# CS 6375 - ASSIGNMENT 5

Please read the instructions below before starting the assignment.

- This assignment has two parts. The first part requires you to answer theoretical questions, while the second part asks you to use machine learning libraries for comparing classifier performance.

- The second part of this assignment will allow you to compare the performance of all the classifiers you have studied so far on a common dataset. You are free to use any **open source** package or library, such as R, Python's Scikit-Learn, or TensorFlow.

- In the code folder, please include a README file indicating which dataset you used, how to compile and run your code. Also, mention clearly which packages/libraries you have used.

- You should use a cover sheet, which can be downloaded at: http://www.utdallas.edu/~axn112530/cs6375/CS6375_CoverPage.docx

- You are allowed to work in pairs i.e. a group of two students is allowed. Please write the names of the group members on the cover page.

- You have a total of 4 free late days for the entire semester. You can use at most 2 days for any one assignment. After that, there will be a penalty of 10% for each late day. **The submission for this assignment will be closed 2 days after the due date.**

- Please ask all questions through Piazza, and not through email.

- In many scenarios, you might have to use your best judgement. We want to see that you have put in a sincere effort in all steps of this assignment and have evaluated the classifiers fairly.

# PART I

(6 points)
1. Consider a regression problem of trying to estimate the function f: X -> y where X is a vector of feature attributes and y is a continuous real valued output variable. You would like to use a bagging model where you first create M bootstrap samples and then use them to create M different models – $h_1$, $h_2$, …, $h_M$. You can assume that all models are of the same type.

The error for each of the models would be described as:

$$\epsilon_i(x) = f(x) - h_i(x)$$

where x is the input data and $h_i$ is the model created using $i^{th}$ bootstrap sample

The expected value of the squared error for any of the models will be defined as:

$$E(\varepsilon_i(x)^2) = E[\big(f(x) - h_i(x)\big)^2]$$

The average value of the expected squared error for each of the models acting individually is defined as:

$$E_{avg} = \frac{1}{M}\sum_{i=1}^{M} E(\varepsilon_i(x)^2)$$

Now, you decide to aggregate the models using a committee approach as follows:

$$h_{agg}(x) = \frac{1}{M}\sum_{i=1}^{M} h_i\,(x)$$

The error using the aggregated model is defined as:

$$E_{agg}(x) = E[\left\{\frac{1}{M}\sum_{i=1}^{M} h_i\,(x) - f(x)\right\}^2]$$

which can be simplified as:

$$E_{agg}(x) = E[\left\{\frac{1}{M}\sum_{i=1}^{M} \epsilon_i\,(x)\right\}^2]$$

where we used the value of $\varepsilon_i$ is defined above.

Prove that

$$E_{agg} = \frac{1}{M} E_{avg}$$

provided you make the following assumptions:

1. Each of the errors have a 0 mean

$$E(\epsilon_i(x)) = 0 \text{ for all i}$$

2. Errors are uncorrelated

$$E\left(\epsilon_i(x)\epsilon_j(x)\right) = 0 \text{ for all i} \neq \text{j}$$

(4 points)

2. Jensen's inequality states that for any *convex* function *f*:

$$f\left(\sum_{i=1}^{M} \lambda_i x_i\right) \leq \sum_{i=1}^{M} \lambda_i f(x_i)$$

In question 1, we had assumed that each of the errors are uncorrelated i.e.

$$E\left(\epsilon_i(x)\epsilon_j(x)\right) = 0 \text{ for all i} \neq \text{j}$$

This is not really true, as the models are created using bootstrap samples and have correlation with each other. Now, let's remove that assumption. Show that using Jensen's inequality, it is still possible to prove that:

$$E_{agg} \leq E_{avg}$$

# PART II

(40 points)
In this part, you will compare performance of classifiers on a chosen dataset. Below is the list of classifiers you have learned in class:

- Decision Trees
- Perceptron (usually specified as a neural net with 0 hidden layers)
- Neural Net
- Deep Learning
- SVM
- naïve Bayes
- Logistic Regression
- k-Nearest Neighbors
- Bagging
- Random Forests
- AdaBoost
- Gradient Boosting

The idea is to choose a single dataset and run all of these classifiers on that dataset using the best set of parameters for each case, and then compare their performance. This is more of a data science assignment, rather than a programming one. You will be allowed to use any **open source** machine learning library, such as R, Python's Scikit-learn, or TensorFlow.

If you decide to do this assignment in R, a good place to search for packages is the RSeek website at http://www.rseek.org. Please make an effort to learn packages or any other software tool that you use.

