CS513-Data Mining

Lecture 8: Ensemble Techniques: Mixing Classifier

Waheed Noor

Computer Science and Information Technology,
University of Balochistan,
Quetta, Pakistan

Outline

Outline

Ensemble Techniques: Mixing Classifiers I

- Every data mining model have distinguished characteristics with their own strengths and weaknesses on different problems.
- The performance of these models depends on factors such as the quantity and quality of training data, and appropriateness of the model to the problem at hand
- Generally, when making critical decision, often services of many experts of the domain are taken into account rather than relaying on the judgment of a single expert
- Here we can assume a learned model as an expert making prediction on test set and oftentime our objective is to get improved predictive performance

Ensemble Techniques: Mixing Classifiers II

- Therefore, we can extend the idea of multiple experts to classification by combining/mixing the prediction of multiple classifiers/models to improve the prediction performance
- Intuitively, this idea is helpful when these models are significantly different from each other and learning the reasonable concepts of the problem correctly in order to compensate one another

Outline

Types of Ensemble Techniques

Some prominent ensemble techniques of combining different learning techniques are:

- Bagging
- Boosting
- Stacking

They have been proved to provide improved performance and some research have been dedicated to identify such factors.

Reading

There is also another technique called *error-correcting output codes*, that, in addition to staking, left as a reading assignment for you

Outline

Bagging: Bootstrap Aggregating

- In bagging, different models of same algorithm are built using different training sets of same size by random sampling by replacement from the actual training data
- Then classification for the test instance is made by majority vote among all classifiers learned
- In case of numeric prediction; average of the predicted value is taken
- Though it is not guaranteed to always achieve better performance through bagging but this technique performs well in practice
- One evidence from theoretical perspective of bagging is through *bias-variance decomposition* that attempts to quantify 1) how well a learning algorithm matches the problem at hand and 2) noise in the training set, in the form of expected error.

Bias-Variance Decomposition

Assumption:

Infinite number of independent training sets of same size to built infinite number of classifiers and prediction for infinite number of independent test instances is made by majority vote. Expected error is calculated by averaging the error of combined classifier.

Definition (Bias)

The error rate for a particular learning algorithm is called its bias for the learning problem that measures how well it is inline with the problem, i.e., persistent error.

Definition (Variance)

Since training set is finite and not fully representative of general population, therefore expected error for all possible training and test instances is the variance of the learning algorithm for the problem.

Then the total expected error will be the sum of bias and variance.

Discussion

- Bagging is good for unstable classifiers such as decision trees where small changes in training set result in high error in prediction due to bias-variance decomposition
- Since in decision trees small change in training set can effect the selection of a feature as node during induction process of decision trees
- Bagging may not be suitable for small training sets due to the fact it creates several training sets from the original training set applying re-sampling by replacement
- Importantly, during testing, bagging assign equal weights to the predictions of the base classifiers
- Bagging is not iterative, i.e., each participating classifier (base classifier) is built separately

Outline

Boosting I

- Boosting is similar to bagging in combining the prediction performance of similar learning algorithms by majority voting (for numeric averaging the prediction)
- However, boosting exploits the intuition of combining multiple models by explicitly seeking models that compensate one another by weighted vote/average
- Moreover, boosting is an iterative procedure where each new model is built on the performance of the model previously built such that the performance of new model is boosted on the instances that are incorrectly predicted previously

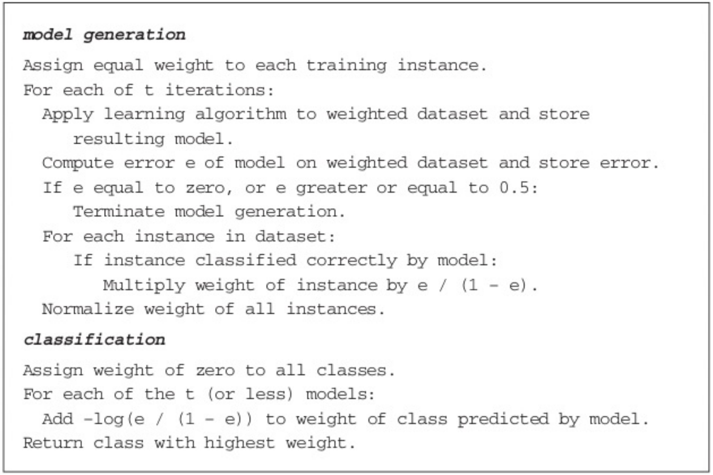
Boosting II

- That is in boosting such instances (hard examples) are more heavily weighted in the next round when building the new model that in result force the new model to concentrate on such hard examples.
- The error in the presence of weighted instances is fraction of sum of weights of misclassified instances and sum of weights of all instances, which we denote by ϵ
- In each iteration of boosting, weights for correct instances, w, are updated $w = w \times \epsilon / 1 - \epsilon$ and remain unchanged for the misclassified instances
- Then these weights are normalized to keep the total weight same by dividing the weight of each instance with the sum of new weights and multiplying with the sum of old weights

Boosting III

- In this way, weights of correctly classified examples is decreased while it is increased for incorrectly classified examples
- The classifier with $\epsilon \geq 0.5$ and $\epsilon = 0$ are removed and iteration stops
- Then the weight for voting by each classifier to make final prediction is determined by $w_v = -\ln \frac{\epsilon}{1-\epsilon}$ that is between $\{0 - \infty\}$
- That is why we remove classifier with $\epsilon = 0$ to avoid infinity while $\epsilon = 0.5$ results in zero weight
- This measure ensures to give higher weight to the classifier with ϵ close to 0 and less weight to the classifier with ϵ close to 0.5
- Prediction for test instance is the class with highest weight after summing the weights of all classifier predicting the same class

Pseudopod of Booting Algorithm



```
model generation

Assign equal weight to each training instance.
For each of t iterations:
  Apply learning algorithm to weighted dataset and store
     resulting model.
  Compute error e of model on weighted dataset and store error.
  If e equal to zero, or e greater or equal to 0.5:
     Terminate model generation.
  For each instance in dataset:
     If instance classified correctly by model:
        Multiply weight of instance by e / (1 - e).
  Normalize weight of all instances.

classification

Assign weight of zero to all classes.
For each of the t (or less) models:
  Add -log(e / (1 - e)) to weight of class predicted by model.
Return class with highest weight.
```

Figure : Boosting Algorithm

Discussion

- The boosting algorithm just explained is called Adaboost.M1, though there exists some other variants of boosting as well
- Boosting is considered to be the most powerful ensemble techniques which is closely reltated to well established statistical technique of additive models
- Moreover, boosting is actually originated from the branch of machine learning research called *computational learning theory*
- If an algorithm can not handle weighted data directly then the orignial training data can be re-sampled where the selection probability of an instance is proportioned to its weight.
- This method will let the hard examples to repeat more frequently than the easy ones

References I

Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer, New York, 2006.

Ian H. Witten and Eibe Frank.
Data Mining: Practical Machine Learning Tools and Techniques, Second Edition.
Morgan Kaufmann, San Francisco, CA, 2005.