J For Res (2008) 13:275–285
DOI 10.1007/s10310-008-0084-6

ORIGINAL ARTICLE

Verification for generalizability and accuracy of a thinning-trees selection model with the ensemble learning algorithm and the cross-validation method

Yasushi Minowa

Received: 21 May 2007 / Accepted: 5 June 2008 / Published online: 24 July 2008

Abstract For the purpose of making a highly effective model in relation to the selection of trees for thinning for various forestry goals, the author examined the generalizability and accuracy of models using various ensemble learning algorithms and the *m*-fold cross-validation method. These techniques make it possible to improve discrimination accuracy by combining or integrating multiple learning results whose accuracies are not very high. WEKA, which is a machine learning tool for data mining programmed in Java machine language, was used to verify the results of the simulation models. The number of samples was 503. Pattern-recognition algorithms in this study used five classification-type models and one function-type model. It was found that: (1) without cross validation, two pattern-recognition algorithms can be classified as having comparatively high discrimination accuracy; (2) with cross validation, discrimination accuracy decreased as a whole, but was not very different from that without cross validation, and (3) from the viewpoint of generalizability, we constructed a model at around 70% discrimination accuracy. In order to construct more effective models, we need to design the model to utilize certain algorithms or to build in re-sampling methods such as ensemble learning and cross validation. Additionally, in the case of small sample datasets, ensemble learning is an effective method for constructing efficient models.

Keywords Ensemble learning · *m*-fold cross validation · Pattern-recognition algorithm · Thinning-trees selection · WEKA

Electronic supplementary material The online version of this article (doi:10.1007/s10310-008-0084-6) contains supplementary material, which is available to authorized users.

Y. Minowa (✉)
Graduate Scholl of Life and Environmental Sciences,
Kyoto Prefectural University,
1-5 Shimogamo-hangi cho, Sakyo-ku,
Kyoto 606-8522, Japan
e-mail: sharmy@uf.kpu.ac.jp

Introduction

In forestry the amount of information that can be acquired from field studies is small in comparison with the size of the tree population. Hence, we need to ask, "What degree of interpretability does this information have?" Recently, high-speed, high-accuracy data processing has rapidly improved computer resources. As for forest information, techniques such as remote sensing, geographic information systems (GIS), and global positioning systems (GPS) enable us to obtain vast amounts of new information in both spatial and temporal domains. However, in the context of these new techniques, it is still necessary to obtain supervised data of higher accuracy based on field studies for purposes of prediction and classification. Although it is preferable to obtain as large a dataset as possible, the number of data often falls short of that required because the time or labor for a tree census in field studies increases in proportion to the number of data obtained from those investigations. To predict or classify using few sample data it is possible to model from hypotheses by statistical processes; however, the population is assumed to be normal, the measured values are assumed to be unbiased, etc. Moreover, because it is thought that many natural phenomena can be described with complicated systems expressed in nonlinear differential equations (Toraichi 1993), it is difficult to define theoretical distribution patterns or functional types with these phenomena. Thus, we need to pay sufficient attention to mechanical statistical processes or their statistical

outcomes because they can lead to flawed results. For example, we estimated the site indices for Japanese cedar using a GIS (Minowa et al. 2005a, b). In these studies, measured tree heights in the training area were related to four topographical factors: effective relief, topographical exposure, storage capacity, and flow accumulation, all calculated by use of a GIS. Finally, using a neural network (Minowa et al. 2005a) or machine learning (Minowa et al. 2005b), we estimated the site indices of the total study area. However, one of the problems we encountered was that there was a large number of sites to be estimated based on a relatively small amount of sample data. As for the number of data, for example when using a classification tree such as automatic interaction detection (AID) or a classification and regression tree (CART), it is desirable that sample data of at least 1,000 cases are used in order to get significant results; sample data of over 2,000 cases can be used as verification (Ootaki et al. 1998). Although it is desirable to obtain as many data as possible, we cannot always gather a sufficient number of data in a tree census because of the above-mentioned reasons. Comparing a number of sample data in precedence studies related to the estimation of site indices, for instance, the number of sample plots of Teraoka et al. (1991) was 52; that of Mitsuda et al. (2001) was 40. Although these sample numbers were not necessarily large, these studies were constructed with relatively high estimation accuracy models. However, it is not guaranteed that the discrimination accuracy for unknown data is always high. The important thing to consider here is the need to evaluate not only error rates for the learning data but also generalizability.

Generalizability refers to the validity of applying learning results calculated from learning sample data to unknown data (Aso et al. 2003). In general, the recognition performance for unknown data is worse than that for learning sample data. It is, however, said that the generalizability is high when the discrimination accuracy for unknown data is not much lower than that for the learning sample data. For instance, in the *k*-NN method (*k*-nearest-neighbors classification rule), one of the classical pattern-recognition methods and a method for memorizing all learning data, error rates for learning data decrease although the discrimination boundary becomes complicated. Besides, for example, the error rate for the learning data in the 1-NN method ($k = 1$ in the *k*-NN method) is 0, unless perfectly duplicated data are classified into different classes. In the method for evaluating the performance of the discriminating function, test sets (data for training, not learning data) are prepared and evaluated by use of the results from the learning data. As the devised method, a full dataset is divided into parts for learning and training; multiple evaluations are performed by varying the means of division (Aso et al. 2003).

The author has been studying modeling for thinning-tree selection with a neural network (Minowa 1997) and machine learning (Minowa 2005), which are used to sample data to be mixed with qualitative and quantitative information. The selection of trees for thinning is highly subjective compared to other aspects of forestry and goes beyond simply measuring tree attributes such as DBH and height. Thus, applications of artificial intelligence such as neural networks can be effectively applied for processing forest information of the sort where the subjective element seems to be high: for instance, visual assessment of tree height, classification of trees or forest types, decision making in forest management and planning, etc. There are some preceding studies of applications to forestry of fuzzy theory (Bare and Mendoza 1992; Yamasaki et al. 1996; Kivinen and Uusitalo 2002), neural networks (Guan and Gertner 1991; Liu et al. 2003), genetic algorithms (Ichihara et al. 1996; Fisher et al. 2000), decision trees (Tasaka et al. 2001), and self-organizing maps (Fujino and Yoshida 2006). In particular, prior research exists on thinning-tree selection using artificial intelligence (Yamasaki et al. 1996). Many of these studies, including this author's, suggest that these methods are more effective applications than conventional methods such as linear regression, principle-component analysis, and quantification methods. Nevertheless, the author's studies are focused on whether these methods are applicable to thinning-tree selection and do not fully examine the generalizability of the model. While increasing sample data would improve the accuracy of the models, this is easier said than done. Furthermore, the number of humans who are trained to select trees with satisfactory accuracy is not sufficient.

