A short introduction to ensemble learning

1 Ensemble learning

Ensemble learning is a very powerful paradigm that can produce high prediction accuracies for classification tasks. A lot of winning algorithms in Kaggle involve designing proper ensemble learning algorithms for the problem at hand.

The motivation of ensemble learning is to combine a lot of weak classifiers that are slightly better than random guessing to produce a powerful committee and to get strong classifiers that can make accurate predictions [3]. The weak classifiers can be decision trees, neural networks or other classifiers, which can be trained in a parallel style, e.g. bagging or in a sequential style, e.g. boosting. Then the results of the weak classifiers can be combined for prediction, for example, majority voting for classification.

Here we give a short introduction to a popular boosting method, called AdaBoost.

2 Boosting: AdaBoost

This section is based on [2].

The motivation for boosting is a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee". The most popular boosting algorithm is called "AdaBoost.M1", which is proposed by Freund and Schapire [1].

Consider a two-class problem, with the output variable coded as $Y \in \{1, 1\}$. Given a vector of predictor variables $\mathbf{X}$, a classifier $G(\mathbf{X})$ produces a prediction taking one of the two values $\{1, 1\}$. The error rate on the training sample is

$$\bar{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(\mathbf{x}_i)),$$

and the expected error rate on future predictions is $E_{XY} I(Y \neq G(\mathbf{X}))$.

A weak classifier is one whose error rate is only slightly better than random guessing. The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of the data, thereby producing a sequence of weak classifiers $G_m(\mathbf{x}), \ \ m = 1, 2, \ldots, M$.

The predictions from all of them are then combined through a weighted majority vote to

produce the final prediction:

$$G(\mathbf{x}) = \text{sign}[\sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})].$$

Here $\alpha_1, \alpha_2, \ldots, \alpha_M$ are computed by the boosting algorithm, and weight the contribution of each respective $G_m(\mathbf{x})$. Their effect is to give higher influence to the more accurate classifiers in the sequence.

The data modifications at each boosting step consist of applying weights $w_1, w_2, \ldots, w_N$ to each of the training observations $(\mathbf{x}_i, y_i)$, $i = 1, 2, \ldots, N$. Initially all of the weights are set to $w_i = 1/N$, so that the first step simply trains the classifier on the data in the usual manner. For each successive iteration $m = 2, 3, \ldots, M$ the observation weights are individually modified and the classification algorithm is reapplied to the weighted observations. At step m, those observations that were misclassified by the classifier $G_{m-1}(\mathbf{x})$ induced at the previous step have their weights increased, whereas the weights are decreased for those that were classified correctly. Thus as iterations proceed, observations that are difficult to classify correctly receive ever-increasing influence. Each successive classifier is thereby forced to concentrate on those training observations that are missed by previous ones in the sequence.

The detailed algorithm of Adaboost is as follows.

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(\mathbf{x})$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(\mathbf{x}_i))}{\sum_{i=1}^{N} w_i}$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \exp(\alpha_m I(y_i \neq G_m(\mathbf{x}_i)))$, $i = 1, 2, \ldots, N$.
3. Output $G(\mathbf{x}) = \text{sign}[\sum_{i=1}^{N} \alpha_m G_m(\mathbf{x})]$.

The current classifier $G_m(\mathbf{x})$ is induced on the weighted observations at line 2a. The resulting weighted error rate is computed at line 2b. Line 2c calculates the weight m given to $G_m(\mathbf{x})$ in producing the final classifier $G(\mathbf{x})$ (line 3). The individual weights of each of the observations are updated for the next iteration at line 2d. Observations misclassified by $G_m(\mathbf{x})$ have their weights scaled by a factor $\exp(\alpha_m)$, increasing their relative influence for inducing the next classifier $G_{m+1}(\mathbf{x})$ in the sequence.

References

[1] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[2] Hastie Trevor, Tibshirani Robert, and Friedman JH. The elements of statistical learning: data mining, inference, and prediction, 2009.

[3] Zhi-Hua Zhou. Ensemble learning. *Encyclopedia of Biometrics*, pages 411–416, 2015.