ACTA UNIVERSITATIS LODZIENSIS
FOLIA OECONOMICA 196, 2006

Eugeniusz Gatnar[*], *Dorota Rozmus*[**]

RANDOM SELECTION OF VARIABLES FOR AGGREGATED TREE-BASED MODELS

Abstract. Tree-based models are popular a widely used because they are simple, flexible and powerful tools for classification. Unfortunately they are not stable classifiers.

Significant improvement of the model stability and prediction accuracy can be obtained by aggregation of multiple classification trees. Proposed methods, i.e. bagging, adaptive bagging, and arcing are based on sampling cases from the training set while boosting uses a system of weights for cases. The result is called committee of trees, an ensemble or a forest.

Recent developments in this field showed that randomization (random selection of variables) in aggregated tree-based classifiers leads to consistent models while boosting can overfit.

In this paper we discuss optimal parameter values for the method of random selection of variables (*RandomForest*) for an aggregated tree-based model (i.e. number of trees in the forest and number of variables selected for each split).

Key words: Tree-based models, aggregation, *RandomForest*.

1. INTRODUCTION

The most successful aggregation methods in classification, i.e. bagging, adaptive bagging, and arcing are based on sampling cases from the training set while boosting is deterministic and uses a system of weights for cases and combined models.

Although resampling causes major modification of the distribution of predictors in the training samples, significant improvement of classification

[*] Professor, Department of Statistics, The Karol Adamiecki University of Economics, Katowice.

[**] Assistant, Department of Statistics, The Karol Adamiecki University of Economics, Katowice.

accuracy can be also achieved by random selection of variables to training samples or directly to the model.

Recent developments in this field showed that the randomization leads to consistent models while boosted models can overfit for large number of their components.

A classifier C is a function that maps from object descriptions ($\mathbf{X}$) to class names (Y):

$$C : \mathbf{X} \rightarrow Y \tag{1}$$

and it is found by learning a set of previously classified objects, called "training set" T:

$$T = \{(y_1, \mathbf{x}_1), (y_2, \mathbf{x}_2), ..., (y_N, \mathbf{x}_N)\}, \tag{2}$$

where $\mathbf{x}_i$ is an element from M-dimensional space $\mathbf{X}$, and y_i is an element from discrete space Y. The goal of the learning is to find a classifier $C(\mathbf{x})$ that gives the lowest prediction error.

In order to measure the error between y and $C(\mathbf{x})$ we use a loss function. In regression it has a form of the squared error:

$$L(y, \hat{C}(\mathbf{x})) = (y - \hat{C}(\mathbf{x}))^2, \tag{3}$$

and in classification it is a 0–1 function:

$$L(y, \hat{C}(\mathbf{x})) = \begin{cases} 0 \text{ if } y = \hat{C}(\mathbf{x}) \\ 1 \text{ if } y \neq \hat{C}(\mathbf{x}) \end{cases}. \tag{4}$$

Mostly the prediction error of $C(\mathbf{x})$ is estimated using an independent test set T. In the absence of this set one can use the training set, but the model is likely to overfit (the classification error is equal zero for large enough model). Different strategies are proposed to get better estimation of the error, e.g. cross-validation or bootstrapping.

2. CLASSIFICATION TREES

The tree corresponds to an additive model in the form of:

$$y = \sum_{k=1}^{K} a_k I(\mathbf{x} \in R_k), \tag{5}$$

where R_k are hyper-rectangular disjoint regions in the M-dimensional feature space, a_k denotes real parameters and I is an indicator function (Gatnar 2001).

Each real-valued dimension of the region R_k is characterized by its upper and lower boundary: $v_{km}^{(d)}$ i $v_{km}^{(g)}$ respectively. Therefore the region induces a product of M indicator functions:

$$I(\mathbf{x} \in R_k) = \prod_{m=1}^{M} I(v_{kn}^{(d)} \leqslant x_m \leqslant v_{km}^{(g)}), \tag{6}$$

If x_m is a categorical variable, the region R_k is defined as:

$$I(\mathbf{x} \in R_k) = \prod_{m=1}^{M} I(x_m \in B_{km}), \tag{7}$$

where B_m is a subset of the set of the variable values.

The parameter estimation formula depends on how the homogenity of the region R_k is measured. In the simplest case the entropy function is applied:

$$H(R_k) = -\sum_{j=1}^{J} p(j|k) \log_2 p(j|k). \tag{8}$$

The parameters of the model (5) are estimated using the majorization method:

$$a_k = \arg\max_j \{p(j|k)\}, \tag{9}$$

where $p(j|k)$ is the fraction of objects in region R_k that belong to the class j.

