AU 332 Artificial Intelligence: Principles and Techniques

By: Meng Zhou, Yangqian Wu

Instructor: Yue Gao

December 15, 2019

I. INTRODUCTION

A. Background

In this homework, We're going to predict the forest cover type using several machine learning techniques. The library *sklearn* makes calling machine learning method really simple. We use those machine learning algorithms that We're familiar with. The method we have tried include Support Vector Machine, KNN, Random Forest and Extra Trees Classifier, which is just a variant of Random Forest but shows a better performance.

B. How to run our code

In our code, to make it easy to tune different algorithms in command line, we use a library called argparse. For example you can run like this:

python algorithms.py –algorithm=RandomForest

The optional algorithms are RandomForest, KNN, SVM and Extratrees.

II. TUNING ALGORITHMS

A. SVM

For Support Vector Machine, it seems that there are not so many parameters to tune. We mainly tune the kernel we use and the regularization term C, which penalize those support vectors that are near the decision boundary. By the way, the fitting process of SVM is obviously slower than other three algorithms, we believe it's mainly because the optimization process of the dual problem using SMO algorithm is relatively slow. The tuning result is shown below:

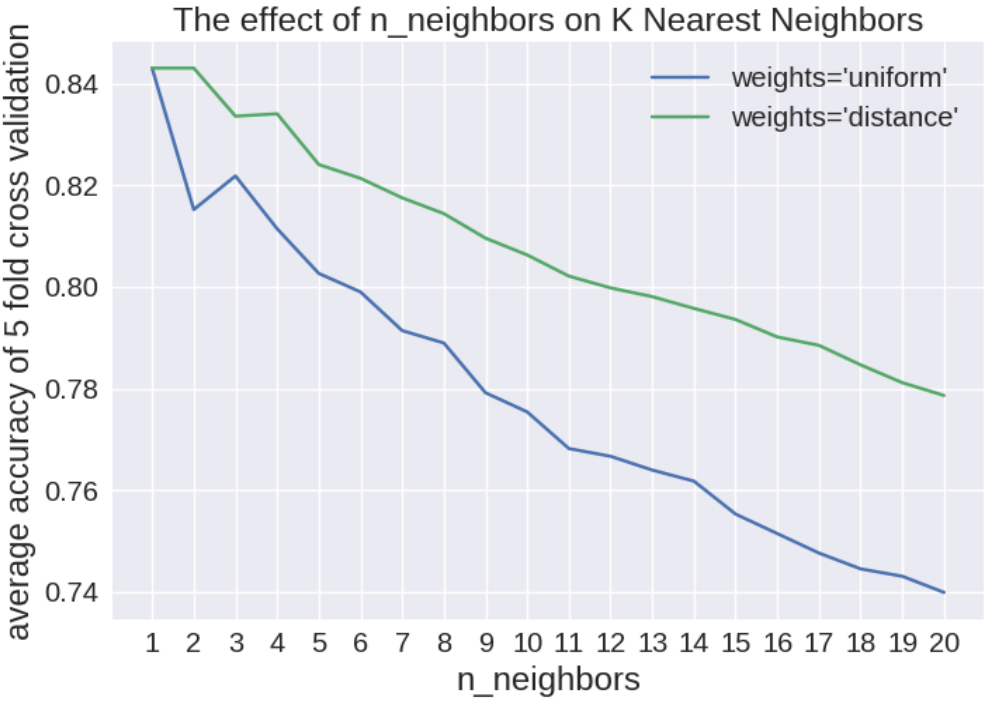
kernel \ C	1.0	5.0	1.0	5.0	100.0
'linear'	72.07%	72.26%	72.18%	72.12%	72.15%
'poly'	69.77%	75.63%	77.75%	80.47%	81.22%
'rbf'	72.84%	77.56%	79.06%	81.18%	81.97%
'sigmoid'	57.16%	54.12%	53.82%	53.82%	54.01%

From the result, we can find that using sigmoid kernel gives us a bad result, this is probably because the corresponding map does not help to map the data to a space where they are more close to linear separable. Generally, if we increase the coefficient of the regularization term, we get a better result. From our opinions, the reason for this is we are forcing the distance between the support vectors and the hyperplane to be smaller. Therefore we have a better classification performance by doing so.