The purpose of this paper is to examine the generalizability and accuracy of thinning-tree selection models in order to make a highly effective model using only sample data used in previous studies. First, adding to two pattern-recognition algorithms adopted in previous studies (neural networks and C4.5 machine learning), four pattern-recognition algorithms were newly applied and the performance of these models was compared. The models were then constructed using various learning models incorporating each pattern-recognition algorithm into a cross-validation method to verify the generalizability. While the cross-validation method is expected to be capable of constructing a highly generalizable model, the estimation accuracy of that model tends to be lower than that without cross validation (Ootaki et al. 1998). To overcome this disadvantage, this study applied a statistical technique called "ensemble learning" and performed many simulations to select higher-generalizability models.

Materials and methods

Ensemble learning and cross validation

In pattern-recognition algorithms, a decision-tree model was used for classification-type models and a neural network model for function-type models.

"Tree"-based models contribute to both nonlinear regression analysis and discriminant analysis (Jin 2005); typically, the model is called a "regression tree" in the case of regression problems and a "classification tree" or "decision tree" in the case of classification problems. A decision tree is one of the methods for expressing knowledge in relation to the purpose of classification; this algorithm is able to clarify the decision-making sequence with a "tree-like" graph (Morimura et al. 1999). The decision tree has the advantage that it offers users a model by which it is clearly shown "how a decision should be performed" or "how a decision was made" (Duda et al. 2001). There are many algorithms for decision trees; in this study, J48, NBTree, RandomTree, and REPTree, all of which are included in WEKA (Witten and Frank 2005; http://www.cs.waikato.ac.nz/ml/weka/), were applied. These algorithms are representative techniques for classification problems and are not wholly restricted to data sets (Jin 2005). Additionally, J48, NBtree, RandomTree, etc., are module names in WEKA. WEKA, which was developed for researchers in machine learning by other researchers, Witten and Frank of the University of Waikato, consists of free software for data mining to be programmed in Java machine language.

J48 is the WEKA version of the C4.5 algorithm. The C4.5 algorithm is a descendant of an earlier program, called iterative dichotomizer version 3 (ID3), developed by Quinlan (1993), using an algorithm that added inductive learning to an expert system. The C4.5 algorithm performs inductive learning of production rules from examples and enables researchers to form simple decision trees. The information entropy, called the gain ratio, was used as the criterion for evaluating the branching in the C4.5 algorithm.

NBTree, developed by Kohavi (1996), is a classifier for inducing a decision tree based on naïve Bayes rules. Naïve Bayes is a simple system based on Bayes' probability model, and is called "naïve" because of the assumption that each attribute is independent and latent attributes do not exert an influence. Because naïve Bayes is a simple system, its fields of application are wide-ranging; for instance, the Bayesian filter is typical of applications with the naïve Bayes (Duda et al. 2001).

RandomTree is an algorithm that randomly utilizes explanatory variables for the branching. In order to use duplicate variables for the branching, the RandomTree algorithm induces more growing "trees" than the C4.5 algorithm (Jin 2005).

REPTree is an algorithm for constructing a decision or regression tree using information gain/variance reduction and prunes it using reduced-error pruning (Witten and Frank 2005); it has the characteristic of calculation at higher speeds than other algorithms.

For function-type models, MultilayerPerceptron, which is the name of the module included in WEKA, was used as a neural network model in a multilayer perceptron. The neural network is a computer model applied to an artificially simulated process of neurons. This enables treatment of both quantitative and qualitative data at the same time. MultilayerPerceptron is a three-layer back-propagation model, which is one of the multilayered feed-forward neural networks; this is a supervised learning method proposed by Rumelhart et al. (1986), and a steepest-descent method for minimizing multivariate nonlinear problems (Rumelhart et al. 1986).

Large-scale and complicated learning models have the advantage of being able to represent complex input–output relationships; in fact, their expressive power is generally superior to that of other models. On the other hand, these models have two definite disadvantages: the amount of calculation increases with an increase of parameters, and the generalizability lowers as it overadjusts for the noise components in the data. To resolve these disadvantages, a statistical technique called "ensemble learning" is proposed (Aso et al. 2003). Ensemble learning is a method for improving learning accuracy by combining or integrating multiple learning results whose accuracies are not very high (Jin 2005). For the methods of combining or integrating for multiple learning results, a majority decision is applied in the case of classification problems, while an average is applied in the case of regression problems. Ensemble learning can acquire the same ability as large and complicated models by virtue of applying comparatively simple learning models and learning rules of appropriate calculation amounts (Aso et al. 2003). Comparing a complicated model which consists of 100 parameters with a model combining 10 sets of a simple model each consisting of 10 parameters, the combined model is expected to achieve higher generalizability than the complicated model. Thus, ensemble learning can achieve a practical learning time and higher generalizability by combining a relatively simple learning model with a learning rule having an appropriate calculation burden.

In this study, bagging, boosting and random forests, which are representative algorithms in ensemble learning, were adopted.

"Bagging" is a coined word combining character strings of the phrase, "bootstrap aggregating". This algorithm

creates multiple learning data by repeating a bootstrap method (called a re-sampling method) using the sample data given; these learned data can then be independently and parallelly learned by multiple classifiers, and the classification is performed to a majority decision (Breiman 1996). The algorithm of bagging is roughly described in the following sequence (Jin 2006; Aso et al. 2003):

Consider the training sets $(x_1, y_1),\ldots,(x_N, y_N)$ composed of N samples ($i = 1,\ldots,N$).

