applied sciences

Article

On the Optimal Size of Candidate Feature Set in Random forest

Sunwoo Han and Hyunjoong Kim *

Department of Applied Statistics, Yonsei University, Seoul 03722, Korea; sunwoo.han@yonsei.ac.kr
* Correspondence: hkim@yonsei.ac.kr; Tel.: +82-2-2123-2545

Received: 27 December 2018; Accepted: 26 February 2019 ; Published: 3 March 2019

Abstract: Random forest is an ensemble method that combines many decision trees. Each level of trees is determined by an optimal rule among a candidate feature set. The candidate feature set is a random subset of all features, and is different at each level of trees. In this article, we investigated whether the accuracy of Random forest is affected by the size of the candidate feature set. We found that the optimal size differs from data to data without any specific pattern. To estimate the optimal size of feature set, we proposed a novel algorithm which uses the out-of-bag error and the 'SearchSize' exploration. The proposed method is significantly faster than the standard grid search method while giving almost the same accuracy. Finally, we demonstrated that the accuracy of Random forest using the proposed algorithm has increased significantly compared to using a typical size of feature set.

Keywords: classification; ensemble; random forest; decision tree; feature set; out-of-bag

1. Introduction

Classification is a predictive modeling whose target variable is categorical. There are several classical classification techniques such as logistic regressions, linear discriminant analysis and K-nearest neighbors. Modern popular methods include generalized additive models, decision trees, support vector machines and neural networks. A classification method that predicts the target variable with high accuracy is preferred. It is known that the best classifier for a dataset is domain dependent [1].

Among many classifiers, a decision tree is a popular choice because the tree output is easy to understand and resembles more closely to human decision-making processes [2]. Moreover, it can be displayed graphically and can handle qualitative predictors without difficulty [2,3]. However, a decision tree generally does not have the same level of predictive accuracy compared to many modern classification methods. Hence, ensemble methods using decision trees were actively proposed to improve the classification accuracy of a single tree classifier [4].

Ensemble method is to combine many individual models to achieve better prediction accuracy for classification or regression. Popular ensemble methods for classification are to combine many weak classifiers, usually trees, to reach a final prediction [4]. Of course, ensemble methods can be applied using other classifiers such as neural networks [5].

The well-known methodologies of classification ensemble are Boosting [6–8], Bagging [9] and Random forest [10]. Random forest, a variant version of bagging, is a tree-based ensemble method. As in bagging, Random forest uses bootstrapped samples at each stage of ensemble. However, it uses different subset of candidate features at each node when trees are constructed. The candidate feature set is a random subset of m predictors from the full set of p predictors. The value m is one of the parameters that a user can control. Several researches studied the effect of parameter values on the performance of Random forest. Huang et al. [11] investigated the parameter sensitivity of Random forest. Boulesteix et al. [12] described a search method on the selection of parameters. Freeman et al. [13] explored the sensitivity of Random forest and stochastic gradient boosting to choices in tuning parameters.

In this article, we will investigate the effect of candidate feature size m on the performance of Random forest. Furthermore, we will develop a novel algorithm to estimate a value m from the training dataset. We will demonstrate that the chosen value m by the proposed algorithm outperforms the default setting in terms of test accuracy.

In Section 2, we will describe Random forest and provide a motivation for this research. In addition, the result of an extensive experiment using 58 datasets to figure out whether the size of candidate feature set affects the performance of Random forest will be reported. A novel algorithm for estimating an adequate size of candidate feature set will be proposed in Section 3. In Section 4, we will compare the performance of Random forest; using the estimated size of candidate feature set, and using the default size of candidate feature set. We will provide a conclusion in Section 5.

2. Motivation

2.1. Random Forest

Given a training dataset, Random forest uses bootstrapped samples at each stage of ensemble. The main difference from Bagging is that it uses different random subset of m features (predictors) instead of using all p features at each node of trees. The best partitioning rules for nodes are selected only from the candidate feature set. Using only a subset of features rather than all, Random forest generally produces many diverse and un-correlated trees. Averaging many un-correlated trees has resulted in a significant reduction in classification variance, thus leads to boost the classification accuracy [9]. It has been shown that the predictive accuracy of Random forest is significantly better on average than that of Bagging [14]. Random forest is a popular method in various fields because it is fast both in training and prediction, robust to label noise and outliers, has an inherent multi-class capability, suitable for parallel processing, and has good performance for high dimensional data. Algorithm 1 provides the algorithm of Random forest. For more details, refer to Breiman [10].

Algorithm 1: The algorithm of Random forest

```
Training Phase:
Given:
- $D$: training set with $n$ instances, $p$ variables and target variable.
- $K$: number of classes of target variable.
- $B$: number of classifiers in Random forest.
Procedure:
For $b$ = 1 to $B$
1. Generate bootstrapped sample $D_b^*$ from the training set $D$.
2. Grow a tree using a candidate feature set from bootstrapped sample $D_b^*$.
   For a given node $t$,
   (i) randomly select $m$ features. Typical choices are $m \approx \sqrt{p}$ in classification or $m \approx p/3$
  in regression.
   (ii) find the best split features and values using the candidate feature set.
   (iii) split a node using the best split features and values.
   Repeat (i)–(iii) until stopping rules are met.
3. Construct trained classifiers $C_b$.
Test Phase:
Aggregate the $B$ trained classifiers using simple majority vote.
For a test instance $x$, the predicted class label from classifiers $C_b$ is:
              $C_B(x) = \text{argmax}_j \sum_{b=1}^{B} I(C_b(x) = j), \text{for } j = 1, \ldots, K$
```

2.2. Effect of the Size of Candidate Feature Set

Breiman [10] proposed the size of candidate feature set at each node as $m \approx log_2(p+1)$ in the original article. Later, many researches related to Random forest have generally used the default size of candidate feature set as $m \approx \sqrt{p}$ in classification problems and $m \approx p/3$ in regression problems.

In addition, the default value of *mtry* option which controls the size of candidate feature set in R program package named *randomForest* is also $m = floor(\sqrt{p})$ in classification and $m = floor(p/3)$ in regression. In general, it is empirically known that good performances of Random forest are obtained using those default values. However, there is no existing specific mathematical proofs or theories.

