18/12/2018

Ben-Gurion University of the Negev

Faculty of Engineering Sciences

Department of Software and Information systems

Implementations of Machine Learning Algorithms

Prof. Lior Rokach

Assignment 2

Maya Schvetz 305249708

Yuval Heffetz 302957139

Introduction

In this exercise we implement and evaluate a machine learning model for the classification task with a variation of the Random Forest (RF) ensemble algorithm. In RF, we use several different decision-tree estimators, trained on the same training set, for classification of unseen examples by accumulating the probabilities from all the estimators for each class. The probability of each estimator is computed according to the training examples in the tree-leaf that the new example ended up at. In an attempt to improve this model, in this exercise we added a Naïve-Bayes model in each tree-leaf in every estimator, trained with the training examples in the leaf and used for the prediction of unseen examples that were assigned to the leaf by the RF model.

Implementation

We chose to implement our model in Python, changing the code of the RandomForesClassifier algorithm in the Scikit-Learn Python library, obtained from here. A short explanation of our implementation in the file **RandomForestClassifierGaussianNB_leaves.py**:

1. We created a class called **RandomForestClassifierNB** that inherits from **ForestClassifier** base class. It is similar to **RandomForesClassifier** except for a few changes.
2. We added the attribute **NaiveBayesModelsDict** which maps a Naïve Bayes (NB) model of the scikit-learn library called **GaussianNB** to each leaf node in each tree estimator.
3. We overloaded the method **fit**. It first calls the fit method of **ForestClassifier** and then use the base class' method **apply** to fetch all the observations in the estimators' leaves. It then maps and fits the NB models for each tree leaf with the method **add_models**.
4. Method **add_models** takes as input a tree estimator, a dictionary that maps the leaves indices with the observations in each leaf and the dataset and maps a NB model to each leaf in the **NaiveBayesModelsDict**, then fits the model with the relevant observations.
5. We overloaded the method **predict**. It calls the method **predict_proba_NB** which accumulates the estimators' probabilities with **_accumulate_prediction_NB**. Both methods work exactly like

predict_proba and **_accumulate_prediction** but use a different method for computing the predictions of all the tree estimators. Instead of using each estimator's **predict_proba** method, it uses a method called **prediction**.

6. **prediction** method gets a tree estimator and a dataset (with no targets) and returns the class probability predictions of that estimator by using the relevant fitted NB model for each observation. It also uses **apply** method to see in which leaf the observation is at.

Evaluation Plan and Hyperparameters Configurations

Our evaluation plan is constructed based on the assumptions we made about the influence of specific hyperparameters on the performance of each of the two models – RF classifier and RF with NB classifier. In the evaluation we compare the two classifiers and evaluate the influence of two hyperparameters we assume will have the most influence on the models, and will have to be changed from their default values for the improved model to perform better:

1. **min_samples_leaf**

   The minimum number of samples that a tree estimator's leaf can have. The RF classifier's default value for this hyperparameter is 1. Applying this value means the trees of the RF classifier can branch out and achieve high level of certainty in its predictions (maybe risking overfitting). Applying this low value on the RF with NB classifier means that the NB models will have an insufficient number of samples to train on and make it redundant. To take full advantage of the NB models, we assume it will have to train on a few dozen samples, so they will contribute to the overall performance of the improved model. However, a value too high might achieve a contradicting effect, making the RF model irrelevant or too inaccurate with almost no branching. Thus, the optimal number of minimum samples will have to be found so it will not be too high or too low. The number of records and number of features in each dataset will probably affect the optimal number so it will have to be found for each dataset separately.

2. **Max_depth**

   The maximum depth of the tree, the default value is None. Applying this value means that the nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. Similarly, to the previous min_samples_leaf parameter, the depth of the trees can be limited, achieving a higher number of samples to train for the NB model at each leaf resulting in a better NB model. However, trees without depth will lower the strength of the tree's classification performance and thus of the RF.

   Again, the number of records and number of features in each dataset will affect the optimal defth.

Hyperparameters Configurations

To check our assumption, we evaluated the two classifiers on 5 different hyperparameters configurations and 10 datasets. We left most of the hyperparameters in their default values and searched for the best configuration of min_samples_leaf and max_depth. The first configuration was chosen with a random greed search using **RandomizedSearchCV** from the scikit-learn library. We defined the search space to be as follows:

min_samples_leaf = [5, 10, 15, 20, 25, 30, 40, 50]

max_depth = [2, 4, 6, 8, 10, 12, 14, 16, 20, 25]

The search was conducted given the training set which constituted 0.8 of the entire dataset and was evaluated on the unseen samples from the test set. The search was done using 4-fold cross validation and for 10 iterations.

The next 4 configurations were manually picked to evaluate the influence of the two mentioned hyperparameters as they increased or decreased:

c1: {'max_depth': 50, 'min_samples_leaf': 1}

c2: {'max_depth': 10, 'min_samples_leaf': 5}

c3: {'max_depth': 6, 'min_samples_leaf': 15}

c4: {'max_depth': 2, 'min_samples_leaf': 25}

Number of estimators was fixed in all configurations to 50.

