10601 Machine Learning Project Milestone 3: Tuning Classifiers

Ashutosh Pandey
ashutosp

Jingfei Zhang
jingfeiz

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.

1. Tuning the "Random Forest" classifier learner

.1 About Classifier:
Random forests are an ensemble learning method for classification (and regression) that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. The method combines "bagging" idea and the random selection of features in order to construct a collection of decision trees with controlled variation.
Out of training set, random numbers of examples are chosen and for these random training examples, random features are selected to create decision tree. This process is repeated to create several decision trees which collective form a forest.

.2Parameters varied:
Here are parameters that we varied to tune the classifier:

- Number of trees: This parameter decides the number of trees to be build in the forest. Increasing the number of

trees may lead to increment in learning time. However, if the number of trees is too less then there is a chance of overfitting.

- Number of features: This parameter decides the number of random features to consider while generating a decision tree. Reducing number of features reduces both the correlation between the tress and their individual strength. Hence it is important to select optimal number of features because considering more than optimal number of features can increases the learning time and correlation among trees. On the other side, if number of features is too less then the strength of an individual tree decreases, which increases the error rate of forest.
- Depth of trees: This parameter decides the maximum depth of the trees. More depth may lead to overfitting however less depth may lead to underfitting.

The training time on the 12 given datasets is about 15min.

.3Final Settings:

In our approach, we used auto-tuning and brute force (nested loops) method to find the optimal values for the parameters described in the previous section. However, we tried multiple iterations to figure out best range and candidate values for a parameter to run the loop. This implies that each parameter we analyzed how it impact the prediction and what should be best range to run a “for loop” for that parameter. In addition, we used different set of candidate parameters for Number of features based on the feature number of the dataset.
The configuration is as follows:
Number of Trees:
1,2,4,6,7,9,10,15,30,50,60,70,100,120,150,200
Number of Features:

Feature Number of Data	Parameter Set
num <= 10	{0:featureNum}
10 < num <= 15	{0,1,2,3,5,7,10}
15 < num <= 25	{0,2,3,5,7,15}
25 < num <= 40	{0,2,4,7,12,25}
40 < num <= 60	{0,3,7,15,25,40}
60 < num	{0,5,8,15,25,40,60}

Depth of Trees: 0,featureNum/2,featureNum*2/3,featureNum
Using this approach, there was improvement in the mean value from 0.748 to 0.724, in the max value from 1.210 to 1.081.

2.Tuning the "Jrip" classifier learner

.1 About Classifier:
Jrip implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which is an optimized version of IREP. Jrip is based in association rules with reduced error pruning (REP), a very common and effective technique found in decision tree algorithms. In REP for rules algorithms, the training data is split into a growing set and a pruning set. First, an initial rule set is formed that over the growing set, using some heuristic method. This overlarge rule set is then repeatedly simplified by applying one of a set of pruning operators typical pruning operators would be to delete any single condition or any single rule. At each

stage of simplification, the pruning operator chosen is the one that yields the greatest reduction of error on the pruning set. Simplification ends when applying any pruning operator would increase error on the pruning set.

.2Parameters varied:

Here are the parameters we varied to tune the classifier:

- Number of Folds: This parameter sets the number of folds for REP. If number of folds is important to avoid the problem of overfitting and underfitting.
- Minimum Weights: This parameter sets of minimal weights of instance within the split. This will directly change the setting of rules and may also impact the learning time of the classifier.
- Number of Runs: This parameter set the number of runs for optimization. If number of runs is high then learning time will be increased. If number of runs if low then learning will not be optimal.
- Pruning: Enable of disable pruning. Pruning helps in reducing overfitting.

The training time on the 12 given datasets is about 7min.

.3Final Settings:

In our approach, we used auto-tuning and brute force (nested loops) method to find the optimal values for the parameters described in the previous section. However, we tried multiple iterations to figure out best range for a parameter to run the loop. This implies that each parameter we analyzed how it impact the prediction and what should be best range to run a "for loop" for that parameter.

The configuration is as follows:

Number of Folds: 1,2,3,4,5,10,15,20

Minimum Weights: 0.001,0.01,0.1, 0.2, 0.5, 1, 1.5, 2, 3, 4, 5

Number of Runs: 1,2,3,4,5

Pruning: 1,0

Using this approach, there was improvement in the mean value from 0.882 to 0.820, in the max value from 1.500 to 1.323.

3.Tuning the "Dagging" classifier learner

.1About Classifier:
This meta-classifier creates a number of disjoint, stratified folds out of the data and feeds each chunk of data to a copy of the supplied base classifier. Predictions are made via majority vote, since all the generated base classifiers are put into the Vote meta classifier.
Dagging is useful for base classifiers that are quadratic or worse in time behavior, regarding number of instances in the training data.

.2Parameters varied:
Because Dagging consists of several base classifiers, so this classifier has nested options/parameters to set. However, CVParameterSelection can only tune parameters of first level classifier, and GridSearch can only tune two parameters at the same time. So we tune multiple parameters with nested loops without any helper functions.