Meanwhile, bagging is a special case of Random forest where the size of candidate feature set is $m = p$. In other words, bagging uses the full feature set at each node of trees. In addition, Random forest generally outperforms bagging in terms of test accuracy [14]. Therefore, we can presume that the performance of Random forest can be affected by the size of candidate feature set. To investigate it, we conducted an experiment using 4 real or artificial datasets. Table 1 provides the description of datasets.

Table 1. The description of 4 datasets.

Datasets	Observations	Classes	Variables	Sources
ail	13,750	2	12	[15]
cmc	1473	3	9	UCI (Contraceptive method choice)
int	1000	2	9	[16]
rng	1000	2	10	R library *mlbench* (Ringnorm)

The design of the experiment is as follows. We used random 60% of the dataset as training set for fitting, and random 40% of the dataset as test set for evaluating. With the training data, a Random forest model is constructed using the fixed feature set size m. We explored all possible values of m, such as $m = 1, 2, \ldots, p$. Finally, we calculated the classification accuracy of each Random forest model using the test data.

As in Breiman [10], we set the number of trees in Random forest to 100, but it is not limited in general. Other parameter values in Random forest package were set to the default ones. For a reliable comparison, we repeated the whole experiment process 100 times to reduce the effect of favorable training or test datasets. The repeated process will also reduce the sampling bias due to unbalanced class frequencies.

The result of the experiment is summarized in Figure 1. The horizontal axis refers to the size of candidate feature set and the vertical axis refers to the test accuracy. In addition, points in the figure represents the average of the test accuracy obtained from 100 repetitions. The filled point indicates the optimal size of candidate feature set which shows the highest test accuracy and the crossed point indicates the default size of candidate feature set.

As a result, firstly, we found that the performance of Random forest was highly affected by the size of candidate feature set. Secondly, the default size of candidate feature set did not assure the best performance of Random forest. In addition, the optimal size of candidate feature set is quite different for each data.

2.3. Further Motivational Experiment

In this section, we expand the experiment in Section 2.2 further to observe any specific pattern on the optimal size of candidate feature set. The experiment was conducted on 58 real or artificial datasets that are suitable for classification problem, which mostly came from UCI Data Repository [17] and R package named *mlbench* [18]. The datasets are summarized in Table 2. The design of the experiment is the same as in Section 2.2.

The result of the experiment is summarized in Figure 2. The horizontal axis refers to 58 datasets and the vertical axis refers the difference between the optimal m and the default m. Here, the optimal m means the size of feature set which showed the highest test accuracy. The datasets are arranged in the order of the difference between the optimal m and the default m.

Table 2. The description of 58 datasets.

Datasets	Observations	Classes	Variables	Sources
aba	4177	2	8	UCI (Abalone)
ail	13,750	2	12	[15]
aus	690	2	14	UCI (Australian credit approval)
bal	625	3	4	UCI (Balance scale)
ban	1372	2	4	UCI (Bank note authentication)
bcw	683	2	9	[1]
bld	345	2	6	UCI (BUPA liver disorders)
blo	748	2	4	UCI (Blood transfusion center)
bod	507	2	24	[19]
bos	506	3	13	UCI (Boston housing)
bre	699	2	9	UCI (Wisconsin Breast Cancer)
cir	1000	2	10	R library *mlbench* (Circle in a square)
cmc	1473	3	9	UCI (Contraceptive method choice)
col	366	3	23	UCI (Horse Colic)
cre	690	2	15	UCI (Credit approval)
cyl	540	2	30	UCI (Cylinder bands)
der	358	6	34	UCI (Dermatology)
dia	768	2	8	[15]
dna	3186	3	60	UCI (StatLog DNA)
ech	131	2	6	UCI (Echocardiogram)
eco	336	4	7	[15]
fis	159	7	6	[20]
ger	1000	2	20	UCI (German credit)
gla	214	6	9	UCI (Glass)
hea	270	2	13	UCI (StatLog heart disease)
hep	155	2	19	UCI (Hepatitis)
hil	606	2	100	UCI (Hill-valley)
imp	205	6	23	UCI (Auto imports)
int	1000	2	9	[16]
ion	351	2	33	UCI (Ionosphere)
iri	150	3	4	UCI (Iris)
lak	259	6	14	[15]
led	6000	10	7	UCI (LED display)
lib	360	15	90	UCI (Libras movement)
mam	961	2	5	UCI (Mammographic mass)
mar	8777	10	4	[15]
pid	532	2	7	UCI (PIMA Indian diabetes)
pks	195	2	22	UCI (Parkinsons)
pov	97	6	6	[21]
rng	1000	2	10	R library *mlbench* (Ringnorm)
sat	6435	6	36	UCI (StatLog satellite image)
sea	3000	3	7	[22]
seg	2310	7	18	UCI (Image segmentation)
smo	2855	3	8	UCI (Attitude towards smoking restrictions)
snr	208	2	60	R library *mlbench* (Sonar)
soy	307	7	28	[15]
spa	4601	2	57	UCI (Spambase)
spe	267	2	44	UCI (SPECTF heart)
thy	7200	3	21	UCI (Thyroid disease)
trn	1000	2	10	R library *mlbench* (Threenorm)
twn	1000	2	10	R library *mlbench* (Twonorm)
usn	1302	3	27	[23]
veh	846	4	18	UCI (StatLog vehicle silhouette)
vol	1521	6	5	[15]
vot	435	2	16	UCI (Congressional voting records)
vow	990	11	10	UCI (Vowel recognition)
wav	3600	3	21	UCI (Waveform)
zoo	101	7	16	R library *mlbench* (Zoo)

