Nanyang Polytechnic

ITI104 MACHINE LEARNING ALGORITHMS

Assignment

Date of Submission: 16-AUG-2020

Submitted By:

20A459H LIM YUAN HER

Table of Contents

List of Tables

List of Figures

1 Introduction and Background

The use of ensemble learning in machine learning, specifically voting and stacking ensembles, involve the use of heterogenous models as base classifiers trained on different algorithms but using the same training set in order to produce prediction accuracies superior to that of individual classifiers. This report documents the study and exploration of these machine learning techniques.

2 Voting

2.1 Description

This involves training the data set using different algorithms and the final output prediction is determined by majority vote via hard or soft voting. The figure below illustrates the typical voting classification scheme:

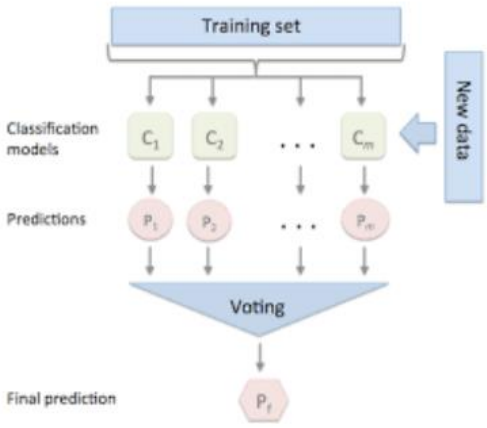

Figure 1 – Voting Classifier [1]

2.2 Voting Schemes

2.2.1 Hard (Majority)

In this case, the output predicted class is the one corresponding to the mode value of the combined predicted classes from all base classifiers.

2.2.2 Soft

In this case, the probability vector for each predicted class (for all classifiers) are summed up and averaged. The output predictor class is the one corresponding to the highest value.

2.2.3 Weighted

In this case, the output prediction is the weighted average value of the predictor classes from each classifier (applicable for regression use cases).

2.3 Implementation

Voting Classifiers are implemented in the following packages:

s/n	Package	Classification	Regression
1	Mlxtend [4]	EnsembleVoteClassifier	-
2	Scikit-Learn [5]	VotingClassifier	VotingRegressor

Table 1 – Voting Implementations

2.4 Applicable Use Cases

Voting is useful when a single model shows some kind of bias as it aggregates the predictions of several models and tries to cover for the potential weaknesses of the individual models. One way of improving the voting ensemble's performance is by selecting base estimators that are as diverse as possible.

Hard voting is useful for models that predict crisp class labels whilst soft voting is useful for models that predict class membership probabilities. Soft voting can still be used for models that do not natively predict class membership probabilities, but requires calibration of their probability-like scores e.g. support vector machines etc.

In addition, voting can achieve lower output prediction variances as compared to individual models. However, this may negatively impact on the overall mean performance, but this might still be desirable given the higher stability of the model.

Also, voting can also be useful in cases where multiple fits of the same classifier model with slightly different hyperparameters are combined in a voting ensemble.

2.5 Limitation

Voting treats all models equally i.e. all models contribute equally to the prediction, thus poorly performing model(s) within the model mix can negatively impact on the overall performance, though this can be mitigated by choosing base classifiers that uses different

3 Stacking

3.1 Description

Stacking was first introduced by David H. Wolpert 1992 in his paper *Stacked Generalization* [9]. It is most famously used in the NetFlix competition [10] to bag the grand prize and is very commonly used in data science competitions e.g. Kaggle [11]. This form of ensemble learning is like that of voting schemes except that instead of using hard or soft voting to determine the final output prediction, a meta-classifier is trained on the predictions of the base classifiers instead. Stacking comes in 2 forms, single-level or multi-level as described in the sections below.