Here are the parameters we varied to tune the classifier:

- Number of Folds: The number of folds for splitting the training set into smaller chunks for the base classifier. When the number is small, the training time is fast, but it cannot make good use of the benefit of majority vote of Dagging. On the other hand, when the number of folds is large, the characteristic of majority vote can be utilized, but for each base classifier, the number of samples will be small and may not get good accuracy, and also the training time will be long.
- Base Classifier: In terms of Dagging classifier, there is a base classifier to be used. So the base classifier is a special parameter. The base classifier can be any basic classifier. In the tuning, we choose SMO (default), Logistic Regression, and Random Forest as the candidate

classifier. It turns out that this the choice of base classifier will have big influence on the performance and training time. There is no dominant winner among these classifiers. Each of these base classifiers may work better on some datasets than other classifiers. But in terms of the training time, Logistic Regression is relatively slower.

Parameters specific to classifier weka.classifiers.functions.SMO:

- Complexity: This parameter represents the complexity of the classifier of SVM. Higher complexity tends to introduce lower bias but higher variance. If the complexity is too high, it is more likely to be overfitting. During the process of auto-tuning, we set the candidate values as 0.001, 0.05, 0.5, 1. The results shows that this parameter will have big influence on the test errors.
- Tolerance: The tolerance parameter represents how much errors the SVM learner can tolerate. During the tuning, we tried 0.00001, 0.001, 0.1, 1, 10, and found that the errors of cross-validation are basically only two values: a normal error when tolerance is low, and a big error, say 0.99, when tolerance is high. So we just fixed this parameter as constant of default value (0.001).
- Round-off Error: This parameter represents the epsilon for round-off error. After tuning, we found that this parameter did not influence the performance too much, so we fixed this parameter as constant of default value (1.0E-12).

Parameters specific to classifier weka.classifiers.functions.Logistic:

- Ridge: Set the ridge in the log-likelihood. This parameter represents the regularization term that is how much you want to penalize the large weights. In the auto-tuning, we set this as 1.0E-8, 1.0E-12, 1.0E-6, 1.0E-4, 1.0E-3, 1.0E-2, 1.0E-1, 1. The results show that this parameter will make big difference on the accuracy.

The training time on the 12 given datasets is about 100min.

.3Final Settings:

The final classifier of Dagging is an auto-tuning classifier such that it will tune the parameters automatically based on the input training dataset and is expected to get optimal performance on the test dataset. To train the classifiers, we used nested loops with cross-validation to tune parameters of Number of Folds, Base Classifier, Complexity, and Ridge. The configuration is as follows:

Number of Folds: 1, 2, 5, 10, 20

Base Classifier: SMO, Logistic, Random Forest

Complexity: 0.001, 0.05, 0.5, 1

Ridge: 1.0E-12, 1.0E-8, 1.0E-6, 1.0E-4, 1.0E-3, 1.0E-2, 1.0E-1,1

Using this approach, there was an improvement in the mean value from 1.021 to 0.699, in the max value from 1.545 to 1.000.

4.Summary

Before tuning the parameters, among the classifiers we used, the best robust one was Random Forest, which can obtain good performance without tuning. After tuning, the performance increases further from 0.882 to 0.820 for the mean value, from 1.500 to 1.323 for the max value. However, after auto-tuning Number of Folds, Base Classifier, Complexity, and Ridge parameters by using nested loops, Dagging was able to obtain the best robustness in terms of the mean and max value. The mean value improved from 1.021 to 0.699, and the max value from 1.545 to 1.0.

1 Dataset	Error_C / Error_NB (Before Tuning)		
	Random Forest	JRip	Dagging
anneal	0.35294117647059	0.64705882352941	1.52941176470588
audiology	0.72727272727273	1.27272727272727	1.54545454545455
autos	0.94736842105263	0.89473684210526	0.86842105263158
balance-scale	0.95454545454546	1.16666666666667	0.48484848484849
breast-cancer	1.0	0.86206896551724	0.93103448275862
colic	0.82608695652174	0.82608695652174	1.0
credit-a	0.70238095238095	0.63095238095238	0.66666666666667
diabetes	1.20967741935484	1.03225806451613	1.12903225806452
glass	0.43137254901961	0.58823529411765	0.74509803921569
heart-c	0.77777777777778	1.0	1.0
hepatitis	0.83333333333333	1.5	1.0
hypothyroid	0.21917808219178	0.16438356164384	1.35616438356164
Mean	0.74849457082679	0.88209790235813	1.0213443064923
Max	1.20967741935484	1.5	1.54545454545455

Dataset	Error_C / Error_NB (After Tuning)		
	Random Forest	JRip	Dagging
anneal	0.17647058824	0.2941176470588235 4	0.35294117647
audiology	0.72727272727	0.9545454545454546	0.63636363636
autos	0.76315789474	0.7631578947368421	0.94736842105
balance-scale	1.04545454545	0.9696969696969697	0.40909090909
breast-cancer	0.86206896552	0.9310344827586207	0.93103448276
colic	0.82608695652	0.9130434782608695	0.82608695652
credit-a	0.7380952381	0.9880952380952381	0.72619047619
diabetes	1.08064516129	1.3225806451612903	0.91935483871
glass	0.41176470588	0.4901960784313725 3	0.47058823529
heart-c	1.05555555556	0.9444444444	0.94444444444
hepatitis	0.83333333333	1.0	1.0
hypothyroid	0.16438356164	0.2739726027397726	0.21917808219
Mean	0.72369076946	0.82040707799	0.69855347159
Max	1.08064516129	1.3225806451611	1.0