Step (1) First, this algorithm calculates m times from supervised data by sampling with a replacement method, and makes a new sub-dataset for training, then constructs an h_t learner of regression/classification models.

Step (2) By repeating Step (1) B times, regression/classification models are constructed in B number of pieces ($h_t(x)$; $t = 1,\ldots,B$).

Step (3) For regression problems, the result of bagging outputs $H_m(x)$, which is an average of B number of pieces, as follows:

$$H_m(x) = \frac{1}{B}\sum_{t=1}^{B} h_t(x)$$

For classification problems, the result of bagging outputs $H_d(x)$, which is a decision by majority, as follows:

$$H_d(x) = \arg\max_{y\in Y}|\{t|h_t(x) = y\}| = \arg\max_{y\in Y}\sum_{t=1}^{B} I(h_t(x) = y)$$

If the event shown by the argument in parentheses is true, $I(\cdot)$ equals 1. If the event is not true, $I(\cdot)$ equals 0.

Boosting is a method for constructing a higher-accuracy classifier from a combination of multiple learned classifiers, while the weight of the sample data sequentially changes. In this study, AdaBoost was used as a representative method (Freund and Schapier 1997; Aso et al. 2003). The algorithm of boosting to be included in AdaBoost is simply described in the following.

Consider the training sets $(x_1, y_1),\ldots,(x_N, y_N)$ composed of N samples ($i = 1,\ldots,N$). Note that $x_i \in X, y_i \in Y = \{-1, +1\}$. Let $D_t(i)$ be the weight of the i sample after t times learning.

Step (1) The initial value of weight is set.

$$D_1(i) = \frac{1}{N}$$

Step (2) For $t = 1, \ldots,T$, the classifier $h_t : X \to Y$, which minimizes the error ratio (ε_t), is based on distribution D_t.

$$\varepsilon_t = Pr_{Dt}\{h_t(x_i) \neq y_i\} = \sum_{i^*=1}^{N} D_t(i^*)$$

(Note that i^* is the number of i's which h_t mistook.)

Step (3) The reliability of results (α_t) is calculated using the error ratio of h_t.

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon t}{\varepsilon t}\right)$$

Step (4) The distribution D_t is renewed.

$$D_{t+1}(i) = \frac{D_t(i)\exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$$

(Note that Z_t is a standardization factor for ensuring $\sum_{i=1}^{N} D_t + 1(i) = 1$, $Z_t = \sum_{i=1}^{N} D_t(i)\exp\{-\alpha_t y_i h_t(x_i)\}$)

Step (5) The results are output to a majority decision $H_d(x)$ with the weight by reliability (α_t).

$$H_d(x) = \mathrm{sign}\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right)$$

"Random forests" is a comparatively new ensemble learning algorithm proposed in 2001 by Breiman (2001), formerly a proponent of bagging. Fundamentally, the algorithm of random forests is a method for repeating the bootstrap and the bagging. In stark contrast to bagging, random forests has the advantage that the calculation burden of higher-dimension data is lightened by using a randomly sampled sub-dataset, while bagging uses all the variables (Jin 2005).

m-fold cross validation

In methods for estimation of prediction accuracy for unknown data, there are many methods such as the resubstitution method, the test-sample method, and the m-fold cross-validation method (Ootaki et al. 1998). The m-fold cross-validation method is a descendant of the test-sample method and was devised in order to estimate unbiased prediction errors from a few sample data. Figure 1 illustrates the conceptual scheme of m-fold cross validation. In m-fold cross validation, the learning data set N is divided into m parts (sub dataset $N_i = N_1,\ldots,N_m$, $i = 1,\ldots,m$), in which the classes are represented in approximately the same proportions as in the full data set. Then, each in turn is used as testing data for a prediction-error estimate and the $(m - 1)$ remainder is used as data for training. Cross validation performs a calculating-error estimate and constructs models with the first sub-dataset for estimating the prediction error and the remainder for training. Next, this method performs a similar procedure with a second sub-dataset for estimating and the remainder for training. After repeating the computation m times, the value which is an average of the error estimates of each turn is used as the estimation accuracy of the model for unknown data (Ootaki et al. 1998; Duda et al. 2001).

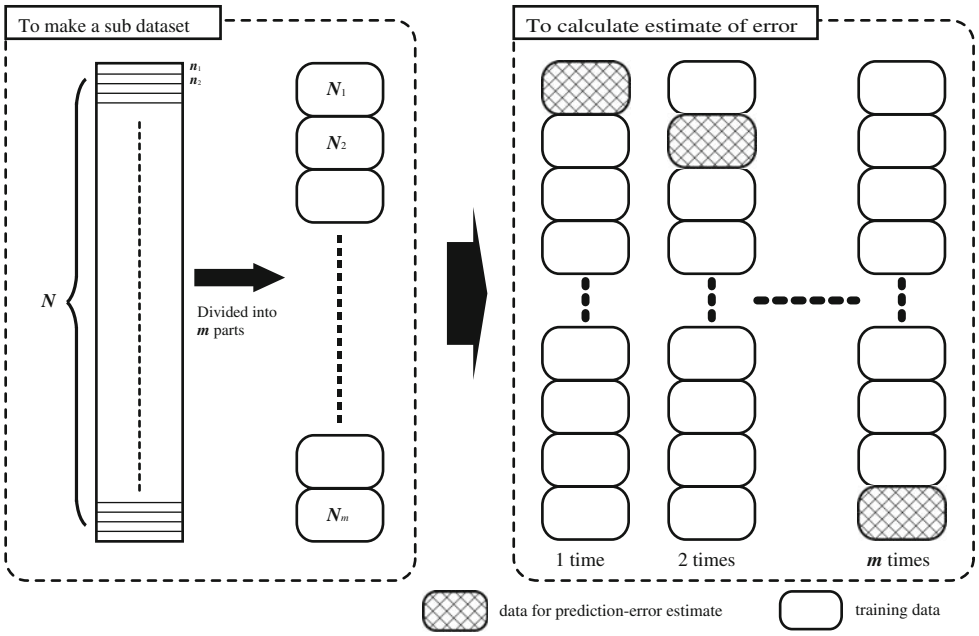

Fig. 1 Conceptual scheme of the *m*-fold cross-validation method

With cross validation, all learning data are used once as data for estimating the prediction error, and this method would construct a model with $(m - 1)$ times training data from learning data N on appearance. In other words, the m-fold cross validation would obtain a number of samples equal to $(m - 1) \times N$ for learning a dataset (Ootaki et al. 1998). Then, the estimate of the error $R_i(d)$ $(i = 1,\ldots,m)$ is obtained from a sub-dataset. The mean value $R(d)$ and variance $SE^2(R(d))$ of the estimated error are shown in the following equations.