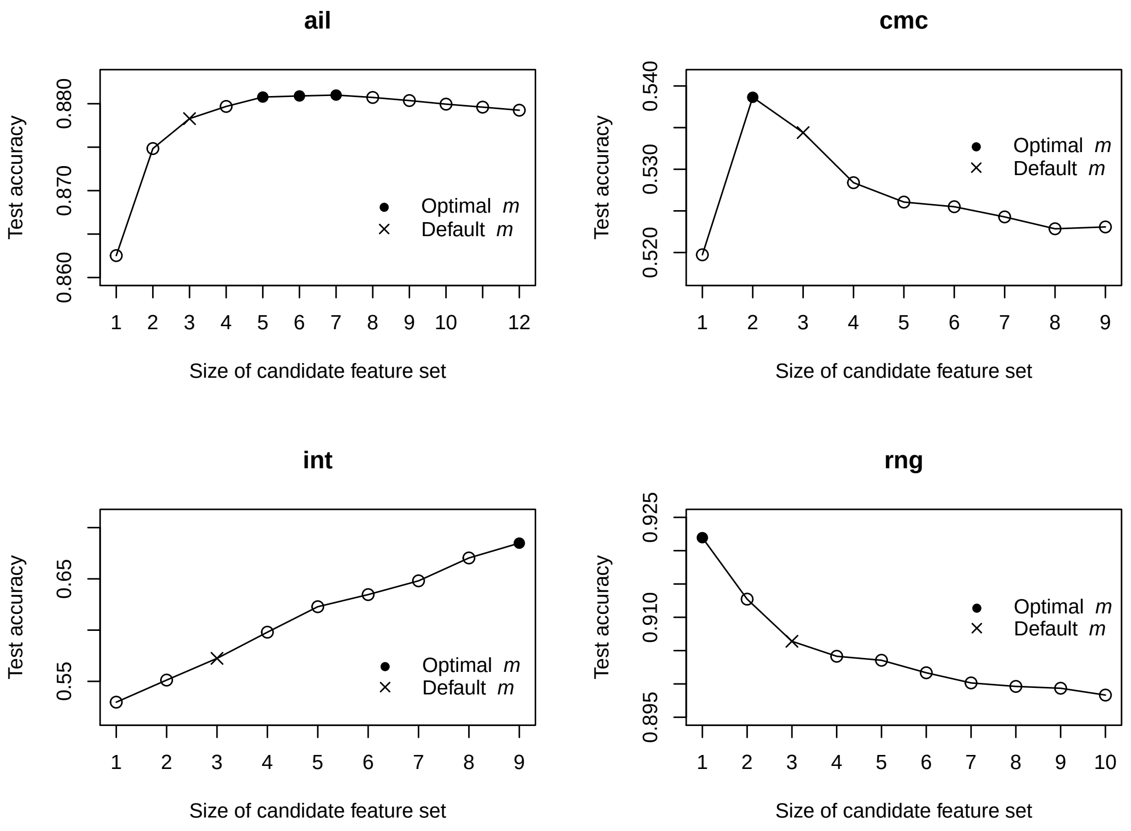

Figure 1. Accuracies for each size of candidate feature set.

As a result, we found that the optimal m was different from the default m in most datasets. Specifically, among 58 datasets, there are 27 datasets where the optimal m was smaller than the default m. In 20 datasets, the optimal m was larger than the default m. There are only 11 datasets where they were the same.

Figure 3 shows the confidence interval plots of the test accuracy differences using two values of m. The horizontal axis indicates 58 datasets with the same order of Figure 2. The vertical axis indicates the test accuracy differences. It is shown that 31 confidence intervals are located above 0. This means that the accuracy using the optimal m is statistically better than using the default m in 31 out of 58 datasets. There also is a little tendency that the accuracy difference becomes more significant as the difference of two sizes are large, especially when larger than 3. This suggests that a gain in accuracy might be achieved if the value m is chosen more accurately.

Is there a functional relationship between m and p? We tried to find any-statistically significant-functional relationships between the optimal m and the characteristic of a dataset including p. In fact, we considered more variables regarding the characteristic of a dataset including p, p/n, stump error, gini impurity [24], depth of a single tree, majority rule error, and the number of classes. Although not reported here, after many analyses, we concluded that there is no specific functional relationship between the optimal m and the characteristics of each dataset. That is, the optimal m is domain dependent, and needs to be found each time we run Random forest on a dataset. This motivates us to develop an algorithm to estimate the optimal m which is unique to each dataset.

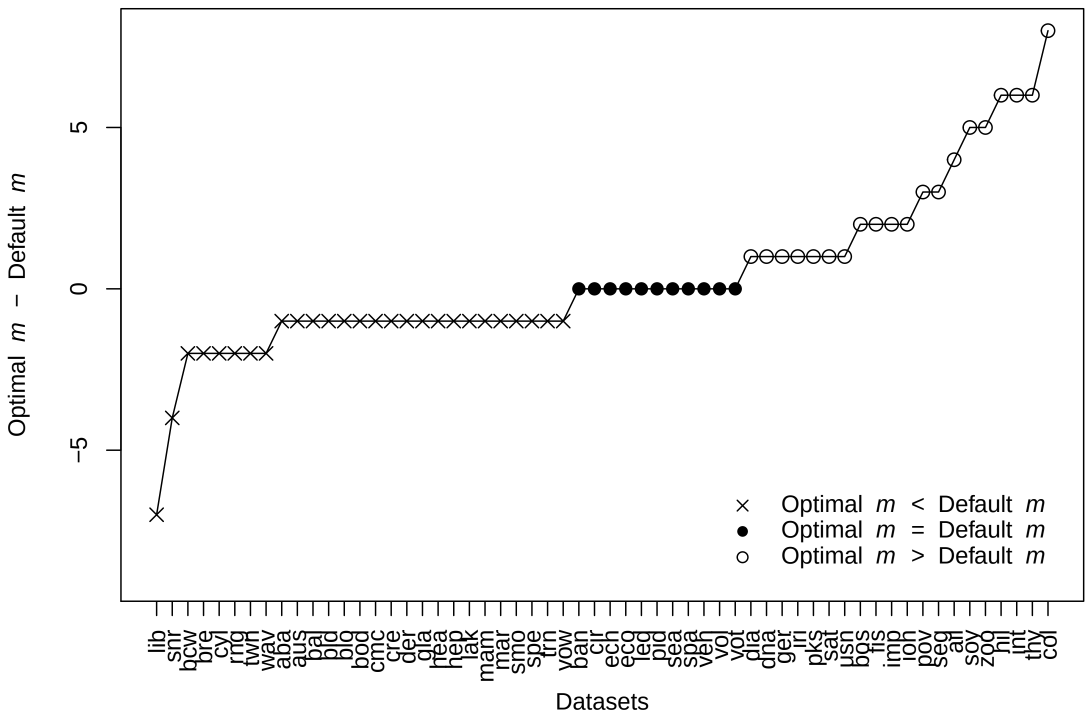

Figure 2. The difference between the optimal size and the default size.

