Abstract—Cross-validation and bootstrap, or resampling methods in general, are examined for applying them to optimizing the number of units of competitive associative nets called CAN2. There are a number of resampling methods available, but the performance depends on the neural network to be applied and functions to be learned. So, we apply several resampling methods to the CAN2 which has been shown effective in many areas so far. By means of numerical experiments, we have observed that a modified bootstrap method and the Lendasse’s bootstrap estimator work well for selecting the number of units. We also describe a new method and its performance for estimating generalization error via resampling methods.

I. INTRODUCTION

One of the most popular tasks of artificial neural networks is regression or function approximation, where the networks are used for predicting the target values corresponding to unknown input vectors after learning a training dataset of input vectors and target values. The generalization error or the performance of this task depends on the parameter values of neural networks. For selecting optimal parameter values, the resampling methods such as cross-validation and bootstrap for estimating generalization error are often utilized [1]-[5]. There are a lot of results reported and we are interested in the following results: the generalization error of $K$-fold cross-validation is less biased but has higher variance for larger $K$. Especially, leave-one-out cross-validation, or $n$-fold cross-validation is almost unbiased and has high variance, where $n$ is the number of examples in the given dataset. The bootstrap has downward bias but has a very low variance. However, the .632 and .632+ estimators are less biased and have a low variance. For 10 UCI real datasets, ten-fold cross-validation works better than .632 estimator [4], but the .632+ estimator has the best performance for a number of datasets including the above UCI datasets[3]. However, the .632 and .632+ estimators involve leave-one-out bootstrap procedures and they require huge computational cost for a large number of given training examples. Thus, reducing the computational cost is actually required [5].

On the other hand, the optimal number of units depends, in a complex way, on (1) the number of training examples, (2) the complexity of the function to be predicted, (3) the amount of noise in the training examples, (4) the architecture of the neural net, and so forth. Therefore, resampling methods to select the optimal number of units may also depend on them. From the point of view of (1), the bootstrap methods with the resample size $n$ may be unable to select the optimal number of units because the number of training examples for each resampled validation dataset use only 63.2% of all training examples (see bellow). From (2), however, the number of training examples may not have to be so large when the complexity of the function is low. The noise referred in (3) may be dealt by the neural net rather than the resampling methods, and the optimal number of the units of many neural networks may be set small for averaging the noise if the noise ratio is high, and it may become difficult to select the optimal number of units if the noise level is very high. The architecture mentioned in (4) is very important because the resampling methods will never succeed if the neural network to be validated is not a good predictor. Inversely, the performance of resampling methods may depend on the neural network to be examined.

So, we in this article try to clarify the properties of several resampling methods for the CAN2 and examine which method is better for the CAN2 and how precisely the number of units can be optimized. Note that the CAN2 is a neural network which has been introduced for utilizing the competitive and associative schemes [7], [8] to learning to perform piecewise linear approximation of nonlinear functions. The CAN2 has been shown effective in various areas such as rainfall estimation, function approximation, control, time-series prediction, and so on [9]-[16]. Especially, in the rainfall estimation contest held by the IEICE (Institute of Electronics, Information and Communication Engineers) of Japan[9], the estimation using the CAN2 with a online learning method has achieved the second least mean square error (MSE) in estimating a huge number of rainfall data. In function approximation problems, online learning methods for the CAN2 are shown to achieve better performance than BPNs (back-propagation nets), RBFNs (radial basis function nets) and SVRs (support vector regressions) [10], [12], [13]. Recently, we have developed a batch learning method for learning finite number of given training data efficiently, and its effectiveness has been shown in the following competitions; i.e., in the CATS benchmark prediction competition [14] held at IJCNN2004 (International Joint Conference on Neural Networks), the prediction using the CAN2 [15] has achieved the third least MSE for all prediction data among the 17 predictions selected from 24 submitted predictions. The batch learning CAN2 with cross-validation method has awarded the regression winner at the competition called Evaluating Predictive Uncertainty Challenge held at NIPS2004 (Neural Information Processing) [16], where there are three different datasets called stereopsis, gaze and outaouais, and our method using the CAN2 has achieved the 1st, 2nd and 1st least MSE for each dataset.
respectively.

In the following sections, after showing several resampling methods, we show and analyze the results of numerical experiments. By means of analysing the properties of resampling methods, we present a new method for estimating generalization error using resampling methods, and show the result of experiments. Finally we show concluding remarks.

