
Problem assignment 10

Due: Thursday, April 14, 2022

In this assignment we will have an opportunity to experiment with different approaches that let you combine multiple classification models into an ensemble with the hope that the ensemble will improve the classification performance. Sklearn python library implements multiple ensemble methods we covered in the class: bagging, boosting (Adaboost), random forests, and gradient boosting. The ensembles learn multiple classification models and outputs the results via voting schemes (classifiers) and averaging (regressors). The sklearn functions implementing the different ensemble classification methods we will study in this assignment are:

- BaggingClassifier
- AdaboostClassifier
- GradientBoostingClassifier
- RandomForestClassifier

Problem 1. Single decision tree baselines

We will use the above ensemble methods in combination with dcision trees. You were given a file *hw10_DT.py* that loads the training and testing data, and learns a decision tree classifier with the limit of five (5) placed on the minimum number instances (samples) in the leaf nodes. This decision tree will be one of our baselines for the single decision tree solution. Overall we will use 3 single decision tree baselines in this assignment:

- (Option 1) Decision Tree classifier with the limit of five (5) placed on the minimum number instances (samples) in the leaf nodes
- (Option 2) Full Decision Tree classifier that is defined with the tree_depth = None
- (Option 3) One level decision tree that is defined with the tree_depth = 1, and consists of just one split.

Please run *hw*10_*DT.py* and collect and record the stats for the baseline 1. Then modify the file so that it implements baseline Options 2 and 3. Please record the stats in the report. Analyze the differences in the baseline performance. Which single tree method appears to be the best? Are there any signs of overfitting or underfitting in any of the three options. Please explain.

Problem 2. Bagging

Part a. Write and submit a file *hw*10_*Bagging_DT.py* that implements the Bagging approach combining 5 classifiers trained using Option 1 decision tree settings in Problem 1. The bagging procedure should use default parameter options for instance subsampling, and feature subsampling. Compare the results for the single decision tree baseline option 1 and the ensemble of 5 of such trees that are built using Bagging. Record the new stats and compare the results. Did bagging help? Rerun the Bagging procedure but now with 10, 30 trees. Analyze the results and report any interesting findings? Did more trees help?

Part b. Take the file *hw*10_*Bagging_DT.py* from Problem 2. Part a. and modify it such that the trees in the ensemble use Option 2 settings. Run the new version of Bagging on 5, 10 and 30 trees. Compare the results to the singe decision tree option 2 results in Problem 1. What do you see? Are ensembles improving the performance of the sngle Tree Option 2 baseline? Explain why? Analyze the results for the method with 5, 10, 30 trees and draw any conclusion you can make.

Part c. Repeat Part b of Problem 2 but this time use Option 3 for the single decision tree to implement the ensemble. Analyze the results. Do the results make sense? Is performance improving with the number of trees?

Part d. Summarize your analysis and findings about bagging and three single decision tree options in a table/s, and make any conclusions about Bagging benefits.

Problem 3. Boosting: Adaboost

In this problem we experiment with Boosting, in particular Adaboost method.

Part a. Write and submit a file *hw*10_*Adaboost_DT.py* that implements the Adaboost ensemble approach combining 5 classifiers trained using Option 1 settings in Problem 1. Compare the results for the single Decision tree option 1 and the ensemble of 5 of such trees that are built using Adaboost. Record the new stats and compare the results. Did Adaboost help? Rerun the Adaboost procedure with 10, 30 trees. Analyze the results and report your findings

Part b. Modify file *hw*10_*Adaboost_DT.py* you wrote for Problem 3 Part a. so that the

trees in the ensemble use Option 2 tree settings. Run the new version of on 5, 10 and 30 trees. Compare the results to the Option 2 solution in Problem 1. What do you see? Are ensembles improving the performance of the Single Tree Option 2 baseline? Explain why? Analyze the results for the method with 5, 10, 30 trees.

Part c. Repeat Part b of Problem 3 but this time use Option 3 for the decision trees to implement the ensemble. Analyze the results. Do the results make sense?

Part d. Summarize your analysis and findings about Adaboost and three Decision Tree options in a table/s, and make any conclusions about Adaboost benefits. Also please compare the new results to Bagging results in Part d. of Problem 2.

Problem 4. Boosting: GradientBoosting

Next we experiment with the Gradient Boosting method.

Part a. Write and submit a file *hw*10_*Gradientboosting_DT.py* that trains the Gradient boosting ensemble approach combining 5 classifiers trained using Option 1 settings in Problem 1. Compare the results for the Single Decision tree option 1 and the ensemble of 5 of such trees that are built using the gradient approach. Record the new stats and compare the results. Did Gradient boosting help? Rerun the Gradientboosting procedure with 10, 30 trees. Comment on the results. Compare the results to Bagging and Adaboost.

Part b. Modify file *hw*10_*Gradientboosting_DT.py* you wrote for Part a. so that the trees in the ensemble use Option 2 tree settings. Run the new version of file on 5, 10 and 30 trees. Compare the results to the Option 2 solution in Problem 1. What do you see? Are ensembles improving the performance of the Single Tree Option 2 baseline? Explain why? Analyze the results for the method with 5, 10, 30 trees and report the findings.

Part c. Repeat Part b of problem 3 but this time use Option 3 for the decision trees to implement the ensemble. Analyze the results as in Part a and b. Do the results make sense?

Part d. Summarize your analysis and findings about GradientBoosting and three Decision Tree options in a table/s, and make any conclusions about Gradient Boosting benefits. Compare the results to Bagging (Problem 2) and Adaboost (Problem 3) and analyze the differences.

Problem 5. Random Forests

Finally we will experiment with the Random Forest method.

Part a. Write and submit a file *hw*10_*RandomForest_DT.py* that trains the Random Forest ensemble combining 5 classifiers trained using Option 1 settings in Problem 1. The Random

forest method should use the default options for the number of instances, number of features, and 'auto' feature subset selection for the splits. Compare the results for the Single Decision tree option 1 and the ensemble of 5 of such trees that are built using Random Forest. Record the new stats and compare the results. Did it help? Rerun the Random Forest procedure with 10, 30 trees. Comment on the results.

Part b. Modify file *hw10_RandomForest_DT.py* you wrote for Part a. so that the trees in the ensemble use Option 2 tree settings. Run the new version of file on 5, 10 and 30 trees. Compare the results to the Option 2 solution in Problem 1. What do you see? Are ensembles improving the performance of the Single Tree Option 2 baseline? Explain why? Analyze the results for the method with 5, 10, 30 trees.

Part c. Repeat Part b of problem 5 but this time use Option 3 for the decision trees to implement the ensemble. Analyze the results. Compare to the problems 2, 3, 4 solutions. Do the results make sense?

Part d. Summarize your analysis and findings about Random Forest and three decision Tree options in a table/s, and make any conclusions about Gradient Boosting benefits. Also please compare the results to Bagging, Adaboost, Gradient Boosting results in Problems 2, 3, and 4, make conclusions about their performance and differences.