$$R(d) = \frac{1}{m}\sum_{i=1}^{m} R_i(d)$$

$$SE^2\{R(d)\} = \frac{1}{m}\sum_{i=1}^{m} \{R_i(d) - R(d)\}^2$$

Additionally, cross validation multiples $m = 2, 3, 4, 5, 10$ are mainly used (Jin 2004). The m sub-dataset is automatically initialized by designating the m number in WEKA (Witten and Frank 2005).

Input and output data

Similar to Minowa (1997), this study used data from a thinning trial from the Tokyo University Forests in Chiba. This plot was composed of a sugi (*Cryptomeria japonica* D.DON) stand (99 years old in 1993) thinned five times (Suzuki et al. 1995). The study area was 1.10 ha, the number of trees, 503, the average tree height, 31.8 m, and the average DBH, 46.8 cm.

Input variables to the classifiers were made up of five factors: DBH, tree height, crown volume, stem quality, and defects. Output variables to the classifiers consisted of one factor (one of four thinning results). Tables 1 and 2 show details of input and output variables.

Simulation

With the first simulation, the author attempted to classify only pattern-recognition algorithms. Although "random forests" is one of the ensemble learning algorithms, it is possible that RandomForest, which is the name of a module included in WEKA, also constructs decision trees itself. Thus, RandomForest was also adopted as a pattern-recognition algorithm. As a result, the number of pattern-recognition algorithms in the classification-type models was five: J48, NBTree, RandomForest, RandomTree, and REPTree.

With the second simulation, the author performed classification by combining ensemble learning and the cross-validation method. By combining either bagging (WEKA module name, Bagging) or boosting (WEKA module name, AdaBoostM1) with five classification types, simulation was carried out. The number of cross-validation times was two ($m = 2$), three ($m = 3$), five ($m = 5$), and ten ($m = 10$). For cross validation in WEKA, training data were divided into m parts for input to the computer. When the list order of the training data differs, the output results are also different. Thus, the author made ten training datasets rearranged using random numbers and respectively simulated ten training datasets, one at a time.

Table 1 Attributes of data and their scales

Attribute name	Scale	Number of trees[a]	Measurement method[b]
DBH (cm)	min.: 22.7, avg.: 46.8, max.: 87.0	–	O
Height (m)	min.: 18.0, avg.: 31.8, max.: 47.0	–	O
Crown volume	{Very small, small, medium, large}[m]	{37, 104, 173, 189}	J
Stem quality	{Bad, normal, good}[n]	{145, 168, 190}	J
Defect-decay[c]	{Yes, no}	{20, 483}	J
Defect-cut[d]	{Yes, no}	{89, 414}	J
Defect-curve[e]	{Yes, no}	{18, 422}	J
Defect-fork[f]	{Yes, no}	{18, 485}	J
Defect-small crown[g]	{Yes, no}	{37, 466}	J
Defect-small DBH[h]	{Yes, no}	{8, 495}	J
Defect-low crown height[i]	{Yes, no}	{17, 486}	J
Defect-adjacent trees[j]	{Yes, no}	{199, 304}	J
Defect-bias[k]	{Yes, no}	{10, 493}	J
Defect-broken top[l]	{Yes, no}	{4, 499}	J

[a] The number of each item shown in the scale

[b] *O* observation; *J* judgement of measurer

[c] Rotted stems

[d] Injured stems

[e] Curved stems

[f] Forked trees

[g] Crown volume is extremely small compared with other parts

[h] DBH is extremely small compared with other parts

[i] Crown height is lower than those of other trees

[j] This tree was cut to widen the distance of stems, even if the quality of wood is superior to that of others

[k] Abnormal crown shape

[l] Top of tree is broken

[m] Each tree is divided into four scales by eye

[n] Each tree is divided into three scales by eye

Table 2 Output data

Thinned/unthinned	Thinning method	Abbreviation	Number of trees	Thinning rate (%)			
				Number of trees		Volume	
				Plan	After thinning	Plan	After thinning
Unthinned	–	Unthinned	152	–	–	–	–
Thinned	Protection of environment[a]	Type-a	200	40	39.8	25	25.5
Thinned	Forest management with long rotations[b]	Type-b	93	60	58.3	40	42.8
Thinned	Multistoried forest management[c]	Type-c	58	70	69.8	55	55.0

[a] The aim was to control the forest density in overcrowded old-aged forests

[b] The aim was to maximize production of high-quality timber with utilization thinning and long rotations

[c] The aim was to maintain an appropriate forest stand density for growing underplanted trees and to widen the distance among stems as compared with thinning for management with long rotations

With the third simulation, for comparison with classification-type models, a neural network (WEKA module name, MultilayerPerceptron) was executed for function-type models. The numbers of hidden units were 10 and 29, and the numbers of learning iterations were 1,000, 5,000, and 10,000.

Additionally, options for each algorithm were adapted to the default parameters.

In the scale of discrimination accuracy, the correctness ratio (CR) expressed by the following equation was used.

$$\mathrm{CR}(\%) = \left(1 - \frac{\mathrm{N_{error}}}{\mathrm{N_{learning}}}\right) \times 100$$

(note that $\mathrm{N_{error}}$ is the number of errors which could not be correctly classified and $\mathrm{N_{learning}}$ is the number of all learning data). The minimum–average–maximum values for ten iterations are shown as simulation results. As a measure of the efficiency of ensemble learning, the improvement ratio (IR) expressed by the following equation was used.

$$\mathrm{IR} = \frac{\text{CR with ensemble learning}}{\text{CR without ensemble learning}}$$

IR shows the efficiency of ensemble learning; for example, ensemble learning is significant when IR is over 1.00.