Below are the steps of the project. Please be sure to do all of them and include details in the report.

**I. Choose dataset:**

UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets.html) is one of the most popular sources of datasets. You can filter them for classification tasks and also choose other characteristics such as attribute type and application areas e.g. Life Sciences, CS/Engineering, etc.

For this assignment, you need to choose a dataset that interests you and that satisfies the following criteria:

1. The task should be "Classification", since you would be implementing classifiers.
2. The data type should be "Multivariate"
3. The number of instances should be more than 100.

It would be advisable to check how many missing or N/A values your dataset contains. Please note that image and text data can be tricky to analyze and requires special pre-processing. If you choose such a dataset, you are responsible for performing pre-processing before using classifiers.

## II. Pre-process the dataset:

Pre-processing involves checking the attributes and predicted values for null values, scaling the attributes, converting factor variables to numeric values (e.g. Yes/No to 1/0), checking for redundant attributes (those that have no correlation with the output or those that are all same), etc. You have done this in a previous assignment, but this time you are free to use open source library. Be sure to mention your techniques in the report.

## III. Finding best classifier parameters

Each classifier requires a number of parameters. For example, a bagging model requires you to specify number of models (generally trees) to use, and the parameters of individual trees, such as maximum depth. Similarly, a Random Forest model requires you to specify the number of variables randomly sampled as candidates at each split. In order to get the best performance from each classifier, **you have to explore multiple parameter choices and choose the best value**. Also, you should try n-fold cross validation for each parameter choice to get a good performance estimate. This is a critical part of data science practice.

In order for us to see that you really did try a number of different parameters and also to help you keep track of performance, **you have to maintain a log of your experiments.**

A tabular form as shown below would be great:

| **Experiment #** | **Classifier** | **Cross-Validation fold** | **Parameter1** | **Parameter2** | **…** | **Average Accuracy of 10 folds** |
|---|---|---|---|---|---|---|
| 1 | LogisticRegression | 10 | parameter 1 = x1 | parameter 2 = x2 | … | |
| 2 | k-NN | 10 | parameter 1 = x1 | parameter 1 = x1 | | |
| … | … | | .. | .. | . | . |

It is up to you how many times you run the experiments or how you many permutations of the parameters you try. However, you need to convince us that you made a sincere effort for finding the best set of parameters.

## IV. Testing all classifiers together

After finding best parameters, you will run all the classifiers together and this will give you relative performance of each algorithm on exactly the same dataset. You should use n-fold cross validation

here also. During testing, you will evaluate models on at least one more parameter besides accuracy. It could be precision, recall, ROC or area under ROC curve. An excellent source for model evaluation is the pROC package in R. Other languages also have evaluation methodologies. Below is the outline of testing methodology:

```
numFolds = k (where k >= 10)
classifiers = {c1, c2, …, cn} // list of n classifiers with best parameters
split the data into k folds d[1…k]
for i in 1 to numFolds
        // create training dataset by combining all folds except d[i]
        train = {d[1] + d[2] + … + d[i-1] + d[i+1] + … + d[k]}
        // create test dataset using d[i]
        test = d[i]
        for c in classifiers classifiers
                // create a model of type c using train
                model <- createModel(c, train)
                // find accuracy of model of type c on test
                for classifier c: accuracy[i] <- findAccuracy(model, test)
                for classifier c: other_parameter[i] <- findEvaluation(model, test)
        next c
next i
```

At the end of the code, you will output the average of the accuracy and other evaluation parameter for each classifier i.e.
average accuracy = average (accuracy[1], accuracy[2], …, accuracy[n])
average other_parameter = average (other_parameter[1], other_parameter[2], …, other_parameter [n])

The above experimental strategy will ensure fair evaluation of all classifiers.

## V. Reporting your results:

You should report your results as follows:

Number of instances in dataset:
Number of attributes in dataset:
How many fold cross-validation performed:

| Classifier | Best Parameters Used | Accuracy | Another evaluation metric |
|---|---|---|---|
| Decision Tree | | | |
| Neural Net | | | |
| … | | | |

## VI. Analysis:

In a few paragraphs, explain your experiments and results. Analyze which methods performed best and why do you think they performed best. Analyze which were weak methods and why. Do some attributes influence the output more than others? Is accuracy a good evaluation metric or did you find another metric that is better than accuracy. Any other details or plots that you wish to include.

## What to submit:

- Project report including details of dataset selected (including URL), pre-processing, pseudocode, evaluation metric used, results table, and analysis.
- Log file showing how you found the best set of parameters for each model.
- Code file.
- README file indicating which languages and packages you used and how to compile your code.