# Machine Learning: Computer Exercise 3

机 53　陈声健　2015010509

## 1. Task Description

With the old data provided at exe1, the first task is to find a package of k-nearest neighbor classifier and study the usage of this package. The second task is to study a package of random forest, especially the method it uses for assessing the relative contribution or importance of each feature in the final decision.

## 2. Experiment Design

I use sklearn, a python package to study the k-nearest neighbor algorithms. In sklearn, user has several option to adjust the performance of the model. I study the effect of k by fitting the model with different k and test the model on training data as well as test data and plot the accuracy. I also compare the running time of two tree search algorithms with the brute force method to see their efficiency on our dataset. I also study two ways to set the parameter "weights" (uniform, distance).

Sklearn also provides a function for random forest algorithms. I study the effect of the following parameters by setting them to different values and compare the result plots: *n_estimators*, *criterion*, *max_depth*, *min_impurity_decrease*, *bootstrap*. I also study how this package assess the relative importance of each feature.

## 3. Method

To study the effect of parameter k, I set k from 1 to 100 and fit the model to the training data, and then test the model on both training set and test set. To study the effect of different choice for *algorithms*, I do the same experiment as mentions above and record the running time of the program to see whether "KD-tree" or "Ball Tree" will improve the efficiency. I also tried different *weights* with the same experiment mentioned above and compare the curve.

To study the effect of n_*estimators*, similar to k mentioned above, I set n_estimators from 1 to 150, fit the model to the data with some setting of other parameters and calculate the prediction accuracy of the model on training and test data. *Criterion* controls the way to measure the quality of a split, usually in impurity, options are 'gini' and 'entropy'. *max_depth* is the maximum depth of the tree, I try the value of 2, 5, 8, and None. If it is None, nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. Setting *min_impurity_decrease* to a certain value, a node will be split if this split induces a decrease of the impurity greater than or equal to this value. By default, it is 0. I also set it to 0.1, 0.05 and 0.01 to study its effect. Bootstrap determines whether bootstrap samples are used when building trees. After the model finishes fitting the training data, it will generate an attribute called feature_importances_ that contains the relative importance of each feature. The feature with high value has high importance. I plot the relative feature importance with bar plot of a simple test case.

## 4. Result and Observation

As we can see in left (weights = uniform) of Fig.1 in the supplementary materials, when k=1, accuracy on training set is 1 and that on test set is also relatively high. As k increases, the

accuracy on training set keeps decreasing but that on test set drops down a little but rises up at around k = 20, and then keeps decreasing. Comparing the two plots in Fig.1, when we set weights = distance, the model always classifies the training data correctly, and also do better than the left plot. The reason can be quiet straightforward, considering data that closer to the sample more will benefit the classification. Table 1 show the of different algorithms. It is stated that KD-Tree can improve the efficiency when the dataset is large. And Balltree does better in high dimension situation. Of course, there is no free launch, they require more preprocess of the dataset. According to the result, we can not see much improvement. Perhaps our dataset is relatively simple, the improvement can not compensate the extra calculation. I also found that different setting of algorithms will result in the same curve, the only difference is the running time.

The result of task 2 is shown in the Fug.2 in supplementary materials. From fig(a) and fig(b), *entropy* performs slightly better than *gini* on the test set. However, according to the literature, entropy metric may take more time because of the logarithmic computation. Their performances are quiet similar on training set. From fig c~f, increasing max_depth will improve the performance of the model when not most of the features are used. Because considering more features will very probably help. However, when the *max_depth* comes close to the feature number (10 in our case), the model will not improve much if we keep increasing *max_depth*. Fig g~i show the result of *bootstrap* = false under different *max_depth.* Not using bootstrap means that we fit each decision tree in the forest to the same dataset and it easily suffers from overing fitting. When I test *bootstrap* = false under a high *max_depth*, the accuracy on training set is always 1, but the result on test set is worse than using bootstrap. Fig j~l and fig.e show the result when setting *min_impurity_decrease* to 0.1, 0.05, 0.01. Higher *min_impurity_decreas* means high impurity of each decision tree, thus lower accuracy of the model.

In sklearn package, the relative feature importance is determined with “gini importance” or “mean decrease impurity”. For each node related to a particular feature, the actual decrease in node impurity is summed and averaged across all trees. And then the values are normalized to keep the sum equal to 1 [3][4]. Result of one test case is show in Fig.3.

## 5. Conclusion

In k-nearest neighbor, selecting a k that not too small or not too large (around 20 in our data) will have the best performance. Using *metric* of “distance” is better than “uniform” on both training and test data. It is stated that using more advanced *algorithms* will save time in large scale and high dimension dataset. But we can not see this advantage in our relatively simple dataset.

In random forest, setting *criterion* to “entropy” may result in better accuracy, but will also cost more time. In most time, “gini” is enough. Increasing *max_depth* is beneficial if it is small. Keeping min_impurity_decrease smalle or even 0 is good for high accuracy on both training and test set. Using *bootstrap* helps to reduce overfitting.

## 6. Reference

[1] Sklearn.neighbors.KNeighborsRegressor. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

[2] Sklearn.ensemble.RandomForestClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.feature_importances_

[3] Breiman, Friedman, "Classification and regression trees", 1984.

[4] How are feature_importances in RandomForestClassifier determined? Stackoverflow. https://stackoverflow.com/questions/15810339/how-are-feature-importances-in-randomforestclassifier-determined