3. AGGREGATION

The successful tree-based methodology has one undesirable feature: lack of stability. That is a tree model depends on the training set and even small change in a predictor value could lead to a quite different model. To solve this problem, single trees are combined into one model and then averaged.

In the aggregate $C^*(\mathbf{x})$ the component models vote for the predicted class:

$$\hat{C}^*(\mathbf{x}) = \arg\max_y \left\{ \sum_{m=1}^{M} I(\hat{C}_m(\mathbf{x}) = y) \right\}. \tag{10}$$

Several variants of aggregation methods were proposed. They manipulate training cases (random sampling) or predictors (random selection) or values of the y (system of weights) or involve randomness directly.

Combined classifiers have lower error rate than single models and can give more insight. But they have two serious disadvantages

- slow learning (requires large computer memory),
- loss of comprehensibility (less structure and huge models).

Perhaps *stacking* or stacking generalization (Wolpert 1992) was the first method used for combining models in statistics. The models $\hat{C}_m^{-1}(\mathbf{x})$ are fitted to training samples U_m^{-1} obtained by leave-one-out cross-validation (i.e. with i-th observation removed).

Quinlan (1993) implemented in his C4.5 system a method called *windowing* that enlarges the initial random sample drawn from the training set by adding in consecutive steps cases misclassified in the previous step.

Bagging (Bootstrap AGGregatING) was the first aggregation method proposed by Breiman (1996). It used multiple training bootstrap samples $U_1, U_2, ..., U_M$ to create classifiers that vote for the final prediction (10).

There are two variants of bagging:

1. *Adaptive bagging* proposed by Breiman (1999) that works by changing output values y by using "out-of-bag" cases in subsequent steps:

$$y_i^{(s+1)} = y_i^{(s)} - \overline{\hat{C}(\mathbf{x}_i)}, \tag{11}$$

where $\overline{\hat{C}(\mathbf{x}_i)}$ is the average over predicted values for training samples U_m such that $\mathbf{x}_i \notin U_m$.

2. *Wagging* introduced by Bauer and Kohavi (1999) is similar to bagging, but reweights cases instead of sampling them from the training set.

The most accurate classifiers are those combined by adaptive boosting. The algorithm *AdaBoost.M1* developed by Freund and Schapire (1997) is deterministic, sequential and works with two systems of weights: assigned to training examples and assigned to component models.

Initially the weights of cases are uniform $w_i = \frac{1}{N}$, but in consecutive steps the weights of cases misclassified in the previous step are increased by a factor inversely proportional to the training sample error:

$$w_i^{(s+1)} = w_i^{(s)} \cdot e^{a_m I(\hat{C}_m(x_i) \neq y_i)}, \tag{12}$$

where:

$$a_m = \log\left(\frac{1 - e_m}{e_m}\right), \tag{13}$$

and the error rate e_m is:

$$e_m = \frac{\sum_{i=1}^{N} w_i I(\hat{C}_m(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^{N} w_i}. \tag{14}$$

The final prediction is:

$$\hat{C}^*(\mathbf{x}) = \arg\max_y \left\{ \sum_{m=1}^{M} a_m I(\hat{C}_m(\mathbf{x}) = y) \right\}. \tag{15}$$

Breiman referred to *AdaBoost* with trees as "the best classifier in the world".

There are two variants of boosting: *boosting by weighting* (AdaBoost.M1) and boosting by sampling (cases are sampled from the training set with probability proportional to their weights).

Because of the apparent success of *AdaBoost*, Breiman (1998) applied a system of weights to his new *Arc-x4* sequential algorithm (based on bootstrap sampling) called *Arcing* (Adaptive Resampling and CombinING). *Arc-x4* increases weights of misclassified cases:

$$w_i = 1 + [n(\mathbf{x}_i)]^4, \tag{16}$$

where $n(\mathbf{x}_i)$ is the number of misclassifications of the case $\mathbf{x}_i$ by classifiers: $C_1, C_2, \ldots, C_{m-1}$.

4. RANDOM SELECTION OF PREDICTORS

Methods presented so far are based on sampling cases to the training samples but there are other methods that use random selection of variables to the tree model.

The first method has been developed by Ho (1998) and called "Random subspaces". Each tree in the ensemble is fitted to the training sample containing all cases from the training set but with randomly selected features.

Random split selection has two variants. Dietterich and Kong (1995) proposed to select the split at random from among the K best splits, while Breiman (1999) proposed to select at random a small group of predictors and then to find the best of them that would form the split.

Recently Breiman (2001) developed a system involving random sampling called *RandomForests* (RF). It contains two procedures: RV and RC. In the first one at each node K variables are selected at random and the best split is chosen from among them.

In the other procedure K variables are randomly selected and added with coefficients that are uniform random numbers from $[-1, 1]$. Then L linear combinations are generated and the best one is selected for split.

RandomForests is equal or better than *AdaBoost* in classification accuracy, but it is robust to noise, faster and it does not overfit.

5. EXAMPLE

Using *RandomForests* to build a classifier two parameters should be taken into consideration: number of trees in the forest and number of features randomly chosen at each node of a single tree. These two parameters are responsible for the accuracy of the model.

As far as the first parameter is discussed, it can be said that the more ensembles are built, the better model we get. It is so because this kind of aggregation doesnot lead to overfitting. The second parameter is the only parameter that needs some judgement to set, but many experiments have shown that forests are not too sensitive to its value (Breiman 2001).

In order to check how these two parameters influence on the model performance, we used the data set "Satimage" from the UCI Repository (Blake et al. 1998). We have chosen it because of a large size (4435 objects) and the fact that there is a separate test set (2000 objects). Objects in this database are fragments of the Earth image from satellite "Landsat". Each object is described by 36 numerical features and belongs to one from six possible classes indicating a kind of ground exploitation.

The aim of the first study is to check the influence of the number of trees in the forest on the aggregated model error rate. The whole experiment was carried out due to the following scheme:

1. The number of features randomly selected at each node take as a constant, equal 6.

2. The number of trees take equal $L = 50$.

3. Build an aggregated model; calculate the error rate on learning set and test set.

4. Repeat the experiment 100 times and average the error rate, correspondingly for learning and test set.

5. Take $L = L + 50$ and go to step 3.

6. When $L = 500$, stop the experiment.

The result of the study is shown in Figure 1. What we can see is that adding more and more trees generally causes dropping of error rate, both on learning and test set. The tendency is more noticeable on learning set while on test set it shows tendency to stabilise. The biggest drop is noticed between 50 and 100 trees.

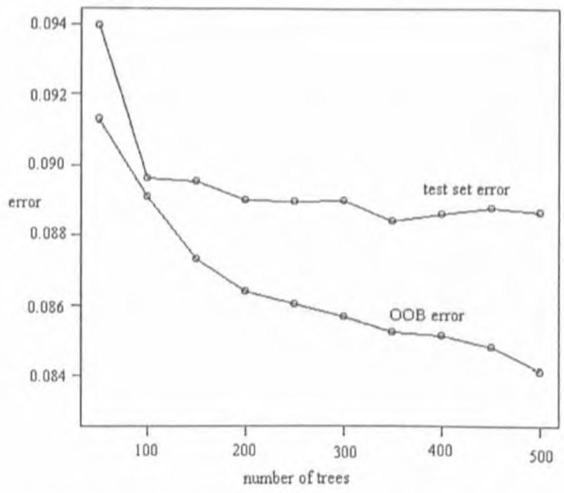

Fig. 1. Effect of the number of trees in the forest on error rate

The second experiment was carried out in order to find the result of the number of features randomly selected for every node on aggregated model performance. The scheme is as follows:

1. The number of trees in the forest take as a constant, equal 100.

2. The number of features take equal $C = 1$.

3. Build an aggregated model; calculate the error rate on learning set and test set.

4. Repeat the experiment 100 times and average the error rate, correspondingly for learning and test set.

5. Take $C = C + 1$ and go to step 3.

6. When $C = 30$, stop the experiment.

Again we get series of values of learning and test set error that is shown in Figure 2. At the beginning, choosing more and more features causes stable drop in error rate, both in learning and test set. Than they increase a little, drop again and then show stable tendency for increasing.

The computer program, *RandomForest*, uses a special formula to estimate the error rate on learning set (*out-of-bag*). That is why we decided to compare the gained results with much more popular way of computing error rate, i.e. *cross-validation*. Using procedure *boosting* that is available in R we have estimated the error rate, dividing the whole learning set into 10 parts. As is shown in Figure 2, the tendency of CV error is generally the same as described above; and it is a little bigger than the OOB error.