As a scale of generalizability, the author defined the generalizability ratio (GR) by the following equation.

$$\mathrm{GR}(\%) = \frac{\text{CR with cross validation}}{\text{CR without cross validation}} \times 100$$

It is shown that the estimation accuracy of the model for unknown data becomes higher as GR becomes higher.

Results and discussion

Classification by classification-type models

Figure 2 illustrates the efficiency of ensemble learning and *m*-fold cross validation in classification-type models. Without cross validation, the discrimination accuracy differed greatly for different pattern-recognition algorithms. The CR for an average of ten simulations (CR_m) was between 76.9 and 100.0%. RandomTree resulted in the best classification (CR_m = 100.0%) while REPTree resulted in the worst classification (CR_m = 76.9%). RandomForest and RandomTree could achieve almost correct classification without ensemble learning. By adding ensemble learning to these, the discrimination accuracy tended to improve. In five pattern-recognition algorithms, REPTree generally resulted in low CR. J48, NBTree, and REPTree tended to improve discrimination accuracy when ensemble learning was applied; furthermore, J48 led to a remarkable improvement in discrimination accuracy when combined with AdaBoostM1. Across the board, the discrimination accuracy of AdaBoostM1 was more effective than that of Bagging.

With cross validation, discrimination accuracy decreased as a whole. By comparison to performance without cross validation, all algorithms gave similar results—the difference in discrimination accuracy between each model became small compared with that without cross validation. Specifically, the difference between maximum and minimum CR_m values with cross validation was smaller than that without cross validation; for example, the difference with cross validation ranged from 5.1% (m = 10, $73.6_{\mathrm{J48}} - 68.5_{\mathrm{RandomTree}}$) to 7.4% ($m$ = 2, $73.1_{\mathrm{NBTree,\ RandomForest}} - 65.7_{\mathrm{RandomTree}}$), while that without cross validation was 23.1% ($100_{\mathrm{RandomForest,\ RandomTree}} - 76.9_{\mathrm{REPTree}}$). The CR of RandomTree was lower than that of other pattern-recognition algorithms; CR_m ranged from 65.7% (m = 2) to 68.5% (m = 10). CR_m of other models ranged from 69.8% (m = 2, REPTree) to 73.6% (m = 10, J48).

In terms of the frequency of cross validation, each algorithm showed the tendency that CR was only slightly improved with an increasing number of cross validations. The degree of improvement for J48 and RandomTree was superior to that for other methods.

Comparison with function-type models

Figure 3 illustrates the efficiency of ensemble learning and cross validation in function-type models. Without cross validation, discrimination accuracy was generally high. The CR_m was between 89.0 and 96.7%. The case in which the hidden units number was 29 and the number of learning iterations was 10,000 resulted in the best classification (CR_m = 96.7%), and the case in which the hidden units number was 10 and the number of learning iterations was 1,000 resulted in the worst classification (CR_m = 89.0%). When modules were added to ensemble learning, the discrimination accuracy tended to improve, and AdaBoostM1 was shown to give good results compared with Bagging.

With cross validation, discrimination accuracy decreased more in function-type than in classification-type models. As with the classification-type models, even though CR itself was reduced, the difference between each model became small; CR_m ranged from 70.9% (m = 2, h = 29, learning iterations = 5,000) to 73.2% (m = 10, h = 10, learning iterations = 1,000).

Depending on the frequency of cross validation, the results of classification-type models were slightly superior to those of function-type models. However, the differences between the maximum and minimum values for function-type models were smaller than those for classification-type models.

Effect of ensemble learning

Table 3 shows the effect of ensemble learning, the values show differences between CR_m of estimation with ensemble learning and that without ensemble learning for each cross-validation multiple. With addition of ensemble learning, discrimination accuracy slightly improved as a

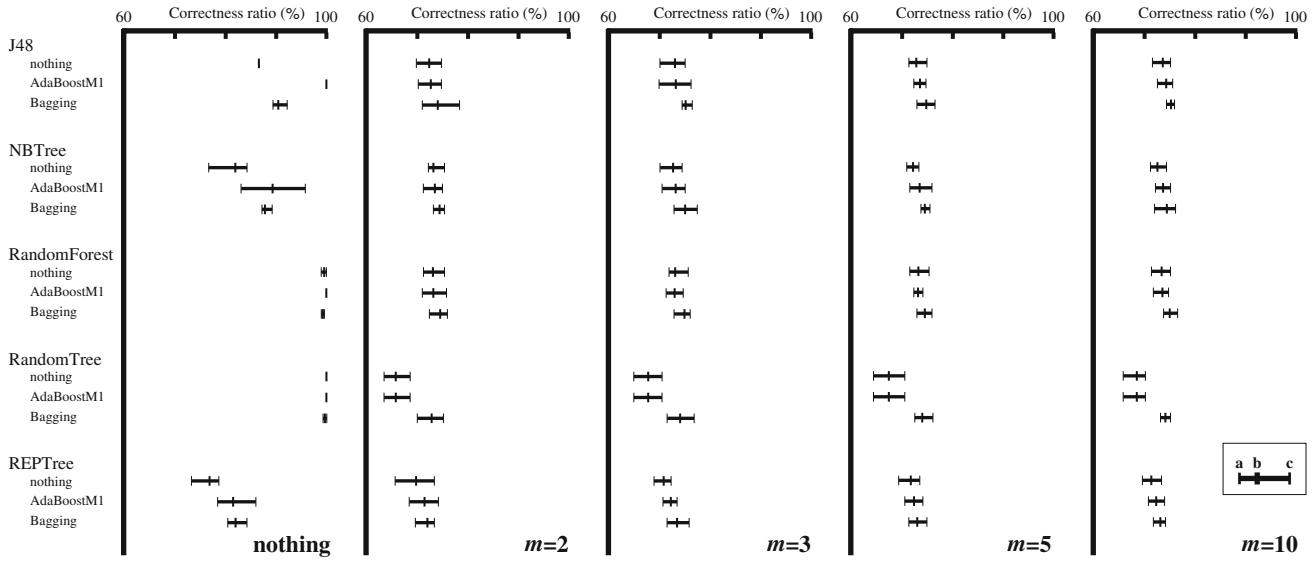

Fig. 2 Correctness ratios of the classification-type methods for the effect of cross validation and ensemble learning: *a* minimum, *b* average, and *c* maximum values for ten iterations