B. KNN

For KNN algorithm, we find that tune the parameters 'n_neighbors', which indicates the number of neighbors to vote for the class of a certain point might be very useful. Besides, we also try to tune 'weights' into 'uniform' or 'distance' and judge which one could be better in the process of voting.

The final comparison diagram is as below:



(a) The effect of n_neighbors and weights on K Nearest Neighbors

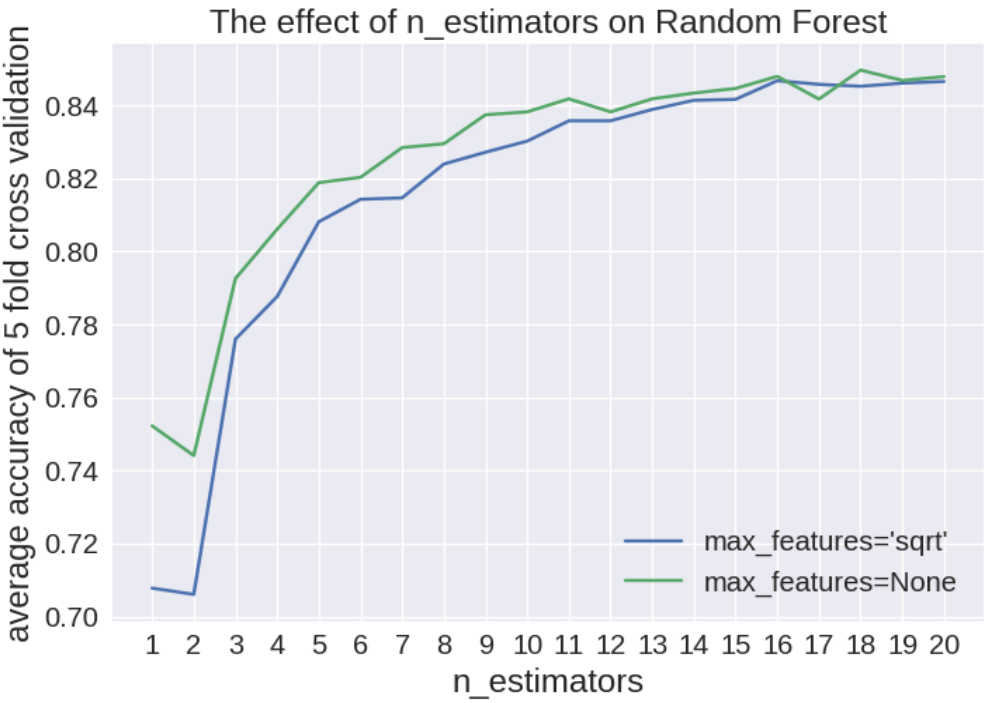
From the diagram shown above, we easily find out that whether 'weights' is 'uniform' or not, with the increasing of 'n_neighbors', the accuracy of cross validation is on the trend of decreasing. It's probably because that if 'n_neighbors' becomes larger, more and more neighbors will have an effect in deciding the its class, the further neighbor might have a false weight on it. By increasing the 'n_neighbors', the model become more simple which might ignore the useful information in training set. In our attempt, the best choice for 'n_neighbors' is 1 although the default value for it is always 5. In general experiment, the usual process is trying several k and deciding a k with best performance.

Besides, we also tune the 'weights' from default 'uniform' to 'distance' and find that 'distance' outperform 'uniform' whatever 'n_neighbors' is. It's because when we use 'distance', we put more confidence in the near neighbors which could be helpful to classify it correctly.

