classification from sklearn.datasets:

X, y = make\_classification(n\_features=2, n\_redundant=0, n\_informative=2,random\_state=1, n\_clusters\_per\_class=1)

Use SGDClassifier to train classifiers using different loss functions: log, hinge, and perceptron. Visualize the trained classifiers using plot\_decision\_regions(X, y, clf=model, legend=2)from the *mlextend* package. Plot the decision region for each of the three loss functions: a sample plot for hinge loss appears below.



Now, create a larger classification dataset. You will use cross\_val\_score from scikit-learn and compare this to bootstrap\_scores from *mlextend*.

Set up the simulated data as follows:

X, y = make\_classification(n\_samples=10000,n\_features=20, n\_redundant=0, n\_informative=20,random\_state=1, n\_clusters\_per\_class=1)

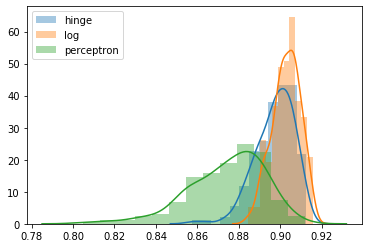
With your classifier in a variable called model (using SGGClassifier, as before), you can get the accuracies as follows:

scores = cross\_val\_score(model, X, y, cv=5,scoring=scoring)

bootstrap\_scores = bootstrap\_point632\_score(model, X, y, method='oob')

Create a table with the average of each cross-validation score and the average of the bootstrap scores.

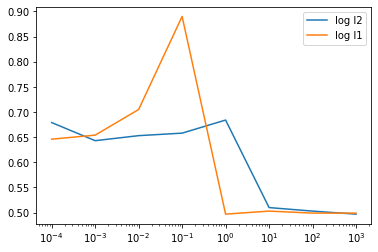
Finally, use distplot from the *seaborn* package to plot your bootstrap samples.



Finally, you will look at the importance of setting the regularization parameter. Create a database with only two informative features:

X, y = make\_classification(n\_samples=1000,n\_features=2000, n\_redundant=0, n\_informative=2,random\_state=1, n\_clusters\_per\_class=1)

Train a regularized classifier using ‘log’ as the loss function. Try both types of regularization ‘l1’ and ‘l2’ and sweep alpha over a range from: [0.0001,0.001,0.01,0.1,1,10,100,1000]. Use *fivefold* cross-validation to measure accuracy. Create a table of accuracy indexed by alpha, and use this table to plot the accuracy for both types of regularization. Your plot should look like this:



# **Large Scale Linear Classification**

Next, you will use data from a dataset of movie reviews.

Unzip the data in aclImdb\_v1.tar.

There are several formats of data here, but one simple way to load the data into python is to load the files individually into a python list, and then use tfidfVectorizer to convert the reviews into bag of words feature files. Make sure to use pd.DataFrame.sparse.from\_spmatrix to make a dataframe from the word features.

Now, you will use *fivefold* cross-validation to look at the effect of the regularization penalty on accuracy. You should be able to simply adapt your code from the earlier section with simulated data to work on this real data. Note that due to the larger size of this text data, this section will take several minutes or more to run.

for alpha in [0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]:

for penalty in ['l1','l2']:

The accuracy you will obtain should look like this:

