CS 584 Machine Learning

Final Project

A gradient approach for value weighted classification learning in naive Bayes

Chang-Hwan Lee

Instructor:	Gady Agam
Student:	Wan-Chun Wu
	Yung Chi Liu
CWID:	A20353003
	A20364639

1. Introduction

Some classification algorithms assume that all features are independent with others, which is called feature independence assumption. Moreover, those algorithms also presume that all feature are equally important which means each feature has equal weight. However, In reality the features do not play the same role in many practical application, in other words, some features are less important than others; some features are more important than others. Therefore, here is a way to improve this problem which gives each feature different weight. For example, the feature which is more important will fit with higher weight; less important feature will fit with lower weight. We call it feature weighting method. By assigning different weight, it makes the classifier more flexible.

Even though there were many feature weighting methods proposed, and most of feature weighting method improved by different way. Like Hall[7] proposed a feature weighting algorithm for naive Bayes (DTNB) using decision tree, Lee et al.[8,9] provided an information-theoretic method for compute feature weight in naive Bayes, and Gartner[6]employs SVM algorithm for feature weighting ...etc.

In this paper, the authors propose a new model of weighting method, called value weighting method. Instead of assigning each feature with different weight, they assign each feature value different weight. Therefore, the value weighting method is more a fine-grained weighting method than feature weighting. On way to improve the classification learning method is mapping the feature to higher dimensions, and compute the boundary discussion at this higher space in order to separate data. The authors propose a new way to extend the dimension of classification learning by assigning weights to feature values. Such that, it maps the feature to higher dimensions with more convolute method based on the variety of feature values.

The value weighting method has potential of being a powerful classifier and generate better performance. Base on the feature weighting method is more complex model classification learning method. We expect that the performance of value weighting method is better than any other classification include weighting method.

Since one of common classification algorithm is naive Bayesian algorithm, they choose naive Bayesian algorithm as basis algorithm. The other reason is that many data mining programs have utilized the naive Bayesian algorithm, and the results are pretty well.

2. Background

2.1. Naive Bayesian Learning

The naive Bayesian learning[1] compute the most likely class label of the new instance. Since all features are considered to be independent given the class value, the classification on d is defined as follows

$$\mathcal{V}_{NB}(d) = \underset{\mathrm{c}}{\operatorname{argmax}} \left\{ P(c) \prod_{a_{ij} \in d} P\left(a_{ij} \middle| c\right) \right\}$$

where a_{ij} represents the *j*-th value of the *i*-th feature. Naive Bayesian classifier is a linear classifier, so it is difficulties with solving non-linearly separable problems.

2.2. Feature Weighted Naive Bayesian

The naive Bayesian classification with feature weighting[2,3] is shown as follow

$$\mathcal{V}_{FWNB}(d) = \underset{\mathrm{c}}{\operatorname{argmax}} \left\{ P(c) \prod_{a_{ij} \in d} P\left(a_{ij} \middle| c\right)^{w_i} \right\}$$

where w_i represents the weight of feature. Each feature *i* has its own weight w_i. Feature weighting is a generalization of feature selection and involves larger search space than feature selection. However, feature weighting naive Bayesian classification is still a linear classifier.

2.3. Gradient descent

Gradient descent[10] is optimization algorithm, main goal is to find a local minimum of a function using gradient descent. The problem minimizing an objective function

$$Q(d) = \sum_{i=1}^{n} Q_i(w)$$

When used to minimize the function, a standard gradient descent method would perform the following iterations

$$w \leftarrow w - \eta \nabla Q(w) = w - \eta \sum_{i=1}^{n} \nabla Q_i(w)$$

And there are two gradient descenet methods, batch gradient descent and Stochastic gradient descent.

Batch gradient descent

Batch gradient descent[4] computes the gradient using all dataset, which is great for relatively smooth error manifolds. Such that, it moves somewhat directly towards an optimum solution, either local or global. Moreover, batch gradient descent given a learning rate, will eventually converge and find the minimum located in its basin of attraction.

Stochastic gradient descent

Stochastic gradient descent[5] computes the gradient using a single sample. Stochastic gradient descent works well for error that have lots of local maxima or minima. In this case, the noisier gradient calculated using the reduced number of samples   to jump out of local minima into a region is better.