C. RandomForest

For RandomForest algorithm, we find that tune the parameters 'n_estimators', which indicates the number of trees in the forest might be very useful. In the intuition, more trees in the random forest will result in higher accuracy. Besides, we also try to tune 'max_features' which indicates the number of features to consider when looking for the best split. In the process, we have tried 'sqrt'(max_features=sqrt(n_features)) and 'None'(max_features=n_features).

The final comparison diagram is as below:



(b) The effect of n_estimators and max_features on Random Forest

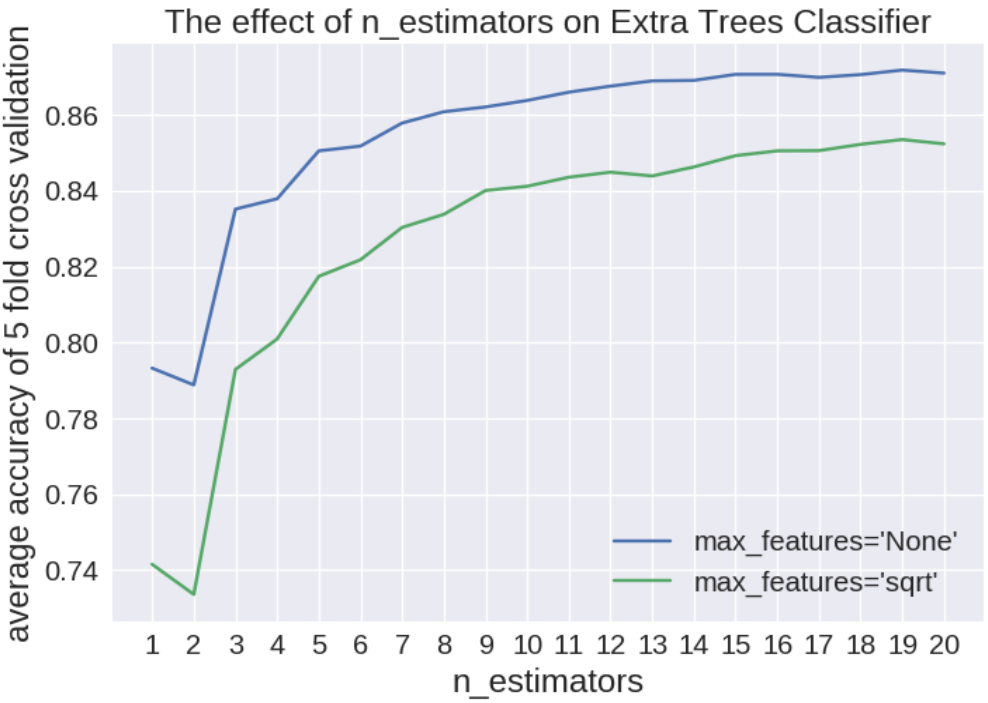
From the diagram shown above, we easily find out that whether we choose 'sqrt' or 'None' as 'max_features', with the increasing of 'n_estimators', the accuracy of cross validation is on the trend of increasing. It's corresponding to our intuition, because more estimators will result in more accurate approximation to real class. We have tried 1 to 20 trees and found that highest accuracy occurs in around 20 and its accuracy achieves about 84%. Generally speaking, higher number of trees will have better performance but will cost more time. Therefore, we should choose a relatively high number, which makes your predictions stronger and more stable.

Besides, we also tune the 'max_features' to 'sqrt' and 'None' and find that 'None' outperform 'sqrt' when 'n_estimators' is small. But when 'n_estimators' increases, the difference becomes small. It's probably because when we use 'sqrt', we limit maximum number of features in an individual tree, which will furthermore limit the options we can consider. In general, increasing max_features generally improves the performance of the model.