II. RESAMPLING METHODS FOR FUNCTION APPROXIMATION

A. Mean squared error loss function

Let \( D^n = \{(x_j, y_j) \mid j = 1, 2, \cdots, n\} \) be a given training dataset consisting of \( n \) pairs of input vector \( x_j = (x_{j1}, x_{j2}, \cdots, x_{jk})^T \) and output value \( y_j \) of a system given by

\[
y_j = f(x_j) + d_j, \tag{1}
\]

where \( f(\cdot) \) is a nonlinear function, and \( d_j \) represents observation noise. Further, assume \( x_j \in D^n \) for \( j = 1, 2, \cdots, n \) are i.i.d. (independently and identically distributed) in the population. Let \( D^m \) be a dataset consisting of \( m \) data sampled from \( D^n \), and a predictor (a learning machine, or the CAN2 for us) with parameter values denoted by \( \theta \) learns \( D^m \) and approximates the target value \( y_j \) by \( \hat{y}_j = \hat{f}(x_j) = \hat{f}(x_j; D^m) \) corresponding to an input vector \( x_j \). Then, we define the mean squared error loss given by

\[
L(D^l; D^m, D^n) = \frac{1}{l} \sum_{j=1}^{l} \left\| y_j - \hat{f}(x_j; D^m) \right\|^2, \tag{2}
\]

where \( D^l = \{(x_{\rho(i)}, y_{\rho(i)}) \mid i = 1, 2, \cdots, l\} \) indicates a dataset sampled with a certain sequence \( \rho(i) \) from the population.

B. Training and Generalization Losses

When the validation dataset \( D^l \) is the same as the training dataset \( D^m \),

\[
L_{\text{train}}^{m,n} = L(D^m; D^m, D^n), \tag{3}
\]

called training (or empirical) loss. When \( D^l \) has no relation with \( D^n \) but the data in \( D^l \) are i.i.d. in the population,

\[
L_{\text{gen}}^{l,n} = L(D^l; D^n, D^n), \tag{4}
\]

called generalization (or prediction) loss.

C. K-fold Cross-Validation

K-fold cross-validation, a.k.a. V-fold or multifold cross-validation, is described as follows; let \( D_{j}^{n/K} \) for \( j = 1, 2, \cdots, K \) be the datasets called folds which partition \( D^n \) with almost the same size, where \( n/K \) is not always the integer but we use this expression for simplicity. Then, the loss is given by

\[
L_{CV}^{K} = \frac{1}{K} \sum_{j=1}^{K} L \left( D_{j}^{n/K}; D^n \setminus D_j^{n/K}, D^n \right), \tag{5}
\]

When \( K = n \) as a special case, K-fold cross-validation is called leave-one-out cross-validation whose loss, therefore, is given by

\[
L_{LOOCV}^{n} = L_{CV}^{n} = \frac{1}{n} \sum_{j=1}^{n} L \left( d_j; D^n \setminus d_j, D^n \right), \tag{6}
\]

where \( d_j = (x_j, y_j) \in D^n \).

D. .632 Estimator

The loss of leave-one-out bootstrap is given by

\[
L_{LOOBS}^{B} = \frac{1}{n} \sum_{j=1}^{n} \frac{1}{B} \sum_{b=1}^{B} L \left( d_j; (D^n \setminus d_j)_b^n, D^n \right), \tag{7}
\]

where \( (D^n \setminus d_j)_b^n \) is the \( b \)th bootstrap sample which is generated via resampling randomly with replacement from \( D^n \setminus d_j \). Since each \( d_j \) for \( j \neq i \) is included in \( (D^n \setminus d_j)_b^n \) with probability \( 1 - (1 - 1/n)^n \) which approximately is \( 1 - e^{-1} = .632 \) for large \( n \), the number of the distinct data in \( (D^n \setminus d_j)_b^n \) is \( .632n \). So, \( L_{LOOBS}^{B} \) is supposed to be biased upwards because many neural networks decreases the generalization error as distinct training examples increase. On the other hand, the training loss \( L_{\text{train}}^{n,n} \) is biased downwards because all validation data here are trained. So, the estimated generalization loss is supposed to be given by

\[
L_{.632} = \lambda L_{LOOBS}^{B} + (1 - \lambda)L_{\text{train}}^{n,n}, \tag{8}
\]

where \( \lambda \) is the optimism which approximately is \( 0.632 \) for large \( n \). As described in [4], this estimator does not work when the neural network is a perfect memorizer such as nearest neighbor classifier. So, the .632+ estimator has been introduced [3], however, the .632+ estimator also uses the leave-one-out scheme, so that the computational load becomes large as the number of training examples grows.