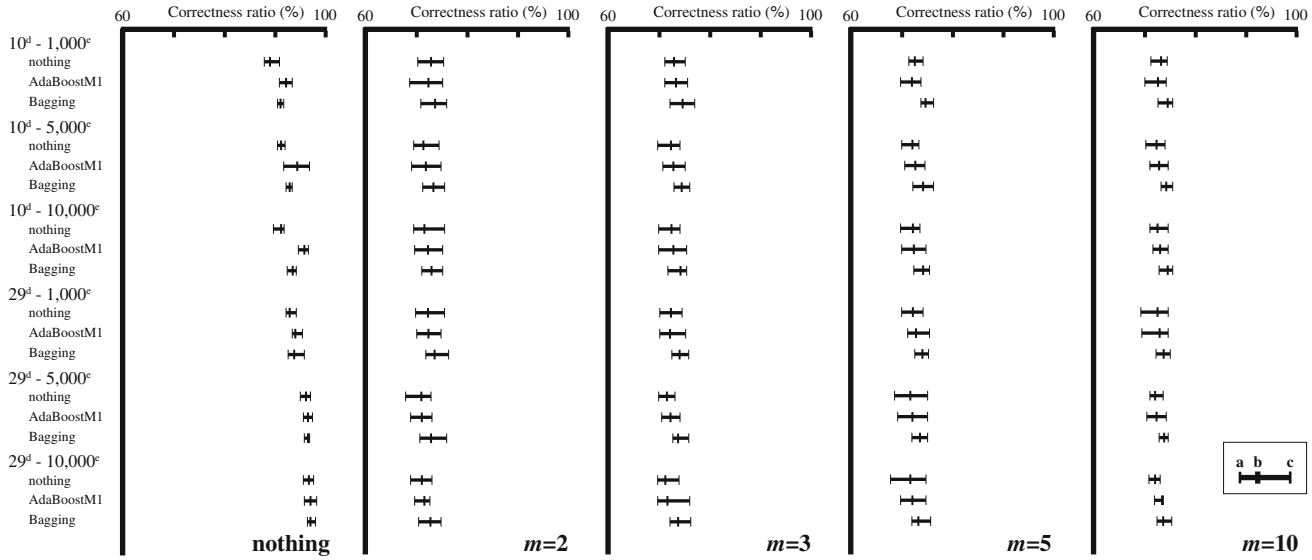

Fig. 3 Correctness ratios of the function-type methods for the effect of cross validation and ensemble learning: *a* minimum, *b* average, and *c* maximum values for ten iterations. [d]Number of hidden units [e]Number of learning iterations

whole. The results of Bagging were superior to those of AdaBoostM1 in both classification-type and function-type methods. The mean value of Bagging for both classification methods was 2.2 whereas that of AdaBoostM1 was 1.0. Moreover, results from classification-type methods were superior to those of function-type methods in both AdaBoostM1 and Bagging. The mean value of all classification-type methods was 1.4 (AdaBoostM1) and 2.8 (Bagging), while that of all function types was 0.6 (AdaBoostM1) and 1.7 (Bagging).

Confusion patterns

Table 4 (see Electronic Supplementary Material) shows the confusion patterns for each thinning type. In order to discuss the general tendency, CR_m and IR were used. Without cross validation of the classification-type models, discrimination accuracy was shown to have good results through all simulations, except for the case of type-b and type-c using REPTree. In particular, RandomForest and RandomTree were able to classify each factor with

Table 3 Difference between CR_ms with and without ensemble learning

Cross validation	Ensemble learning	J48	NBTree	RandomForest	RandomTree	REPTree	MultilayerPerceptron					
							10[a]–1,000[b]	10[a]–5,000[b]	10[a]–10,000[b]	29[a]–1,000[b]	29[a]–5,000[b]	29[a]–10,000[b]
Nothing	AdaBoostM1	13.3	7.4	0.4	0.0	4.7	3.2	3.2	4.6	1.1	0.4	0.3
	Bagging	3.8	5.9	−0.3	−0.3	5.1	2.1	1.7	2.3	0.9	0.4	0.3
$m = 2$	AdaBoostM1	0.3	0.3	0.1	0.0	1.7	−0.5	0.5	0.7	0.1	0.1	0.5
	Bagging	1.7	1.2	1.4	7.1	2.2	0.8	2	1.4	1.3	1.9	1.9
$m = 3$	AdaBoostM1	0.2	0.5	−0.1	0.0	1.4	0.4	0.5	0.4	−0.2	0.7	0.4
	Bagging	2.1	2.4	1.8	6.3	2.7	1.7	2.1	1.8	1.7	2.2	2.5
$m = 5$	AdaBoostM1	0.7	1.3	−0.1	0.0	0.6	−0.6	0.6	0.2	0.6	0.4	0.4
	Bagging	2.0	2.3	1.3	6.6	1.3	2.1	2.1	2	1.9	1.9	1.6
$m = 10$	AdaBoostM1	0.7	1.1	0.2	0.0	1.0	−0.6	0.5	0.5	0.4	0.3	−0.1
	Bagging	1.6	1.8	1.6	5.7	1.8	1.3	1.9	2	1.2	1.8	1.6

[a] Number of hidden units

[b] Number of learning iterations

approximately 100% accuracy. Furthermore, the discrimination accuracy was improved by addition of ensemble learning, and the effect of combining J48 and AdaBoostM1 was remarkable.