3.1.1 Single-Level Stacking

The figure below illustrates the typical single-level stacking classification scheme:

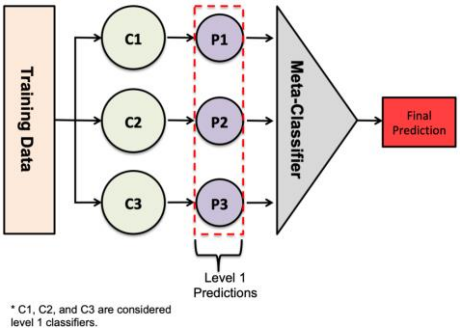

Figure 2 – Stacking Classifier [2]

This involves training a set of base classifiers using different algorithms and the final output predictions are used as new features to train a meta-classifier which makes the final prediction

3.1.2 Multi-Level Stacking

Figure 3 – Multi-level Stacking [3]

In multi-level stacking, additional levels of base classifiers are added to the model thus increasing the complexity of the model with each level (see illustration above). At each level, the predictions from the prior level classifiers are used as inputs to train new models until the final level where the predictions are used as inputs to a meta-classifier to produce the final output prediction.

However, without proper cross-validation, problems like overfitting can occur when many levels are involved. In addition, there is no recognized method as to the choice of base classifiers used for each level, which limits the practicality of using multi-level stacking as a solution. Nevertheless, multi-level stacking is a plausible solution in use cases where bias reduction using plain basic or ensemble models or even voting schemes have reached practical limits.

3.2 Implementation

Stacking implementations are available in the python packages as listed in the table below:

s/n	Package	Classification	Regression	Single-Level	Multi-Level
1	Mlxtend [4]	StackingClassifier/ StackCVClassifier	StackingRegressor/ StackCVRegressor	x	x
2	Scikit-Learn [5]	StackingClassifier	StackingRegressor	x	x
3	Vecstack [6]	StackingTransformer	StackingTransformer	x	x
4	Pycaret [7]	stack_models	models	x	x
5	ML-Ensemble [8]	SuperClassifier	SuperClassifier	x	x

Table 2 – Stacking Implementations

3.3 Applicable Use Cases

Stacking is useful when different models produce predictions where the errors in predictions are uncorrelated or have a low correlation to enhance the performance of the meta-classifier.

Base models chosen should be varied enough as they will make very different assumptions about how to solve the predictive modelling task e.g. linear models, decision trees, support vector machines etc. whilst the meta-classifier chosen should be simple enough to provide a smooth interpretation of the predictions made by the base models e.g. linear regression (regression), logistic regression (classification).

3.4 Limitation

Stacking performance is dependent on the choice of base models and whether they are a good enough approximation of the input data and are sufficiently uncorrelated in their predictions (or errors).

4 Comparison

Voting and stacking both make use of multiple basic/ensemble classifiers to build high performance models but using different approaches to achieve this. Thus, this results in a diversity of pros and cons when adopting each type.

The differences between voting and stacking are listed in the table below:

s/n	Voting	Stacking
1	Involves only heterogenous base classifiers only	Involves both heterogenous base classifiers and meta classifier
2	Supports single level scheme only	Supports both single and multi-level schemes
3	Uses hard/soft voting to produce final output predictor class	Uses meta classifier trained on features based on output predictions of base classifiers to produce final output predictor class
4	Simpler due to use of simple majority/ arithmetic averaging to determine output	More complex due to need for training meta classifier to determine output
5	Output prediction accuracy is dependent on performance of base classifiers only	Output prediction accuracy depends on performance of both base classifiers and meta classifier
6	Shorter training times as only training of base leaners is required	Longer training times as training of both base classifiers and meta classifier are required
7	Easier to interpret predictions as only single-level and pre-defined output function (majority vote/averages)	Difficult to interpret predictions due to complex level stacking and meta-classifier intermediaries
8	Straightforward selection of base classifiers for voting model as all base classifiers are treated equally	Difficult to select combinations of base classifiers resulting in optimal performance, especially in multi-level stacking classifier model
9	Overall performance still dependent on individual base classifiers	Use of stacking can result in performance benchmarks not achievable by other model types e.g. voting/ensemble/basic