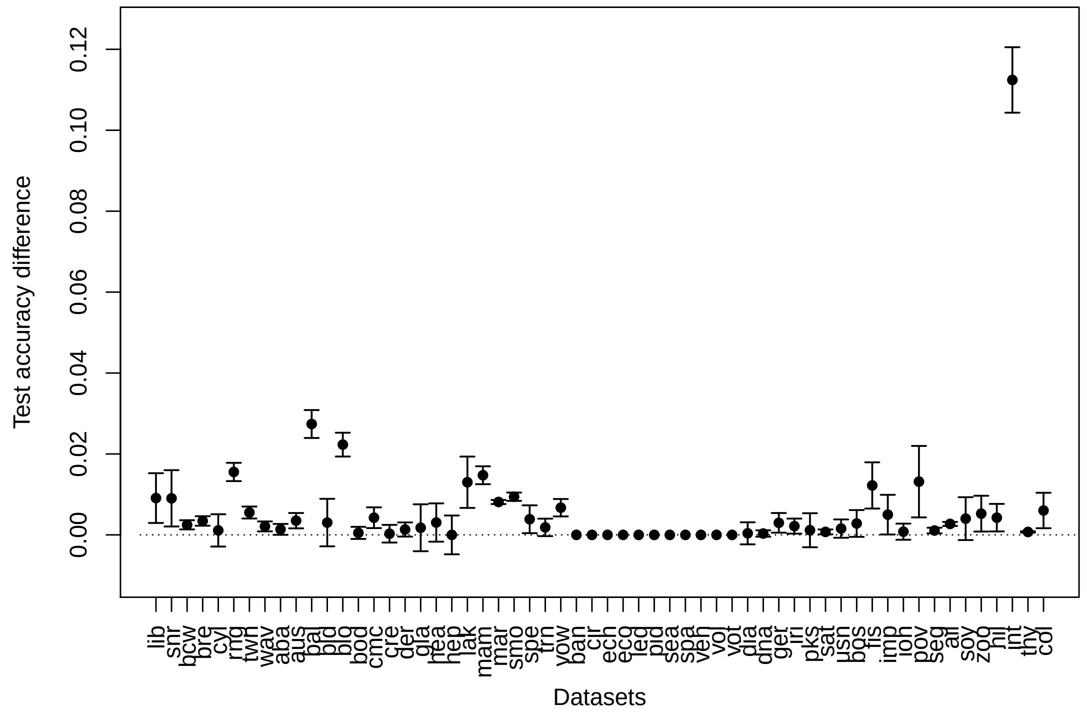

Figure 3. Confidence intervals of accuracy differences between the optimal m and the default m. The difference is significant when the confidence interval does not contain zero.

3. Proposed Algorithm

The beginning idea of our proposed algorithm is to use the out-of-bag (OOB) error as in Boulesteix et al. [12]. Recall in Algorithm 1 that the bootstrapped sample of the training data is used in each tree construction. The sample which are not selected in the bootstrapping is defined as out-of-bag (OOB) sample. The OOB error is the error rate calculated on the OOB sample [14]. Since the

OOB sample is not used in the training stage, the OOB error is a good estimate of the true performance of ensemble methods. Similarly, an estimated size of candidate feature set using the OOB error can provide a good estimate for the optimal m.

The main characteristics of the proposed algorithm are as follows.

Firstly, the algorithm compares the OOB errors starting from the smallest size of candidate feature set, which is one, step by step. The rationale is that there are more datasets in which the optimal m is smaller than the default m as shown in Figure 2. Another advantage is that as the value m gets smaller, training Random forest gets faster. Therefore, a simple and fast Random forest model can be produced.

Secondly, since the bootstrapping has a randomness, the algorithm uses a statistical tolerance limit to compare the OOB errors. The definition of the statistical tolerance limit we used is $e^* = \hat{e} - 0.5\sqrt{\frac{\hat{e}(1-\hat{e})}{n}}$ where $\hat{e}$ is the OOB error and n is the OOB sample size. The second characteristic means that if the OOB error of current size of candidate feature set ($\hat{e}_{current}$) is less than the statistical tolerance limit of the previous size of candidate feature set (${e^*}_{previous}$), the algorithm moves on to the current size and continues to compare with the next size. For this reason, the algorithm prefers the smaller size of feature set if several sizes are in the same statistical significance range.

Finally, the algorithm uses the concept of *SearchSize* to control the number of searches, where *SearchSize* is defined as $ceiling(p/10)$. Note that the number of searches to be compared is different depending on the number of predictors of datasets. For example, consider a dataset with total of $p = 41$ variables. In this case, $SearchSize = ceiling(41/10) = 5$. Let e_1^* be the tolerance limit at $m = 1$. Let's now consider $SearchSize = 5$ as the number of contending sizes, i.e. $m = 2, 3, 4, 5, 6$. Let the number of features with the smallest OOB error be $m = 4$. Assume that the OOB error at $m = 4$ is less than e_1^*. Then $m = 4$ becomes the provisional optimal value. Let e_4^* be the tolerance limit when $m = 4$. We again consider another five ($SearchSize = 5$) OOB errors beyond $m = 4$. That is, $m = 5, 6, 7, 8, 9$ become new contending candidates. Let the smallest OOB error be $m = 8$ among the candidates. However, if the OOB error at $m = 8$ is not less than e_4^*, we will still choose $m = 4$ and stop the search. If the OOB error at $m = 8$ is less than e_4^*, the new tentative optimal value will be $m = 8$. Then we will search for additional five candidates, $m = 9, 10, 11, 12, 13$.