3. Value Weighed Naive Bayesian

3.1. Approximation function for VWNB

This paper define several symbols and variables, and for reading formulas and algorithm clear, they are shown in Table. 3.1.

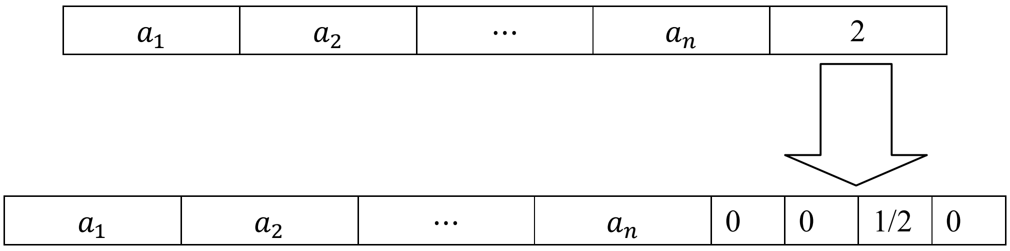
They use optimization approach to find the best weight of each feature value, and the objective function is defined as

$$\frac{1}{2}\sum_{k=1}^{N}(t_k - o_k)^2$$

where t_k represents the true label of instance *k*, and o_k represetns the predicated label of instance *k*. The objective function is divided by two because of the computational convenience. For value weighted naive Bayesian, the formula of o_k is defined as

$$o_k = \underset{\mathrm{c}}{\operatorname{argmax}}\left\{P(c)\prod_{a_{ij}\in D_k} P(a_{ij}|c)^{w_{ij}}\right\}$$

Instead of solving multi-class problems, the authors transform it into multiple binary class problem. The main reason for transform the problem is make predicated function continuous and differentiable. For implementing this, they transform class value *C* into class value set $\{c_l\}_{l=0}^{L-1}$ where *L* reperesnts the total number of class. If one instance has a class value *C* = *p*, it's class value $c_p = 1/2$ and others are zero after transformation. For example, assuming the number of class is 4 and one instance with *C* = 2. The class value set is $\left\{0, 0, \frac{1}{2}, 0\right\}$:



The reason that assigning new class value 1/2 rather than 1 is that the error function (objective function) will caculate the same error twice. And the objective function is redefined as

$$o_{kl} = \begin{cases} \frac{1}{2}, & \text{if } l = \underset{l}{\operatorname{argmax}}\left\{P(c_{kl})\prod_{a_{ij}\in D_k} P(a_{ij}|c_{kl})^{w_{ij}}\right\} \\ 0, & \text{otherwise} \end{cases}$$

Instead of continuously product, we can take the log of function

$$o_{kl} = \begin{cases} \frac{1}{2}, & \text{if } l = \underset{l}{\operatorname{argmax}} \left\{ \ln P(c_{kl}) + \sum_{a_{ij} \in D_k} w_{ij} \ln P(a_{ij} | c_{kl}) \right\} \\ 0, & \text{otherwise} \end{cases}$$

If we let

$$A_{kl} = \left(\ln P(c_{kl}) + \sum_{a_{ij} \in D_k} w_{ij} \ln P(a_{ij} | c_{kl}) \right)$$

and

$$A_{k*} = \max\{A_{kl}\}$$

Then the predicated function is rewritten as

$$o_{kl} = \begin{cases} \frac{1}{2}, & \text{if } A_{kl} = A_{k*} \\ 0, & \text{if } A_{kl} < A_{k*} \end{cases}$$

Since the VWNB is approach algorithm, we have to transform uncontinuous o_{kl} function into continuous function, like sigmoid function:

$$o_{kl} = \frac{1}{1 + e^{-(A_{kl} - A_{k*})}}$$

According the characteristic of log function:

$$\ln \mathrm{A} - \ln \mathrm{B} = \ln \frac{\mathrm{A}}{\mathrm{B}}$$

the candidcate function can be rewritten as follows

$$o_{kl} = \left(1 + exp\left(\ln \frac{P(c_{k*})}{P(c_{kl})} + \sum_{a_{ij} \in D_k} w_{ij} \ln \frac{P(a_{ij} | c_{k*})}{P(a_{ij} | c_{kl})} \right) \right)^{-1}$$

now, our value of o_{kl} is between 0 and 1/2.

3.2. Regularization

As we said before, the value weighted method is a more find-grained method, and it increases the number of parameters and makes more flexible. Assuming we have m feature in the naive Bayes method, then the number of parameters is 2m+1. And the feature weighted naive Bayes method increase the number of parameters to 2m+1+m = 3m+1. Now, let's make each feature has r different values, the number of parameters in the value-weighted naive Bayes is 2m+1+m*r = (2+r)m+1.

Even though the model is more powerful with large of parameters, they also make the model more sensitive to noise. Besides, the overfitting problem can happen with high dimensions dataset or poor training dataset. Based on these situations, the performance of VWNB isn't reliable.

To deal with this problem, they add a penalty term **W** into objective function. In this paper, they use l_2 regularization and add $\frac{\lambda}{2}||\mathbf{W}||_2^2$ as the penalty term. The final objective function is defined as follows

$$E = \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{L}(o_{kl} - t_{kl})^2 + \frac{\lambda}{2}||\mathbf{W}||_2^2$$

where N represents the total number of instances in the dataset, and L represents the total number of classes.

3.3. Gradient Descent

Instead of finding the solution directly, they use gradient descent to find the weights w_{ij} with minize the error function. The update formula is given as

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\nabla E$$

where η represents the learning rate which affects the step size in each update iteration. In the formula,

$$\nabla E = \frac{\partial E}{\partial w_{ij}} = \sum_{k}\sum_{l}(o_{kl} - t_{kl})\nabla o_{kl} + \frac{\lambda}{2}\nabla||\mathbf{W}||_2^2$$

where

$$\frac{\lambda}{2}\nabla||\mathbf{W}||_2^2 = \frac{\lambda}{2}\frac{\partial||\mathbf{W}||_2^2}{\partial w_{ij}} = \lambda w_{ij}$$

and

$$\nabla o_{kl} = \left[\frac{\partial o_{kl}}{\partial w_{11}}, \cdots, \frac{\partial o_{kl}}{\partial w_{1|a_1|}}, \frac{\partial o_{kl}}{\partial w_{21}}, \cdots, \frac{\partial o_{kl}}{\partial w_{2|a_2|}}, \cdots, \frac{\partial o_{kl}}{\partial w_{M1}}, \cdots, \frac{\partial o_{kl}}{\partial w_{M|a_M|}}\right]$$

If we let

$$\gamma_{ij} = \left(\ln\frac{P(c_{k*})}{P(c_{kl})} + \sum_{a_{ij}\in D_k} w_{ij}\ln\frac{P(a_{ij}|c_{k*})}{P(a_{ij}|c_{kl})}\right)$$

Then the o_{kl} function is follows

$$o_{kl} = \frac{1}{1 + e^{\gamma_{ij}}}$$

and the differential o_{kl} function can be rewritten as

$$\frac{\partial o_{kl}}{\partial w_{ij}} = \frac{\partial o_{kl}}{\partial \gamma_{ij}} \frac{\partial \gamma_{ij}}{\partial w_{ij}} = o_{kl}(1 - o_{kl}) \ln \left(\frac{P(a_{ij}|c_{kl})}{P(a_{ij}|c_{k*})} \right)$$

Finally, the w_{ij} update formula is given as

$$w_{ij} \leftarrow w_{ij}(1 - \eta^{\lambda}) - \eta \sum_{k} \sum_{l} (o_{kl} - t_{kl})\, o_{kl}(1 - o_{kl})\, ln \frac{P(a_{ij}|c_{kl})}{P(a_{ij}|c_{k*})}$$

According to our assignments of CS 584, learning rate is a major factor of performance. A small value of η will slow down the speed in gradient approach learning, and large learning rate causes the bigger step which may make the approach miss the best solution or local minimum. In this paper, the value η may change in eacg iteartion:

$$\eta_t = \frac{\eta_0}{1 + t/N}$$

3.4. Algorithm of VWNB

Algorithm 1 shows the detail of the algorithm of VWNB. In each iteration, first, we compute the probability of each class value of instance c_{kl} and the probability of each value of each feature a_{ij} given c_{kl}. Second, we calculate the predicted class values of each instance *k*. Next, updating the values of w_{ij} with the results of $\{P(c_{kl})\}$, $\{P(a_{ij}|c_{kl})\}$, and $\{o_{kl}\}$ from previous two steps.