All four configurations were evaluated on both classifiers with 5-fold cross validation sets and the results were averaged.

Metrics

We used four different metrics suited for multi-class classification to evaluate the performance of each classifier on each configuration:

1. **Accuracy**

$$accuracy(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1} 1(\hat{y}_i = y_i)$$

   This metric could be problematic in unbalanced datasets, so we made sure the datasets are approximately balanced and added three more metrics.

2. **Macro-average precision**

   Precision is computed by:

$$precision = \frac{TP}{TP + FP}$$

   For multi-class problems we can use macro-averaging of the precision (or recall) which is a method for averaging the sum of precision of each label.

3. **Macro-average Recall**

   Similarly, we use macro-averaging over the labels' recall values:

$$precision = \frac{TP}{TP + FN}$$

4. **Macro-average F1**

   F1 is the harmonic-mean of precision and recall and computed by:

$$F_1 = 2\frac{precision \cdot recall}{precision + recall}$$

Another important factor we decided to evaluate and compare is **time** of the training process of each classifier and of the prediction process.

Results

The two classifiers were tested on ten different datasets of classification problems, obtained from here over all configurations' including the random greed searched configuration (RGS)

1. **QSAR biodegradation [link]**

   Number of instances: 1055

   Number of attributes: 41

   Number of classes: 2

   Best configuration RF: min_samples_leaf = 5, max_depth = 15 (RGS)

   Best configuration RF & NB: min_samples_leaf = 5, max_depth = 10 (c1)

	RF	RF with NB
Accuracy	0.872	0.886
Precision*	0.87	0.872
Recall*	0.848	0.866
F1*	0.857	0.869
Time [sec]	0.146	5.816

*macro-averaged

2. **Glass identification [link]**

   Number of instances: 214

   Number of attributes: 10

   Number of classes: 6

   Best configuration RF: min_samples_leaf = 5, max_depth = 10 (c2)

   Best configuration RF & NB: min_samples_leaf = 5, max_depth = 10 (c2)

	RF	RF with NB
Accuracy	0.698	0.674
Precision*	0.799	0.707
Recall*	0.691	0.712
F1*	0.678	0.685
Time [sec]	0.04	2.415

*macro-averaged

3. Image segmentation [link]

Number of instances: 2310

Number of attributes: 19

Number of classes: 7

Best configuration RF: min_samples_leaf = 1, max_depth = 50 (c1)

Best configuration RF & NB: min_samples_leaf = 5, max_depth = 10 (c2)

	RF	RF with NB
Accuracy	0.978	0.978
Precision*	0.9776	0.98
Recall*	0.978	0.979
F1*	0.9778	0.979
Time [sec]	0.186	9.88

*macro-averaged

4. Indian Liver Patient [link]

Number of instances: 583

Number of attributes: 10

Number of classes: 2

Best configuration RF: min_samples_leaf = 50, max_depth = 8 (RGS)

Best configuration RF & NB: min_samples_leaf = 5, max_depth = 10 (c2)

	RF	RF with NB
Accuracy	0.741	0.75
Precision*	0.561	0.661
Recall*	0.532	0.643
F1*	0.528	0.654
Time [sec]	0.215	5.77

*macro-averaged

5. ISOLET [link]

Number of instances: 7797

Number of attributes: 617

Number of classes: 26

Best configuration RF: min_samples_leaf = 1, max_depth = 50 (c1)

Best configuration RF & NB: min_samples_leaf = 5, max_depth = 10 (c2)

	RF	RF with NB
Accuracy	0.9374	0.9334
Precision*	0.9384	0.9366
Recall*	0.938	0.9336
F1*	0.9372	0.9326
Time [sec]	7.586	63.006

*macro-averaged

6. MAGIC Gamma telescope [link]

Number of instances: 19020

Number of attributes: 11

Number of classes: 2

Best configuration RF: min_samples_leaf = 1, max_depth = 50 (c1)

Best configuration RF & NB: min_samples_leaf = 40, max_depth = 16 (RGS)

	RF	RF with NB
Accuracy	0.878	0.881
Precision*	0.8774	0.882
Recall*	0.8506	0.853
F1*	0.8614	0.864
Time [sec]	3.17	73.1

*macro-averaged

7. **Libras Movement [link]**

Number of instances: 360

Number of attributes: 91

Number of classes: 24

Best configuration RF: min_samples_leaf = 5, max_depth = 10 (c2)

Best configuration RF & NB: min_samples_leaf = 20, max_depth = 10 (RGS)

	RF	RF with NB
Accuracy	0.764	0.778
Precision*	0.788	0.785
Recall*	0.772	0.781
F1*	0.748	0.759
Time [sec]	0.07	5.815