The algorithm is shown in Algorithm 2. The prediction performance of the proposed algorithm will be evaluated in the next section.

Algorithm 2: Algorithm to estimate the optimal size m.

```
Require: $F \leftarrow 1$, $SearchSize \leftarrow ceiling(p/10)$
Require: $\hat{e}_f$: out-of-bag error with $m = f$
Require: $e_f^*$: statistical tolerance limit with $m = f$
Procedure:
1: Compute $\hat{e}_1$ and $e_1^*$
2: for $x$ from $(F+1)$ to $(F + SearchSize)$ with increment 1
3:     if there is $\hat{e}_x$ already then skip
4:     else compute $\hat{e}_x$
5:     end if
6: end for
7: $j \leftarrow argmin_j(\hat{e}_j \mid j \in [F+1, \ldots, F + SearchSize])$
8: Compute $e_j^*$
9: if $\hat{e}_j < e_F^*$ then $F \leftarrow j$ and go to 2
10: Return $F$
```

4. Experimental Study

4.1. Comparison between the Optimal and the Estimated Values

In this section, we conducted an experiment to see the performance of Random forest using the estimated m by the proposed algorithm. We used the same datasets in Section 2.3 with the same experiment design.

First, we compare the optimal m with the estimated m using the proposed algorithm. Figure 4 presents the difference between the optimal m and the estimated m. The horizontal axis indicates 58 datasets with the same order of Figure 2.

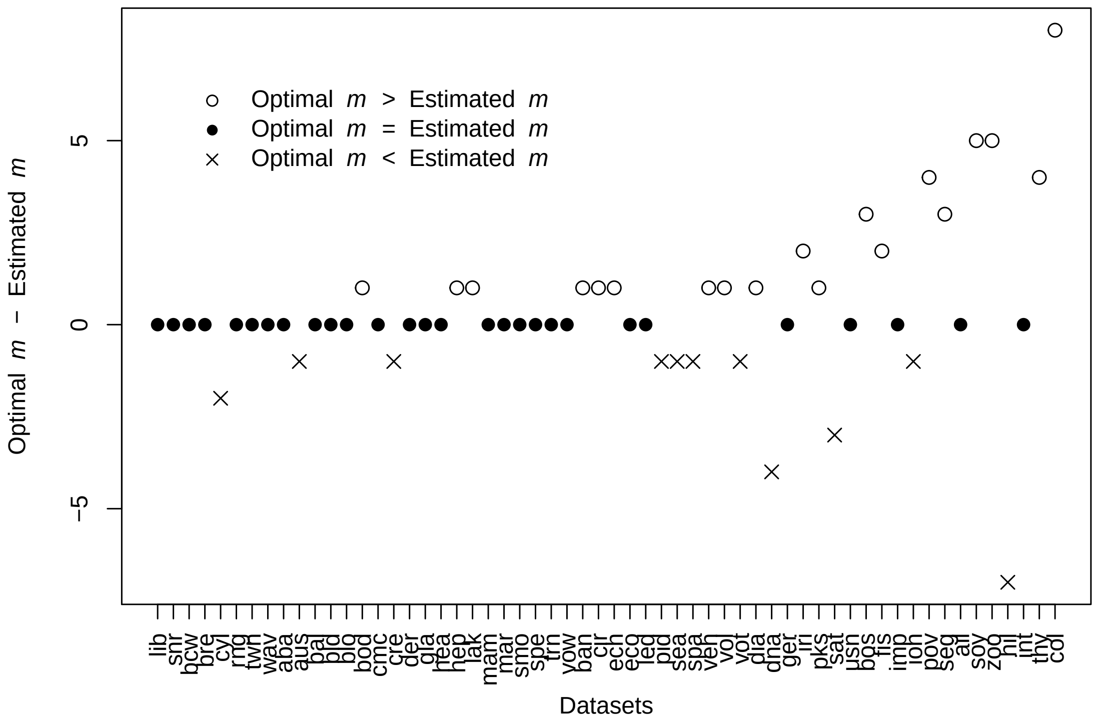

Figure 4. The difference between the optimal m and the estimated m.

By comparing Figures 2 and 4, we notice that the estimated m is much closer to the optimal m. Since the optimal m may not be unique and other values can also give almost the same accuracies, it is interesting to see if the estimated m shows similar accuracy with the optimal m in general. Figure 5 presents the test accuracy differences between the optimal m and the estimated m. According to Figure 5, 45 confidence intervals contain zero. That is, the estimated m and the optimal m produced statistically similar accuracies mostly. Only 13 datasets among 58 showed statistically significant differences. Recall that 31 datasets showed significant differences in accuracy in Figure 3.

Therefore, we can conclude that the proposed algorithm can estimate the candidate feature set reasonably well because (1) the estimated m is close to the optimal m frequently and (2) the estimated m gives similar accuracy with the optimal m even if they are not exactly same.