E. Lendasse’s Bootstrap Estimator

In order to reduce the computational cost, Lendasse [5] uses a slightly different version of .632 estimator as

\[
L_{LBS}^{B} = 0.632 \text{optimism}^B + .368 L_{\text{train}}^{n,n}, \tag{9}
\]

where

\[
\text{optimism}^B = \frac{1}{B} \sum_{b=1}^{B} \left( L(D^n; D_b^n, D^n) - L(D_b^n; D_b^n, D^n) \right). \tag{10}
\]

This loss \( L_{LBS}^{B} \) uses \( L(D^n; D_b^n, D^n) \) instead of time consuming leave-one-out method, but \( L_{LBS}^{B} \) might have downward bias than \( L_{.632} \) because \( L(D^n; D_b^n, D^n) \) involves the training loss. Note that the original version of \( L_{LBS}^{B} \) is given by \( \text{optimism}^B + L_{\text{train}}^{n,n} \) but it does not work so well in the experimental bellow, so we do not show its result in the following.
F. Modified Bootstrap for Model Selection

First, from the point of view of selecting optimal number of units, it is only necessary to select the number of units which achieves the minimum of the generalization loss. Namely, it does not matter whether the estimated generalization loss is biased or not when the bias is almost the same for all numbers of units.

So, we use the following bootstrap error loss for model selection although we know it is biased upwards;

$$L_{\text{MBS}}^{B,a} = \frac{1}{B} \sum_{b=1}^{B} L(D^n \setminus D_b^{n*}; X_b^{n*}, D^n). \quad (11)$$

Here, note that we are going to use $a$ bigger than 1 ($a = 2.3$ in the following experiments), so that the bootstrap sample size $m = an$ is for increasing the number of distinct training examples in $D_b^{n*}$. Namely, the number of distinct data is $(1 - e^{-a})n$ for large $n$, and 0.90$n$ for $a = 2.3$. Further, we use the validation dataset $D^n \setminus D_b^{n*}$ for estimating the upper bound of the generalization loss via excluding the training examples.

III. NUMERICAL EXPERIMENTS

We have examined the resampling methods for the CAN2 and the following two benchmark functions shown in [10], [12] (see Fig.1):

$$f_1(x) = \frac{3}{4} \exp \left( \frac{-(9x_1 - 2)^2 + (9x_2 - 2)^2}{4} \right)$$

$$+ \frac{3}{4} \exp \left( \frac{-(9x_1 + 1)^2 - (9x_2 + 1)^2}{49} \right)$$

$$+ \frac{1}{2} \exp \left( \frac{-(9x_1 - 7)^2 + (9x_2 - 3)^2}{10} \right)$$

$$- \frac{1}{5} \exp \left( -(9x_1 - 4)^2 - (9x_2 - 7)^2 \right),$$

$$f_2(x) = \begin{cases} 
1 & (x_2 - \xi > 1/2) \\
2(x_2 - \xi) & (0 \leq x_2 - \xi \leq 1/2) \\
\cos(4\pi r) + 1/2 & (r \leq 1/4) \\
0 & \text{(otherwise)} 
\end{cases}$$

$$\xi = 2.1x_1 - 0.1; \quad r = \sqrt{(\xi - 3/2)^2 + (x_2 - 1/2)^2}. \quad (12)$$

For a training dataset $D^n = \{(x_j, y_j) = f(x_j) + d_j) \mid j \in 1, 2, \cdots, n\}$, the vectors $x_j = (x_{j1}, x_{j2})$ were chosen uniformly and randomly from the unit square (see Fig.1). For training datasets, we have prepared $n = 1,000$ and $n = 5,000$ data, and the data with noise and without noise, where the noise injected were generated uniformly and randomly from $[-0.01, 0.01]$. We denote these training datasets by $D^n_{f_1, \text{noise}}$ and $D^n_{f_1, \text{noiseless}}$ for $n = 1000, 5000$ and $i = 1, 2$. For the validation dataset, we have generated $l = 10000$ data for $D^l = D^{10000}$ whose input vectors $x_j$ are on the 100 $\times$ 100 rectangular grids in the unit square (see Fig.1). We have calculated the following losses; the generalization loss $L_{\text{gen}}$, the leave-one-out cross-validation loss $L_{\text{LOOCV}}$, the Leave-one-out cross-validation loss $L_{\text{LOOCV}}$, the ‘‘MAE of selected number of units’’.