D. ExtraTrees

For ExtraTrees algorithm, just as what we tuned in the 'RandomForest' part, we have tried to tune the parameters 'n_estimators', which indicates the number of trees in the forest and 'max_features' which indicates the number of features to consider when looking for the best split. The difference between RandomForest and ExtraTrees is that ExtraTrees have more randomness and it will increase the accuracy of whole forest. Moreover, in the intuition, more trees in the forest will result in higher accuracy. And in the process, we have tried 'sqrt'(max_features=sqrt(n_features)) and 'None'(max_features=n_features) for choosing the 'max_features'.

The final comparison diagram is as below:



(c) The effect of n_estimators and max_features on Extra Trees

From the diagram shown above, we also find out the similar comparison that whether we choose 'sqrt' or 'None' as 'max_features', with the increasing of 'n_estimators', the accuracy of cross validation is on the trend of increasing. We have tried 1 to 20 trees and found that highest accuracy occurs in around 20 and its accuracy achieves about 87%. Besides, for option 'max_features', 'None' outperform 'sqrt' whatever 'n_estimator' is. The difference between them is quite obvious than in Random Forest. The reason for the difference around 'n_estimators' and 'max_features' above is quite similar with the analysis in Random Forest.

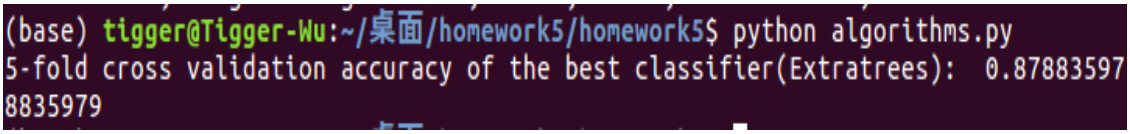
Besides, the reason for Extra Trees's better performance compared to Random Forest(84% to 87%) is probably as below:

In Extra Trees, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This will increase the randomness of forest(ExtraTrees) and furthermore increase the accuracy.

Therefore, we finally choose to use ExtraTree algorithm with 'n_estimators' equals to 100 , 'max_features' equals to None and get the accuracy of 87.8%.

E. The Algorithm We Choose

Comparing all the algorithms and hyber parameters we have tried, we finally use ExtraTree with 'n_estimators' = 100 and 'max_features' = None to implement. In the labeled data, we have got the accuracy of more than 87.8%, the the labels of test data could be seen in 'predict.csv'.



(d) The final accuracy of Extra Trees with 100 estimators

III. COMPARISON BETWEEN EACH ALGORITHMS

1. SVM

For SVM part, we have tried four kernels and tuned hyber parameter C. We found that SVM doesn't perform well in this classifiction task. That's probably our training set is too big to seperate it correctly to some extent. It's hard for find the suitable kernel to seperate it properly.

Besides, as a eager laerning approach, SVM costs a very very long time to train, that's also why we don't want to use it. But SVM will be suitable for the small data set.

2. KNN

For KNN part, we have tried to tune the n_neighbors and weights, and found that KNN perform a little better than SVM. When we choose n_neighbors as 1, the accuracy could reach about 84%. KNN does well in this traing set , that's probably because in this data set, the feature that more close two sample are more likely they are the same class is pretty protruding. As a lazy learning approach, KNN cost less time than other eager learning approach like SVM.

3. RandomForest

For RandomForest part, we have tried to tune number of trees 'n_estimators' and the number of features to consider when looking for the best split 'max_features'. We found that more trees in the forest and more features could be split will result in better performance.

In this task, Random Forest outperform SVM when the number of trees is bigger than 5. Random Forest is especially suitable for the discrete and nonlimited-value features and also perform well in big data set. Tree structure is still a good model for classification.

4. ExtraTrees

For ExtraTrees part, we also have tried to tune number of trees 'n_estimators' and the number of features to consider when looking for the best split 'max_features' and found the same trend as Random Forest. In our task, Extra Trees outperforms Random Forest because of its higher randomness. It's a really good model for discrete and nonlimited-value features.