With cross validation, the CR_m of "unthinned" was considerably high compared to that of other factors. In contrast, the discrimination accuracy of other factors for "thinning" resulted in extremely bad classification. As for the number of cross validations, CR_m resulted in approximately the same values regardless of the number of cross validations. In comparison with the learning models that used all sample data as a training data, it is considered that the type of supervised patterns that must be originally included in the training data is decreased by dividing the sample data using cross validation.

As for IR values through classification-type models, the minimum, average and maximum IR values were 0.94, 1.05, and 1.62, respectively. In terms of the effect of ensemble learning, the cases of type-b and type-c were able to obtain higher efficiency for ensemble learning. Moreover, the case of few validation times tended to improve results compared with the case of many validation times.

Without cross validation of function-type models, these models resulted in higher CR_m than classification-type models through all simulations. In addition, discrimination accuracy hardly decreased, even if the numbers of hidden units and learning times were low.

With cross validation, the CR_m of type-b and type-c were substantially reduced; about half the learning data were incorrectly classified. Even if the number of hidden units and learning iterations increased, CR_m was not always improved. Furthermore, there were many cases in which the CR_m was reduced in type-c.

As for IR values through function-type models, the minimum, average, and maximum IR values were 0.70, 1.01 and 1.12, respectively. The IR values of function-type models were smaller than that of classification-type models as a whole. Regarding the effect of ensemble learning through all simulations, the IR was lower in function-type models than in classification-type models; there were many cases to this effect.

Generalizability

Figure 4 illustrates the generalizability according to the different classification algorithms. In classification-type methods, the GR of each model was significantly different. The GR for classification-type methods ranged from 65.7% (m = 2, RandomTree) to 93.3% (m = 5, REPTree). The CR of REPtree was not very high; however, REPtree could construct a model whose generalizability was higher than that of other models. As for function-type methods, the GR for function-type methods ranged from 73.4% (m = 2, h = 29, learning iterations = 10,000) to 82.2% (m = 10, h = 10, learning iterations = 1,000).

Generally, neural networks can improve the estimation accuracy of models with increasing numbers of learning iterations or hidden units. However, the problem called "overlearning" or "overfitting" occurs, when the learning is carried out excessively: i.e., the prediction accuracy for unknown data decreases. Taking into account the facts that each model obtained comparatively high estimation accuracies and that the GR for models with many numbers of hidden units and learning iterations resulted in low values, the author considered that a model with few hidden units and learning iterations would be useful for constructing a high-generalizability model.

Relationship between pattern algorithms and simulation time

Figure 5 shows the relationship between pattern-recognition algorithms and simulation time. Ten trial results were summed for simulation time. The neural network model spent an enormous amount of simulation time with increases in the number of hidden units or learning iterations, even if the number of simulations was varied by computer specification. Especially, when the number of hidden units was 29 and the number of learning iterations was 10,000, the neural network in this study required about two weeks to perform ten simulations. Clearly, this situation becomes a problem when many simulations are expected to be carried out. As for classification-type models, each model except for NBTree could be executed in a comparatively short simulation time; moreover, even when the number of cross validations increases, the simulation time hardly increases at all. Thus, classification-type models are superior to function-type models from the viewpoint of simulation time, while function-type models have the advantage that their constraints such as input–output formats are less strict than those of classification-type models. For instance, when the output variables are continuous values, the classification models have to encode these to discrete values.

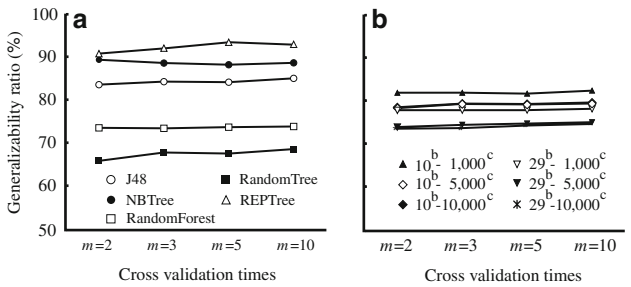

Fig. 4 Generalizability according to the different classification algorithms for **a** classification-type methods and **b** function-type methods

Conclusions

In the case without cross validation, when all samples were used as training data, RandomForest and RandomTree performed the best classification. Conversely, for constructing higher-generalizability models, J48, NBTree, and REPTree resulted in better estimates than the former algorithms, which resulted in low CR values.

Because neural network learning requires an enormous amount of calculation time for simulation, especially in combination with ensemble learning or cross validation or when dealing with numerous hidden units or performing many iterations, it is impossible to simulate some patterns given realistic restrictions on time and labor, although the specifications of the computer system certainly have an influence.

As for ensemble learning in this study, this technique was effectively verified, and Bagging was superior to AdaBoostM1 through all simulation results. In general, it is known that boosting is able to obtain a higher accuracy than bagging. However, bagging has advantageous characteristics such as higher robustness for noise compared with boosting (Quinlan 1996; Bauer and Kohavi 1999; Yasumura and Uehara 2005).

Without cross validation, the CR was shown to differ considerably for different pattern-recognition algorithms. However, these differences are diminished after cross validation. The CR_m through all simulation results for cases with cross validation and without ensemble learning ranged from 65.7 (m = 2, RandomTree) to 73.6 (m = 10, J48). From the viewpoint of generalizability, it is considered that