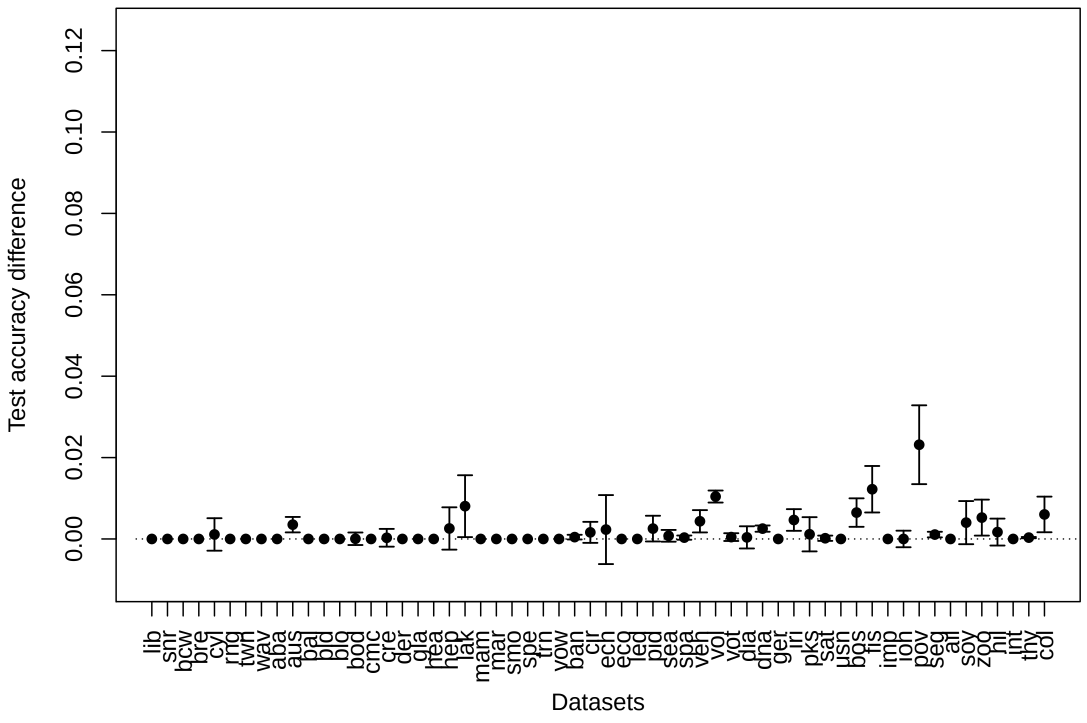

Figure 5. Confidence intervals of accuracy differences between the optimal m and the estimated m. The difference is significant when the confidence interval does not contain zero.

4.2. Comparison between the Estimated and the Default Values

According to Figure 2, there are 9 datasets in which the differences between the optimal m and the default m are larger than 3. Does the estimated value of m improve the accuracy of Random forest on these 9 outlier datasets? In Figure 6, we compared the performances between the two values. As shown in the left panel, the accuracy of the default m is statistically worse in 7 datasets among 9, compared to the optimal m. Contrarily in the right panel, the accuracy of the estimated m is statistically worse in only 2 datasets. In summary, estimating the size of m nicely can provide significant improvements in more outlying cases (e.g., $|$ optimal m - default m $| > 3$).

Now, we compare the accuracy of the estimated m and the default m directly. Table 3 shows the summary of comparison between the estimated m and the default m in terms of test accuracy. The values on the table represent the number of times that the vertical method was more accurate than the horizontal method. Therefore, the value 32 in the first row and second column means that using the estimated m was more accurate than using default m in 32 datasets among 58. On the other hand, there were 14 datasets where using default m performed better than using estimated m. In remaining 12 datasets, the results were the same. In addition, the values inside the parenthesis represent the frequency that these differences are statistically significant from the paired t-test. The estimated m from our proposed algorithm significantly performed better than using the default m in 21 datasets. There were only 5 datasets on the opposite side.

Figure 7 shows the confidence interval plots of the test accuracy differences between two methods. Twenty one confidence intervals were located above 0. There were only 5 intervals which were located below 0.

Table 3. Summary of test accuracy comparison.

		Win	
		Default size	Estimated size
Loss	Default size		32 (21)
	Estimated size	14 (5)	

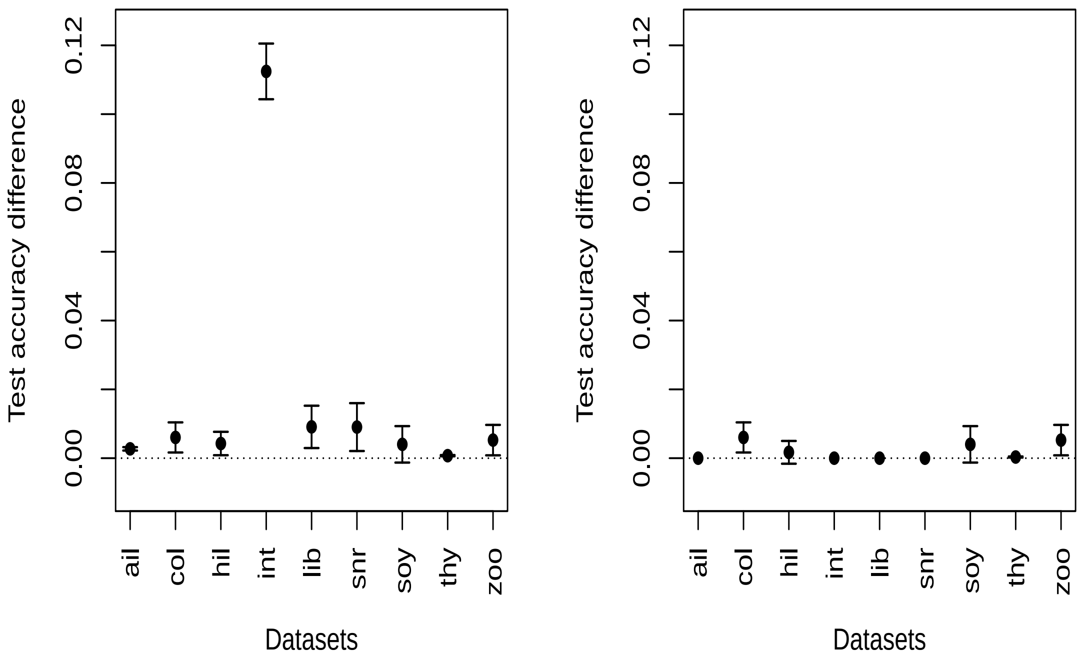

Figure 6. Confidence intervals of accuracy differences between the optimal m and the default m (**left**); between the optimal m and the estimated m (**right**).