However, we found out that all training instances have the same probability of each class and the same probability of *j*-th value of *i*-th feature given c_{kl}. Therefore, we modified the algorithm by extracting this part out of the first for-loop. And it is confused that for each instance D_k, the program will update every value of w_{ij} since it's impossible that one instance has every value of *i*-th feature. Hence, we rewrite the step that for each w_{ij}, we calculate every o_{kl} with a_{ij}. The new algorithm is shown in Algorithm 2.

Algorithm 1. Value weighted naive Bayesian.

```
Input: N: total number of records, $T$: training data, $S$: test data, $C$: target feature, $\eta$: learning rate, $\lambda$: regularization factor
initialize $w_{ij}$
initialize $\eta$ and $\lambda$
**while** $w_{ij}$ changes **do**
    **for** each training data $D_k \in T$ **do**
        **for** each feature value $j$ of each feature $i$ **do**
            **for** each class value $l$ **do**
                calculate $P(C_{kl})$ and $P\left(a_{ij}\middle|c_{kl}\right)$
            **end for**
            calculate $P(C_{k*})$ and $P\left(a_{ij}\middle|c_{k*}\right)$
            **for** each class value $l$ **do**
                calculate $o_{kl} = 1 + exp\left(ln\frac{P(C_{k*})}{P(C_{kl})} + \sum_{a_{ij}\in D_k} w_{ij}\, ln\frac{P\left(a_{ij}\middle|c_{k*}\right)}{P\left(a_{ij}\middle|c_{kl}\right)}\right)^{-1}$
            **end for**
            update $w_{ij} \leftarrow w_{ij}\left(1-\eta^{\lambda}\right) - \eta\sum_k\sum_l (o_{kl}-t_{kl})\, o_{kl}(1-o_{kl})\, ln\frac{P\left(a_{ij}\middle|c_{kl}\right)}{P\left(a_{ij}\middle|c_{k*}\right)}$
        **end for**
    **end for**
    update $\eta_{new} \leftarrow \frac{\eta_{old}}{1+t/N}$
**end while**
**for** each test data $d \in S$ **do**
    class value od $d = \mathrm{argmax}_{c\in C}\, P(C)\prod_{a_{ij}\in d} P\left(a_{ij}\middle|c\right)^{w_{ij}}$
**end for**
```

Algorithm 2. Imporved value weighted naive Bayesian.

```
Input: $N$: total number of records, $T$: training data, $S$: test data, $C$: target feature, $\eta$: learning rate, $\lambda$: regularization factor
initialize $w_{ij}$
initialize $\eta$ and $\lambda$
for each training data $D_k \in T$ do
  for each class value $l$ do
    calculate $P(C_{kl})$ and $P(a_{ij}|C_{kl})$
  end for
  calculate $P(C_{k*})$ and $P(a_{ij}|C_{k*})$
while $w_{ij}$ changes do
  for each feature value $j$ of each feature $i$ do
    for each training data $D_k \in T$ do
      for each class value $l$ do
        calculate $o_{kl} = 1 + exp\left(ln\frac{P(C_{k*})}{P(C_{kl})} + \sum_{a_{ij} \in D_k} ln\frac{P(a_{ij}|C_{k*})}{P(a_{ij}|C_{kl})}\right)^{-1}$
      end for
    end for
    update $w_{ij} \leftarrow w_{ij}(1 - \eta^{\lambda}) - \eta \sum_k \sum_l (o_{kl} - t_{kl})\, o_{kl}(1 - o_{kl})\, ln\frac{P(a_{ij}|C_{kl})}{P(a_{ij}|C_{k*})}$
  end for
  update $\eta_{new} \leftarrow \frac{\eta_{old}}{1+t/N}$
end while
for each test data $d \in S$ do
  class value od $d = \text{argmax}_{c \in C} P(C) \prod_{a_{ij} \in d} P(a_{ij}|c)^{w_{ij}}$
end for
```

4. Evaluation

All of the experiments are tested on one private laptop. The enviroment is:

- Intel(R) Core(TM) i5-4200H CPU @2.8GHz 2.79Hz
- 8G RAM
- Windows 10 x64 family
- Eclipse Mars (4.50)
- Python 2.7

We used N-fold cross-validation[11] (where N in our evaluation and the paper is 10), initial weights are given as 1, and regularization factor λ is given as 2. The learning rate η in each dataset are different since the range of each feature are different. Like this paper, we compared the performance with naive Bayes[1], Classification, and Regression Trees (CART)[12], and Random Forst (RF)[13].