The numbers of units $N = 10^{i}$ for $i = 5, 6, \cdots$ have been searched by the resampling methods and the selected numbers are shown in TABLE I, while $L_{\text{LOOCV}}$, the leave-one-out cross-validation loss requires about $50 = n/B$ times of it (see Eq.(6) and Eq.(10)), where note that the most of the calculation time is used for learning training datasets. From TABLE I, all the methods have successfully selected the number of units within the maximum selection error of 70 units. Further, the “MAE of selected numbers” at the bottom row of TABLE I indicates the mean absolute error of the selected numbers of units of each method. From this point of view, we can say $L_{\text{MBS}}^{100}$ is the best, $L_{\text{BS}}^{100}$ the second, and $L_{\text{CV}}^{10}$ the third.

Next, we show two examples of the relation of the estimated error and the number of units in Fig.2, where we can see that $L_{\text{MBS}}^{100}$ is biased upward from $L_{\text{gen}}$ and the bias is almost uniform. However, from Fig.2(b), we can see that the bias of $L_{\text{MBS}}^{100}$ is not so uniform. We can also see that $L_{\text{gen}}^{100}$ does not change so smoothly, which is supposed to owing that $f_2$ is not so smooth (or exactly speaking, it is not $C_2$ continuous everywhere) and the number of training data is not so large, where note that the learning method of the CAN2 is derived with the supposition that the function to be learned is $C_2$ continuous [10]. Further, we can say the number of training examples, $n = 1,000$, is not so sufficient because Fig.1 is drawn with $1089 = 33 \times 33$ data which seem to be necessary.

<table>
<thead>
<tr>
<th>training dataset</th>
<th>$L_{\text{gen}}^{100}$</th>
<th>$L_{\text{CV}}^{10}$</th>
<th>$L_{\text{LOOCV}}^{100}$</th>
<th>$L_{\text{BS}}^{100}$</th>
<th>$L_{\text{MBS}}^{100}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{f_1, \text{noise}}^{1000}$</td>
<td>120</td>
<td>80</td>
<td>100</td>
<td>100</td>
<td>120</td>
</tr>
<tr>
<td>$D_{f_1, \text{noiseless}}^{1000}$</td>
<td>160</td>
<td>180</td>
<td>180</td>
<td>130</td>
<td>160</td>
</tr>
<tr>
<td>$D_{f_1, \text{noise}}^{5000}$</td>
<td>270</td>
<td>300</td>
<td>—</td>
<td>270</td>
<td>270</td>
</tr>
<tr>
<td>$D_{f_1, \text{noiseless}}^{5000}$</td>
<td>660</td>
<td>690</td>
<td>—</td>
<td>630</td>
<td>660</td>
</tr>
<tr>
<td>$D_{f_2, \text{noise}}^{1000}$</td>
<td>70</td>
<td>130</td>
<td>120</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>$D_{f_2, \text{noiseless}}^{1000}$</td>
<td>150</td>
<td>180</td>
<td>80</td>
<td>130</td>
<td>150</td>
</tr>
<tr>
<td>MAE of selected numbers</td>
<td>—</td>
<td>35</td>
<td>40</td>
<td>20</td>
<td>3</td>
</tr>
</tbody>
</table>
for figuring out the overall shape of the function. So, when
the number of training examples are small, the learning result
may change largely. Further, we can see that \( L_{100}^{\text{LBS}} \), may works
well for selecting the number of units because it is biased
uniformly, but the bias is downwards from \( L_{100}^{\text{gen}} \), which is
supposed to be owing that \( L_{100}^{\text{LBS}} \) involves the training error
although the parameters in Eq.(9) are supposed to be tuned
for reducing the bias.

IV. ESTIMATING GENERALIZATION ERROR

Even if the optimal number of units may be selected by
the bootstrap loss \( L_{MB}^{BBS} \) or \( L_{BS}^{BBS} \), we would like to estimate
the generalization error more precisely as the next step. So,
we examine the losses of resampling methods, and derives a
method for estimating generalization loss.