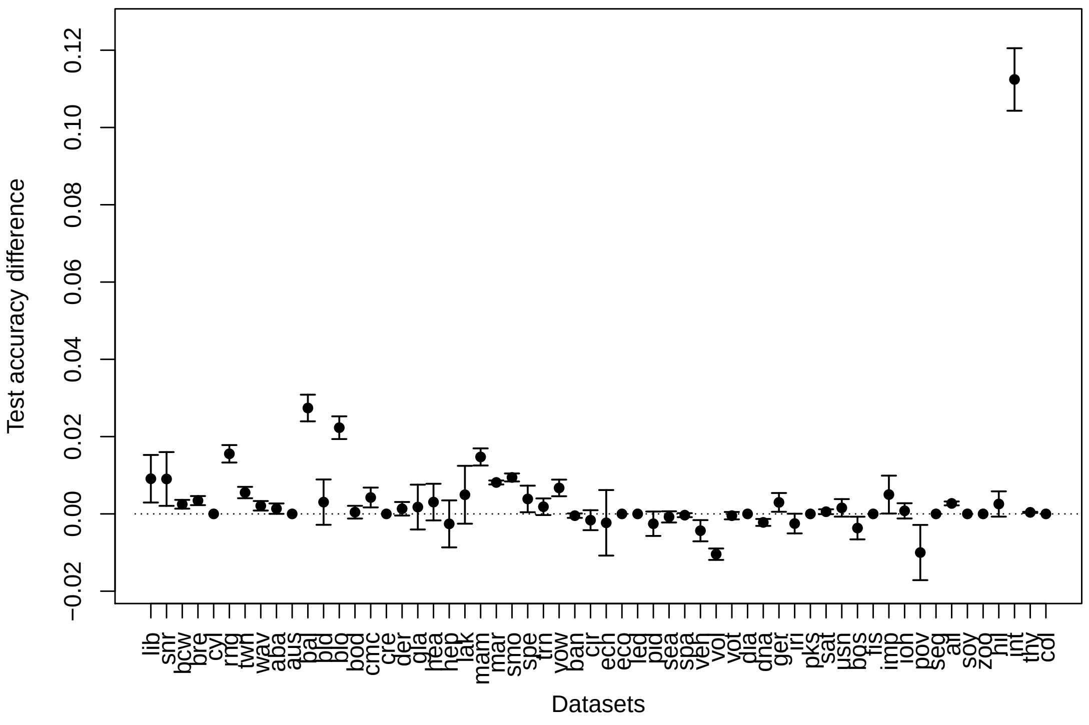

Figure 7. Confidence intervals of accuracy differences using the estimated m and the default m. The difference is significant when the confidence interval does not contain zero.

In Table 4, we conducted the randomized complete block design (RCBD) to test the overall significance of test accuracies between the estimated m and the default m. The block indicates 58 datasets in which 100 replications exist. In Table 4, the p-value of the main effect was <0.0001. Therefore, we can conclude there is strong evidence that the estimated m of the proposed method is significantly better than the default m in terms of test accuracy.

Table 4. Result of the analysis using randomized complete block design.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Treatment	1	0.05	0.052	22.11	<0.0001
Block	57	240.26	4.215	1777.32	<0.0001
Residual	11541	27.37	0.002		

4.3. Computational Consideration

A simpler method for estimating the feature set size is to use a grid search as in Boulesteix et al. [12]. The grid search method would explore all possible values of m using the OOB sample. We compared it with the proposed method in terms of computation time and accuracy to see if the proposed method is worthwhile.

Figure 8 shows the accuracy differences between the grid search and the proposed method (scaled on the left y-axis). It is quite clear that the two methods produced very similar outcomes. The ratios of two computing times are also presented in Figure 8 (scaled on the right y-axis). If the ratio (represented by the dashed line) is larger than 1, it means that the proposed method is faster. It is obvious that the proposed method is consistently faster than the grid search method while giving almost the same accuracy.

Obviously, not tuning m at all and keeping the default value will be faster than our proposed method. There might be a case that the increased training time by using the proposed method is not sufficiently beneficial. However, in general, the estimated m gives more accurate results than the default m as shown in Section 4.2. Therefore, we believe that the proposed method has sufficient merits.

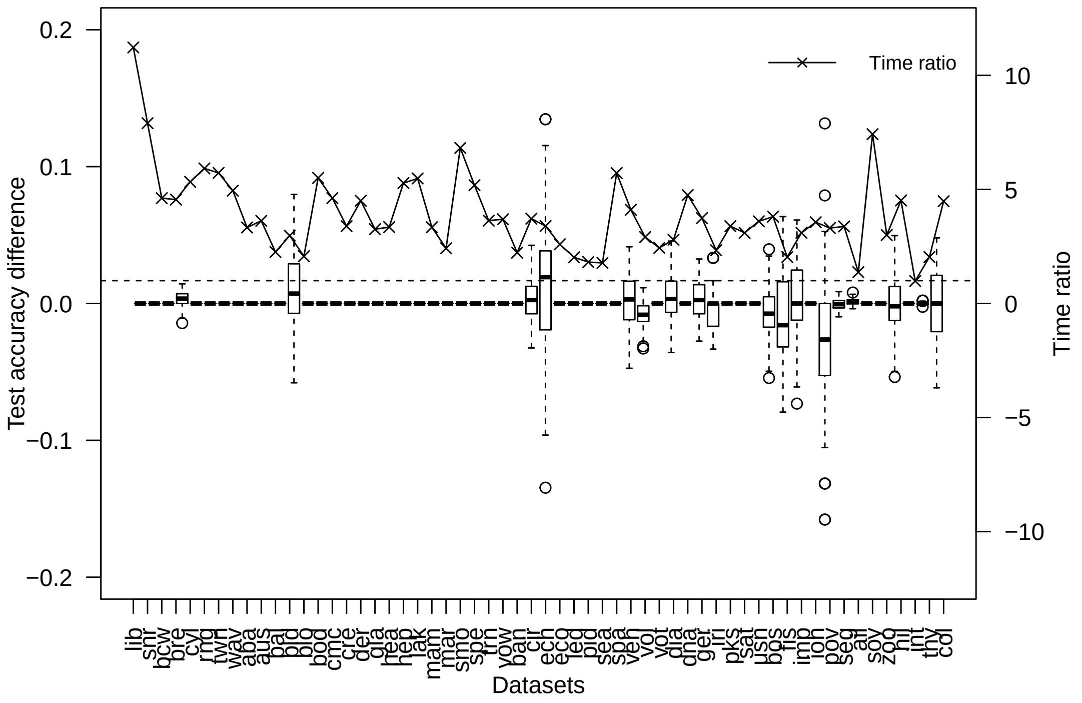

Figure 8. Confidence intervals of accuracy differences and the ratio of computing times between the proposed method and the grid search method.