Table 3 – Voting/Stacking Comparison

5 Advantages and benefits

5.1 Advantages

Both voting and stacking schemes adopt a non-biased approach by taking advantage of different algorithms to produce better performance than any single one by aggregating the predictions of multiple models to compensate for potential weaknesses of the individual models. There are 3 cases in which using voting/stacking can provide a more optimal solution than single classifiers as described in the sections below:

5.1.1 Limited Training/Test Data

When limited training and testing data is available, evaluation of single classifiers using test sets will exhibit high variance due to high dependence on small number of train/test instances. Using voting/stacking schemes will reduce the possibility of being misled by a bad classifier exhibiting good performance on a small but non-representative test set.

5.1.2 Hill-climbing/random search type classifiers

Some classifiers using hill-climbing/ random search techniques relying on random initializations exhibit different convergence routes to an optimal solution during training. Using voting/stacking schemes to aggregate such classifiers can tap on the diversity of solutions offered by these classifiers, of which none may be optimal, to arrive at a solution that is better than the solution offered by any one individual classifier.

5.1.3 Linear/Non-linear classifiers

In cases where non-linear classifiers perform best, if linear classifiers are used instead, the resultant performance will be adversely affected. Using voting/ stacking schemes can derive an approximate non-linear decision boundary, which could potentially perform better than any single linear classifier possibly could.

6 Applications

Multiple classifier systems e.g. voting/stacking have wide applicability in various problem domains where they can overcome the limitations of single/ homogeneous classifier systems. The domains in which such systems have been researched/ implemented [12] include:

1. Remote sensing – including land cover mapping and change detection based on multispectral satellite images
2. Computer Security – Distributed denial of service (DDOS) attack, malware detection, intrusion detection, wireless sensor network anomaly detection
3. Banking – Fraud detection e.g. credit card, money laundering etc., credit risk prediction, financial risk modelling
4. Medicine – coronary disease detection, proteomics, neuroscience
5. Recommendation Systems – product recommendation, telecommunications subscriber churn prediction, conference papers review

The list above covers a broad spectrum of areas where multi-classifier systems are involved in, but is by no means the definitive list as new areas are under active research and development.

With further technological advancements in semi-conductor electronics and improving computing prowess, it is envisaged that use of such systems will become more practically feasible and prevalent.

7 Conclusion

Even though the use of ensemble methods like voting and stacking can result in high accuracy, it is often not preferred in use cases where interpretability is more important e.g. healthcare etc. In addition, other consideration factors e.g. training times, memory utilization, prediction latency etc. need to be accounted for due to higher computational/memory requirements as compared to single classifier models.

Nonetheless, the use of ensemble methods is a viable alternative for use cases whereby model high bias reduction cannot be achieved using single classifier models or ensemble methods e.g. bagging/ boosting.

8 References

[1] *Voting Classifier* https://medium.com/@sanchitamangale12/voting-classifier-1be10db6d7a5.

[2] *Stacking Classifiers for Higher Predictive Performance* https://towardsdatascience.com/stacking-classifiers-for-higher-predictive-performance-566f963e4840.

[3] *Stacking made easy with Sklearn* https://towardsdatascience.com/stacking-made-easy-with-sklearn-e27a0793c92b

[4] *Mlxtend* http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/

[5] *Scikit-Learn* https://scikit-learn.org/stable/modules/ensemble.html#stacked-generalization

[6] *Vecstack* https://github.com/vecxoz/vecstack#usage-scikit-learn-api

[7] *PyCaret* https://pycaret.org/stack-models/

[8] *ML-Ensemble* http://ml-ensemble.com/info/tutorials/start.html#building-an-ensemble

[9] *Stacked Generalization*, David H. Wolpert, 1992

[10] *The BigChaos Solution to the Netflix Grand Prize*, Andreas Toscher, Michael Jahrer https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

[11] *Why do stacked ensemble models win data science competitions?* https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/

[12] *A survey of multiple classifier systems as hybrid systems*, Michal Wozniak, Information Fusion 16(1):, 3–17 March 2014