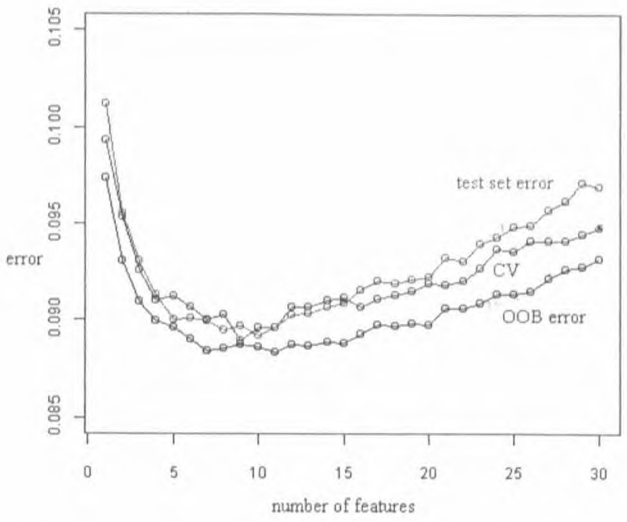

Fig. 2. Effect of number of randomly selected features on error rate

Results from the second study correspond to results described by Breiman (2003). He found that setting number of trees equal to the square root of number of all number of variables gives generally near optimum results. He advised to begin with this value and try a value twice as high and half as low, monitoring the result on error rate.

REFERENCES

Blake C., Keogh E., Merz C. J. (1998), *UCI Repository of Machine Learning Databases*, Departament of Information and Computer Science, University of California, Irvine, CA.

Bauer E., Kohavi R. (1999), "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants", *Machine Learning*, **36**, 105–142.

Breiman L. (2003), Manual on Settings up, Using and Understanding Random Forest, http://oz.berkeley.edu/users/breiman/UsingrandomforestsV3.1.

Breiman L. (2001), "Random Forests", *Machine Learning*, **45**, 5–32.

Breiman L. (1999), "Using Adaptive Bagging to Debias Regressions", *Technical Report* **547**, Statistics Department, University of California, Berkeley.

Breiman L. (1998), "Arcing Classifers", *Annals of Statistics*, **26**, 801–849.

Breiman L. (1996), "Bagging Predictors", *Machine Learning*, **24**, 123–140.

Dietterich T., Kong E. (1995), "Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms", *Technical Report*, Department of Computer Science, Oregon State University.

Freund Y., Schapire R. E. (1997), "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", *Journal of Computer and System Sciences*, **55**, 119–139.

Gatnar E. (2001), *Nonparametric Method for Discrimination and Regression*, (in Polish Wydawnictwo Naukowe PWN, Warszawa.

Ho T. K. (1998), "The Random Subspace Method for Constructing Decision Forests", *IEEE Trans. on Pattern Analysis and Machine Learning*, **20**, 832–844.

Quinlan J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo.

Wolpert D. (1992), "Stacked Generalization", *Neural Networks*, **5**, 241–259.

Eugeniusz Gatnar, Dorota Rozmus

ZASTOSOWANIE LOSOWEGO DOBORU ZMIENNYCH W AGREGACJI DRZEW KLASYFIKACYJNYCH

(Streszczenie)

Drzewa klasyfikacyjne, z uwagi na swoją prostotę, elastyczność i skuteczność stają się coraz częściej wykorzystywaną metodą klasyfikacji. Mimo wielu zalet, wadą tej metody jest brak stabilności.

Poprawę stabilności i dokładności predykcji można osiągnąć poprzez agregację wielu drzew klasyfikacyjnych w jeden model. Proponowane w literaturze metody agregacji, takie jak: *bagging*, *adaptive bagging* i *arcing* opierają się na losowaniu obiektów ze zbioru uczącego; natomiast *boosting* stosuje dodatkowo system wag. W efekcie otrzymujemy zbiór drzew klasyfikacyjnych, tworzących model zagregowany.

Ponieważ losowanie obiektów może powodować zmiany rozkładu zmiennych w zbiorze uczącym, dlatego poprawę dokładności predykcji można uzyskać poprzez losowy dobór zmiennych do prób uczących, w oparciu o które powstają modele składowe agregatu.

W niniejszym artykule przedmiotem rozważań jest oszacowanie optymalnej wielkości parametrów dla procedury *RandomForest*, realizującej losowy dobór zmiennych do modelu w postaci zbioru zagregowanych drzew klasyfikacyjnych.