5. Conclusions

In this research, we investigated whether the performance of Random forest is affected by the size of candidate feature set at each node of trees. We have shown through real data experiments that the hypothesis is true and the optimal size of candidate feature set differs from data to data. We also found that there is no specific functional relationship between the optimal size and the characteristics of each dataset.

We developed an algorithm to estimate the size of feature set using training data. The main idea of our proposed algorithm is to efficiently explore the search space using *SearchSize* method and the out-of-bag error. Using the 58 real data experiment, we have shown that Random forest using the estimated size from the proposed algorithm significantly outperformed those using the default size in many datasets. Moreover, we confirmed that the proposed method is statistically better than the default method in terms of overall test accuracy through the analysis of randomized complete block design.

The contribution of this research is that we provided a method to choose a better feature size m which is a critical option in Random forest. It would be greatly beneficial if one could choose a good feature size m in advance, because it will eventually increase the prediction accuracy of future observations. In addition, the proposed method is computationally efficient compared to the standard grid search method.

We only considered the classification problem in this article. However, a similar algorithm can be extended to the regression problem without difficulty.

Author Contributions: Conceptualization, H.K.; Methodology, H.K. and S.H.; Formal analysis, S.H.; Supervision, H.K.; Manuscript writing, H.K. and S.H.; Manuscript editing, H.K. and S.H.

Funding: This work was supported by Basic Science Research program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (No. 2016R1D1A1B02011696).

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsor had no role in the design of the study; in the analyses of data; in the writing of the manuscript; and in the decision to publish the results.

References

1. Lim, T.S.; Loh, W.Y.; Shih, Y.S. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach. Learn.* **2000**, *40*, 203–228.:1007608224229. [CrossRef]
2. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning: With Application in R*; Springer: New York, NY, USA, 2013; ISBN 978-1-4614-7137-0.
3. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: New York, NY, USA, 2001; ISBN 978-0-387-84857-0.
4. Dietterich, T.G. *Ensemble Methods in Machine Learning*; Springer: Berlin, Germany, 2000; ISBN 978-3-540-67704-8.
5. Hansen, L.K.; Salamon, P. Neural network ensembles. *IEEE Trans. Pattern Anal.* **1990**, *12*, 993–1001. [CrossRef]
6. Schapire, R.E. The strength of weak learnability. *Mach. Learn.* **1990**, *5*, 197–227. [CrossRef]
7. Freund, Y.; Schapire, R.E. Experiments With a New Boosting Algorithm. In Proceedings of the Thirteenth International Conference on Machine Learning (ICML '96), Bari, Italy, 3–6 July 1996; pp. 148–156.
8. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of online learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
9. Breiman, L. Bagging Predictors. *Mach. Learn.* **1996**, *24*, 123–140.:1018054314350. [CrossRef]
10. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32.:1010933404324. [CrossRef]
11. Huang, B.F.F; Paul, C.B. The parameter sensitivity of random forests. *BMC Bioinform.* **2016**, *17*, 331. [CrossRef] [PubMed]
12. Boulesteix, A.L.; Janitza, S.; Kruppa, J.; König, I.R. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2012**, *2*, 496. [CrossRef]
13. Freeman, E.A.; Moisen, G.G.; Coulston, J.W.; Wilson, B.T. Random forests and stochastic gradient boosting for predicting tree canopy cover: comparing tuning processes and model performance. *Can. J. For. Res.* **2015**, *46*, 323–339. [CrossRef]
14. Banfield, R.; Hall, L.; Bowyer, K.; Kegelmeyer W. A comparison of decision tree ensemble creation techniques. *IEEE Trans. Pattern Anal.* **2007**, *29*, 173–180. [CrossRef]
15. Loh W.Y. Improving the precision of classification trees. *Ann. Appl. Stat.* **2009**, *3*, 1710–1737. [CrossRef]

16. Kim H.; Kim H.; Moon H.; Ahn H. A weight-adjusted voting algorithm for ensemble of classifiers. *J. Korean Stat. Soc.* **2010**, *40*, 437–449. [CrossRef]
17. Asuncion, A.; Newman, D.J. *UCI Machine Learning Repository*; University of California, Irvine, School of Information and Computer Science: Irvine, CA, USA, 2007. Available online: http://archive.ics.uci.edu/ml/index.php (accessed on 2 October 2018).
18. Leisch, F.; Dimitriadou, E. *Mlbench: Machine Learning Benchmark Problems*; R package version 2.0-0; R Foundation for Statistical Computing: Vienna, Austria, 2010; ISBN 3-900051-07-0.
19. Heinz G.; Peterson L.J.; Johnson R.W.; Kerk C.J. Exploring relationships in body dimensions. *J. Stat. Educ.* **2003**, *11*. [CrossRef]
20. Kim H.; Loh W.Y. Classification trees with bivariate linear discriminant node models. *J. Comput. Grap. Stat.* **2003**, *12*, 512–530. [CrossRef]
21. Kim H.; Loh W.Y. Classification trees with unbiased multiway splits. *J. Am. Stat. Assoc.* **2001**, *96*, 589–604. [CrossRef]
22. Terhune J.M. Geographical variation of harp seal underwater vocalizations. *Can. J. Zool.* **1994**, *72*, 892–897. [CrossRef]
23. Statlib. Datasets Archive. Carnegie Mellon University, Department of Statistics. 2010. Available online: http://lib.stat.cmu.edu/datasets (accessed on 2 October 2018).
24. Breiman, L.; Friedman, J.; Olshen, R.; Stone, C.; *Classification and Regression Trees*; Chapman and Hall-Wadsworth: New York, NY, USA, 1984; ISBN 978-0412048418.