Table 4.1 shows the results in this paper, and Table 4.2 shows our results. We implement CART and RF method by importing the Python library. According the first table, VWNM doesn't always perform the best result, but it's performance is better than NB in most of dataset. However, according to our evaluation, VWNB only give the best performance in balance dataset. The main reason is that the performances of CART and RF which we implemented are better than their results.

According both tables, VWNB plays better performance than NB. But, the execution time of VWNB is much longer than NB. Thus, we have to make a choice between higher accuracy rate or faster execution time.

Table 4.1. The results from the paper.

Method / Dataset	VWNB	NB	CART	RF
balance	74.1 ± 5.3	70.7 ± 4.2	70.0 ± 4.0	70.5 ± 4.0
derma	98.6 ± 1.9	98.0 ± 2.2	94.9 ± 3.4	96.3 ± 3.4
glass	76.0 ± 8.6	74.3 ± 7.9	76.6 ± 8.7	76.6 ± 9.4
iris	94.4 ± 6.2	94.0 ± 5.6	94.0 ± 7.8	94.6 ± 5.6
spect	74.0 ± 14.4	75.0 ± 14.3	67.5 ± 13.4	70.0 ± 15.7

Table 4.2. The testing results of different dataset.

Method / Dataset	VWNB	NB	CART	RF
balance	88.70 ± 4.87	91.94 ± 6.45	76.50 ± 8.61	82.24 ± 7.84
derma	51.42 ± 15.71	51.43 ±14.83	48.61 ± 13.88	52.78 ± 9.72
glass	52.38 ± 16.67	47.62 ± 21.43	70.17 ± 14.28	76.71 ± 18.18
iris	93.33 ±10.00	93.34 ± 10.00	95.33 ± 6.667	96.00 ± 6.67
spect	80.77 ±23.07	80.76 ± 21.15	74.55 ± 9.90	81.62 ± 7.90

