# Project Milestone 3

Zhe He (zheh)

Songzhe Cheng (songzhec)

- Did you receive any help whatsoever from anyone in solving this assignment? **No**
- Did you give any help whatsoever to anyone in solving this assignment? **No**

## Tuning ADTree classifier

ADTree stands for alternate decision tree, which is a boosting decision tree algorithm for classification. It improves the boosting algorithm with alternating structure so correlations between attributes of objects can be inferred in the tree and thus results in a better result. It is first formulated in a 1999 paper by Freund, Y., Mason, L, "The alternating decision tree learning algorithm" in Proceeding of the Sixteenth International Conference on Machine Learning, Bled, Slovenia.

The ADTree classifier has two parameters: number of boosting iterations and expand nodes. Number of boosting iterations are the number of boosting iterations where boosting decision stumps are generated. Expand nodes decides how the nodes are expanded in the tree: all nodes, the heaviest nodes, the z-pure nodes or a random walk.

It is stated in the paper of ADTree that the selection of number of boosting iterations are related to the data model and should be selected manually for a balance between complexity and testing accuracy. Also, since ADTree supports only 2-class model, we have to use MCC to classify multi-class datasets. After experimenting with the given sets, we found that the ADTree is extremely sensitive to the data and each datasets do have different best number of boosting iterations. Also, for 2-class datasets, best number ranges from 2 to 30. For multiple datasets, best number ranges from 10 to 200. Based on the result, we used CVParameter for 2-class and manually parameter experiment for multi-class datasets.

As for the expand nodes parameter, it varies from dataset to dataset. We think it is related to how the attributes correlates with each other. So we choose the best for each dataset.

**Best result is:**

```
anneal          -B 132 -E -3
audiology       -B 12 -E -1
autos           -B 42 -E 0
balance-scale   -B 46 -E 0
breast-cancer   -B 10 -E -3
colic           -B 6 -E -3
credit-a        -B 8 -E 0
diabetes        -B 26 -E 0
glass           -B 34 -E -1
heart-c         -B 14 -E -2
hepatitis       -B 30 -E -1
```

```
hypothyroid      -B 10 -E -2
```

And a final average error rate is 0.661.

## Tuning RBFNetwork classifier

The RBFNetwork classifier class implements a normalized Gaussian radial basis function network. A radial basis function network is an artificial neural network that uses radial basis functions as activation functions. It is first formulated in a 1988 paper "Multivariable functional interpolation and adaptive networks" by Broomhead and Lowe.

The RBFNetwork classifier has five possible parameters to adjust: number of clusters to generate, random seed to be used by k-means, ridge value for logistic or linear regression, maximum number of iterations for the logistic regression and minimum standard deviation for the clusters. The random seed does not affect classifier performance, so it is left as untouched. Besides, the default setting for maxim number of iterations for logistic regression is to iterate until convergence. It is already the best setting of this option since iterating until convergence should provide better result than prematurely stopping the iteration before convergence. As a result, we only tuned remaining three parameters in the experiments: number of clusters (B), ridge value (R) and minimum standard deviation for clusters (W). The whole range we explored is as follows:

number of clusters: *1, 2, 3, 4, 5*

ridge value: *2e-8, 4e-8, 6e-8, 8e-8, 1e-7*

standard deviation: *0.01, 0.02, 0.03, 0.04, …, 0.2*

All combinations of those values are explored and the best parameter settings are chosen as the model for that dataset. RBFNetwork seems to be not sensitive to standard deviation for clusters. When varying standard deviation values with other two value fixed, the error rate only varies several times, so the max and average error rate is merely affected by this parameter. RBFNetwork also tends to take much longer time when training certain dataset. The training time is also greatly affected by number of clusters parameter. Normally it takes about 20 seconds on my computer to train a dataset, though certain dataset consumes several minutes in training phase.

For each dataset, the classifier is dynamically constructed using possible optimal parameter settings by exploring parameter ranges shown above. We do not use CVParameterSelection class to help choose best parameters because it consumes excessive time on certain dataset (more than 20 minutes).

## Tuning END classifier

END classifier stands for Ensembles of Balanced Nested Dichotomies for Multi-class Problems. Similar to MCC, END is a meta classifier for handling multi-class datasets with 2-class classifiers. Lin Dong, Eibe Frank and Stefan Kramer first published the idea in "Ensembles of nested dichotomies for multi-class problems" in Twenty-first International Conference on Machine Learning, 2004.

There are in total 4 parameters to set: Random number seed, number of iterations, debug mode and full name of base classifier. Since Random number seed and debug mode don’t have practical influence on our result, we would just leave that default. As for base classifier, we chose the default settings also, since experimenting with all possible base classifier takes too much time and we already have great result with ADTree.

From the result of experimenting with number of iterations, it depends on the datasets as well. For some datasets, the result is at best when taking 5 as the value, while for others it might be 30. So we used CVParameter to choose it from range 5 to 50. And the resulting average error rate is quite high, around 0.8.

## Conclusion

Among the classifiers we used, the best robustness was obtained by alternating decision tree, which was naturally quite robust without parameter tuning. However, by using internal CVParameter to set the two critical values of ADTree for 2-class datasets and using nested loop of manually selection of parameters, we were able to improve its average robustness from around 0.8 to 0.66, and improve it's maximum robustness from around 1.2 to 1.167.”