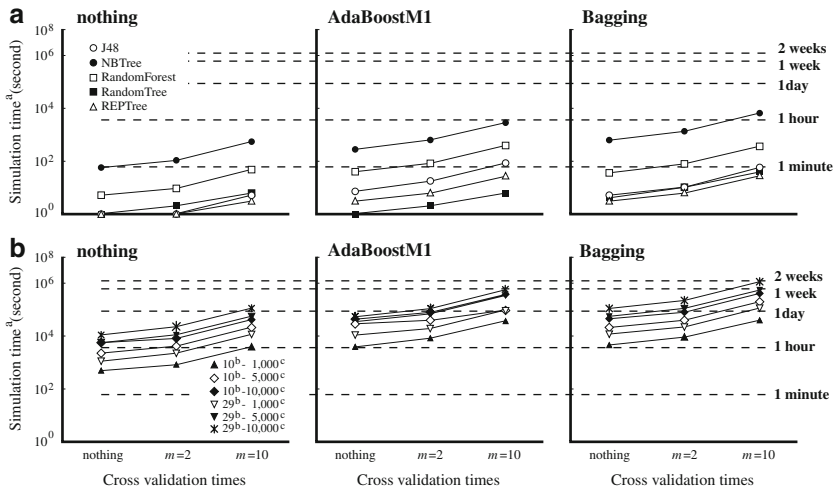

Fig. 5 Simulation time according to the difference in ensemble learning for **a** classification-type methods and **b** function-type methods Computer specification (Processor: 1.42 GHz PowerPC G4, Memory: 1 GB DDR SDRAM). [a]Sum of ten simulations [b]Number of hidden units [c]Number of learning iterations

we were able to construct models achieving a CR of around 70%.

In order to construct more effective models, we need to design some algorithms to apply to the model or to build in re-sampling methods such as ensemble learning and cross validation. Ensemble learning is one of the effective methods for constructing efficient models, particularly in the case of small sample datasets.

References

Aso H, Tsuda K, Murata N (2003) Statistics of pattern recognition and learning, new concepts and methods. Iwanami Shoten, Tokyo

Bare BB, Mendoza GA (1992) Timber harvest scheduling in a fuzzy decision environment. Can J For Res 22:423–428

Bauer E, Kohavi R (1999) An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach Learn 36:105–139

Breiman L (1996) Bagging predictors. Mach Learn 24:123–140

Breiman L (2001) Random forests. Mach Learn 45:5–32

Duda RO, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley, New York

Fisher DS, Steiner JL, Endale DM, Stuedemann JA, Schomberg HH, Franzluebbers AJ, Wilkinson SR (2000) The relationship of land use practices to surface water quality in the Upper Oconee watershed of Georgia. For Ecol Manage 128:39–48

Freund Y, Schapier RE (1997) A decision theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55:119–139

Fujino M, Yoshida M (2006) Development and validation of a method of forestry region classification using PCA and cluster analysis together with the SOM algorithm. J Jpn For Soc 88:221–230

Guan BT, Gertner G (1991) Modeling red pine tree survival with an artificial neural network. For Sci 37:1429–1440

Ichihara K, Toyokawa K, Sawaguchi I (1996) Runoff analysis in a basin where a forest road was constructed by a complex tank model whose optimum arrangement of tanks and parameters were identified by a genetic algorithm. J Jpn For Soc 78:134–142

Jin M (2004) R and discriminant analysis. In: ESTRELA, no 129. Statistical Information Institute for Consulting and Analysis, Tokyo, pp 61–67

Jin M (2005) Decision trees and ensemble learning. In: ESTRELA, no 133. Statistical Information Institute for Consulting and Analysis, Tokyo, pp 62–67

Jin M (2006) R and ensemble learning. In: ESTRELA, no 144. Statistical Information Institute for Consulting and Analysis, Tokyo, pp 64–69

Kivinen VP, Uusitalo J (2002) Applying fuzzy logic tree bucking control. For Sci 48:673–684

Kohavi R (1996) Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In: Proceedings of the 2nd international conference on knowledge discovery and data mining, Portland, pp 202–207

Liu C, Zhang L, Davis CJ, Solomon DS, Brann TB, Caldwell LE (2003) Comparison of neural networks and statistical methods in classification of ecological habitats using FIA data. For Sci 49:619–631

Minowa Y (1997) An analysis of subjective forest information with nonlinear engineering methods: selection of trees to be thinned using artificial neural network. J Jpn For Soc 79:143–149

Minowa Y (2005) Classification rules discovery from selected trees for thinning with the C4.5 machine learning system. J For Res 10:221–231

Minowa Y, Suzuki N, Tanaka K (2005a) Estimation of site indices with an artificial neural network. Jpn J For Plann 39:23–38

Minowa Y, Suzuki N, Tanaka K (2005b) Estimation of site indices with a machine learning system C4.5. Jpn J For Plann 39:143–156

Mitsuda Y, Yoshida S, Imada M (2001) Use of GIS-derived environmental factors in predicting site indices in Japanese larch plantations in Hokkaido. J For Res 6:87–93

Morimura H, Tone K, Iri M (eds) (1999) Encyclopedia of operations research and management science. Asakura Shoten, Tokyo

Ootaki A, Horie Y, Steinberg D (1998) Applied tree-based method by CART. Union of Japanese Scientists and Engineers (JUSE) Press, Tokyo

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Francisco

Quinlan JR (1996) Bagging, boosting, and C4.5. In: Proceedings of the 13th national conference on artificial intelligence, Portland, pp 725–730

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536

Suzuki M, Tatsuhara S, Ishihara T, Nagumo H (1995) Considerations of the thinning method for old sugi stands in the Tokyo University Forest in Chiba Prefecture. J Jpn For Soc 77:314–320

Tasaka T, Kumasakura Y, Uchiage S, Yano Y, Saito K, Shinoda T, Ueki S, Saito N (2001) Evaluation of method for the estimation of logging productivity using data-mining program C4.5. Bull Utsunomiya Univ For 37:177–186

Teraoka Y, Masutani T, Imada M (1991) Estimating site index of sugi and hinoki from topographical factors on maps for forest management. Sci Bull Fac Agr Kyushu Univ 45:125–133

Toraichi K (1993) Wavelets, fractals, and chaos with fluency analysis. In: Mathematical sciences, no 363. Saiensu-sha, Tokyo, pp 8–12

Witten IH, Frank E (2005) Data mining, practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco

Yamasaki H, Yoshimura T, Kanzaki K (1996) The selection of trees for thinning with the fuzzy reasoning model. J Jpn For Soc 78:143–149

Yasumura Y, Uehara K (2005) An ensemble learning method integrating bagging and boosting. In: Proceedings of the 19th annual meeting of the Japanese society for artificial intelligence, Kitakyushu