*macro-averaged

8. **Wilt [link]**

Number of instances: 4889

Number of attributes: 6

Number of classes: 2

Best configuration RF: min_samples_leaf = 1, max_depth = 50 (c1)

Best configuration RF & NB: min_samples_leaf = 15, max_depth = 6 (c3)

	RF	RF with NB
Accuracy	0.9814	0.9842
Precision*	0.9518	0.9622
Recall*	0.858	0.8782
F1*	0.8986	0.911
Time [sec]	0.278	17.97

*macro-averaged

9. Wine classification [link]

Number of instances: 178

Number of attributes: 13

Number of classes: 3

Best configuration RF: min_samples_leaf = 20, max_depth = 25 (RGS)

Best configuration RF & NB: min_samples_leaf = 20, max_depth = 25 (RGS)

	RF	RF with NB
Accuracy	1	1
Precision*	1	1
Recall*	1	1
F1*	1	1
Time [sec]	0.085	2.835

*macro-averaged

10. Iris [link]

Number of instances: 150

Number of attributes: 4

Number of classes: 3

Best configuration RF: min_samples_leaf = 1, max_depth = 50 (c1)

Best configuration RF & NB: min_samples_leaf = 25, max_depth = 16 (RGS)

	RF	RF with NB
Accuracy	0.94	0.967
Precision*	0.942	0.974
Recall*	0.941	0.963
F1*	0.94	0.967
Time [sec]	0.05	1.92

*macro-averaged

Results summary

The following table summarize the results by averaging the best scores of all the datasets. We also included the average number of min_samples_leaf and max_depth of the best configurations.

	RF	RF with NB
Accuracy	0.879	0.883
Precision*	0.871	0.875
Recall*	0.841	0.863
F1*	0.843	0.862
Time [sec]	1.183	18.876
min_samples_leaf	9	14.5
max_depth	31.8	12.3

Significance of the results

Accuracy results

In order to test the significance of the accuracy results between the two methods, we performed a paired samples t-test (one tail, α=0.05, null H-there isn't a significant difference between RF method and RFNB accuracy results), the output:

	RFNB	RF
Mean	0.88316	0.87898
Variance	0.01277326	0.011904368
Observations	10	10
Pearson Correlation	0.99324925	
Hypothesized Mean Difference	0	
df	9	
t Stat	0.980371078	
P(T<=t) one-tail	0.176259984	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.352519968	
t Critical two-tail	2.262157163	

We got a T-value of 0.98 and we need a T value of at least 1.83 to reject the null hypothesis. Thus, we can't reject the null hypothesis- No one of the two methods has significant better accuracy results.

F1

In order to test the significance of the F1 results between the two methods, we performed a paired samples t-test (one tail, α=0.1, null H-there isn't a significant difference between RF method and RFNB F1 results), the output:

	RFNB	RF
Mean	0.86206	0.8426
Variance	0.015155369	0.022245
Observations	10	10
Pearson Correlation	0.978175987	
Hypothesized Mean Difference	0	
df	9	
t Stat	1.599890201	
P(T<=t) one-tail	0.072043587	
t Critical one-tail	1.383028738	
P(T<=t) two-tail	0.144087175	
t Critical two-tail	1.833112933	

We got a T-value of 1.59 and we need a T value of at least 1.38 to reject the null hypothesis. Thus, we can reject the null hypothesis- the RF with the NB model F1 results are significantly better.

The full results and analysis are available at the results.csv and results_for_statistical_analysis.xls files.

Conclusions

1. The Random Forest model with the integration of Naïve Bayes in the estimators' leaves managed to perform better than the RF base model in most of the datasets (7/10) over all four metrics – accuracy, precision, recall and F1. The average results also indicate the varied model has a certain advantage over the base model and is significantly better looking at the F1 metric.
2. In average, the best configuration for RF with NB used a larger minimum number of samples in the leaves of the trees than the base RF model and a smaller tree with a lower number of the trees' maximum depth. Furthermore, in many cases the minimum number of samples was 1 (configuration c1) in the RF model, but RF with NB model presented the worse results in this configuration. The RF with NB model must have at

least several samples in each leaf for it to train properly and improve the classification ability of the model. The results strengthen our pre-assumptions.

3. The most significant metric the RF with NB model improved was the recall, meaning that the model has a better ability to find the relevant cases within the dataset which could be important in some classification problems.
4. The combined time of fitting the model on the training set and predicting the test set is significantly higher in the varied model, even though we used the same parallelization method as the one in the base model. This can be explained by the greater complexity of the model that has several NB models as the number of leaves in the forest. The time factor can be very important in some cases and might rule out using the model on the expense of reduction in performance in other metrics.
5. The size of the datasets has an influence on the results, when a more complex method needs more data to train on. The complexity of the model also means it needs more tuning for it to perform well, so we assume that with more time-consuming tuning of the hyperparameters the improved model would have perform even better.