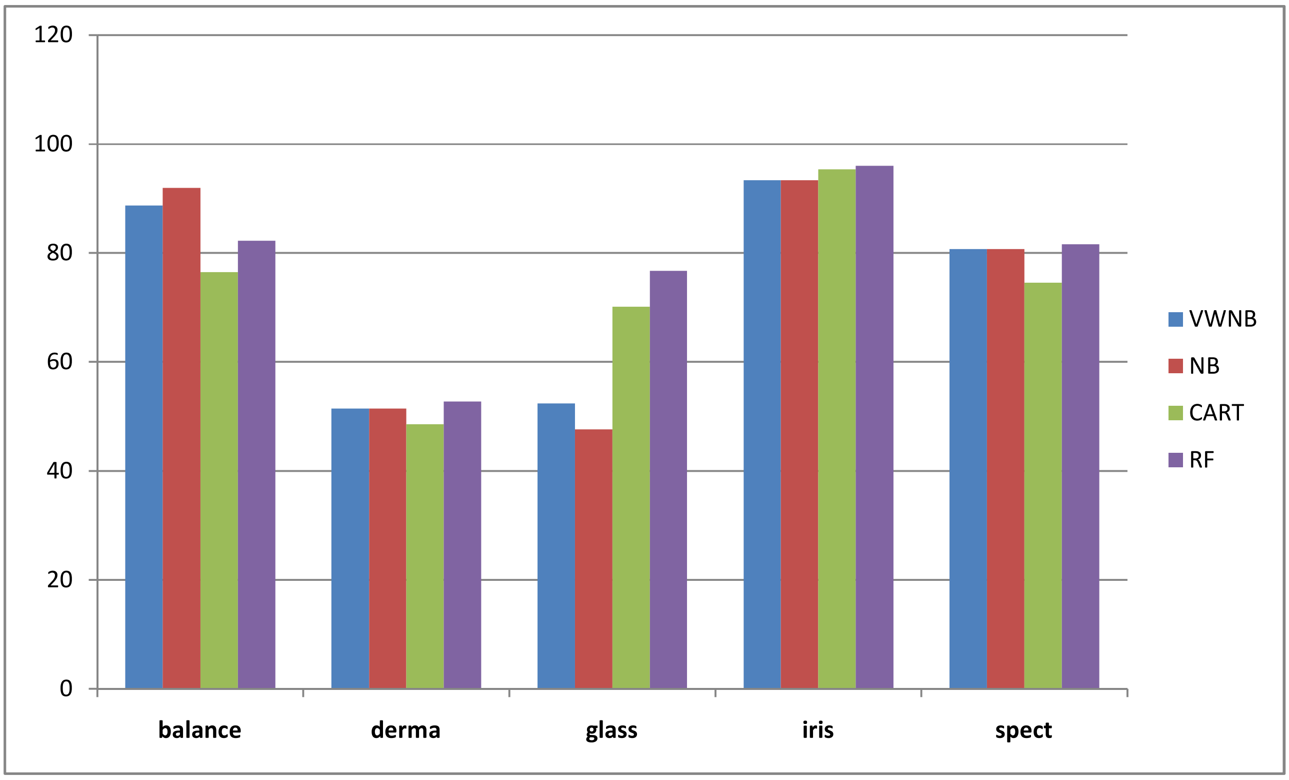


Figure 4.1. The average accuracy rate in different dataset.

5. Conclusion

In this paper, the authors proposed a new paradigm of weighting method which is called value-weighting method. Value weighting method is a find-grained weighted methods, and it makes higher dimensions parameters by assigning each value of feature different weights. And they implement the method with naive Bayes. The results show that VWNB perform significantly well.

According to our implement, VWNB does improve the accuracy rate compared with NB, but the result is not pretty as this paper shows. And, our results show that. CART[12] and RF[13] in Python are better than this paper shows.

As our future work, we can keep modifying the algorithm. We guess that we may don't have to compute the conditional class value of each instance for each value of a feature. Based on this modification, the execution can be at least 1.5 times faster than before. Second, since we only update one weight at one iteration, we can utilize the parallel programming. By using mutiply threads, we can implement updating several weights of a_{ij} different at same time. Another way to improve the performance is combine VWNB with different classification learning method. According the result tables, VWNB doesn't give the best accuracy evertime. Since Python library show that the perfromance is better than before, utilizing the Python function is good way to improve VWNB

Reference

[1] Lewis, David D. "Naive (Bayes) at forty: The independence assumption in information retrieval." Machine learning: ECML-98. Springer Berlin Heidelberg, 1998. 4-15.

[2] Chang-Hwan Lee, An information-theoretic filter method for feature weighting in naive bayes, Int. J. Pattern Recogn. Artif. Intell. 28 (5) (2014).

[3] Chang-Hwan Lee, Fernando Gutierrez, Dejing Dou, Calculating feature weights in naive bayes with kullback-leibler measure, in: 11th IEEE International Conference on Data Mining, 2011.

[4] Wilson, D. Randall, and Tony R. Martinez. "The general inefficiency of batch training for gradient descent learning." Neural Networks 16.10 (2003): 1429-1451.

[5] Recht, Benjamin, et al. "Hogwild: A lock-free approach to parallelizing stochastic gradient descent." Advances in Neural Information Processing Systems. 2011.

[6] Thomas Gärtner, Peter A. Flach, Wbcsvm: weighted bayesian classification based on support vector machines, in: The Eighteenth International Conference on Machine Learning, 2001.

[7] Mark Hall, A decision tree-based attribute weighting filter for naive bayes, Knowl.-Based Syst. 20 (2) (2007).

[8] C hang-Hwan Lee, An information-theoretic filter method for feature weighting in naive bayes, Int. J. Pattern Recogn. Artif. Intell. 28 (5) (2014).

[9] Chang-Hwan Lee, Fernando Gutierrez, Dejing Dou, Calculating feature weights in naive bayes with kullback-leibler measure, in: 11th IEEE International Conference on Data Mining, 2011.

[10] Burges, Chris, et al. "Learning to rank using gradient descent." Proceedings of the 22nd international conference on Machine learning. ACM, 2005.

[11] Refaeilzadeh, Payam, Lei Tang, and Huan Liu. "Cross-validation."Encyclopedia of database systems. Springer US, 2009. 532-538.

[12] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth and Brooks, Monterey, CA, 1984.

[13] Leo Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.