A. Properties of Generalization Error

When training and validation data are actually i.i.d in the
population, the losses of (leave-out) resampling methods are
supposed to be represented by the same form as

\[
L_{\text{valid}}^{\text{an,n}} = \frac{1}{K} \sum_{j=1}^{K} L \left( D^n \setminus D_j^{\text{an}}, D_j^{\text{an}}, D^n \right)
\]  

(13)

for \( 0 < \alpha < 1 \), where we use the number of distinctive
training vectors for bootstrap methods, namely the bootstrap
dataset \( D_j^{\text{an}} \) is replaced by \( D_j^{\text{an}} = D_j^{(1-\alpha)n} \). The above
form indicates that \( L_{\text{valid}} \) is the mean of \( L(d_i, D_{\text{an}}^{(n)}, D^n) \) for
\( d_i \in D^n \setminus D_{\text{an}}^{(n)} \). Since \( x_i \) of \( d_i = (x_i, y_i)^T \in D_{\text{an}}\setminus D_{\text{an}}^{(n)} \) are
i.i.d., the expected value of \( L_{\text{valid}}^{\text{an,n}} \) is supposed to be the same
as \( L_{\text{gen}}^{\text{an}} \). Thus, if the variance of \( L_{\text{valid}}^{\text{an,n}} \) is small, we may
be able to write \( L_{\text{gen}}^{\text{an,n}} \approx L_{\text{valid}}^{\text{an}} \) where note that we cannot
obtain \( L_{\text{valid}}^{\text{an,n}} \) from resampling methods because \( \alpha < 1 \). Next,
for \( \alpha < 1 \), it is expected that

\[
L_{\text{train}}^{\text{an,n}} \leq L_{\text{train}}^{\text{an}} \leq L_{\text{gen}}^{\text{an,n}} \approx L_{\text{valid}}^{\text{an,n}}
\]  

(14)

because it is supposed that the training error is smaller and
the validation error is larger for smaller training dataset, while
the generalization error involves both training and validation
error. Thus, we may be able to write the relation by

\[
L_{\text{gen}}^{\text{an,n}} = \gamma_1 (L_{\text{valid}}^{\text{an,n}} - L_{\text{train}}^{\text{an,n}}) + \gamma_2 L_{\text{train}}^{\text{an,n}}
\]  

(15)

where note that this form is the same as of .632 and .632+
estimators for leave-one-out bootstrap where the former uses
fixed parameter values as \( \gamma_1^{(an,n)} = .632 \) and \( \gamma_2^{(an,n)} = .368 \)
as shown above, the latter decides these parameters with
additional information[3]. Further, .632 methods are supposed
to be applicable only for the bootstrap resampling with \( \alpha = .632 \). Now, from an analogy of Lendasse’s optimism, we dare
use the following form

\[
L_{\text{gen}}^{\text{an,n}} \approx \gamma_1 (L_{\text{valid}}^{\text{an,n}} - L_{\text{train}}^{\text{an,n}}) + L_{\text{train}}^{\text{an,n}}.
\]  

(16)

When this equation holds for any \( n \), we have \( L_{\text{gen}}^{\text{an,n}} \approx \gamma_1 L_{\text{valid}}^{\text{an,n}} + L_{\text{train}}^{\text{an,n}} \), and then the estimation \( L_{\text{gen}}^{\text{an,n}} \) can be obtained by \( L_{\text{valid}}^{n-m,n-m} (\approx L_{\text{gen}}^{n-m,n-m}) \) for \( m \) smaller than \( n \). Thus, we have the estimation of \( \gamma_1 \) by

\[
\gamma_1 \approx \frac{L_{\text{valid}}^{n-m,n-m} - L_{\text{train}}^{n-m,n-m}}{L_{\text{valid}}^{an,n} - L_{\text{train}}^{an,n}}.
\]  

(17)

We suppose this equation holds for every number of units
of the CAN2 (see Fig.2), so that we had better use some
optimization method such as the least mean square for the
above estimation. Then, the estimation of \( L_{\text{gen}}^{\text{an,n}} \) can be obtained
from Eq.(16) by means of replacing \( \gamma_1 \) by \( \gamma_1 \).

B. A Method for Estimating Generalization Loss

From the above inspection, a method for estimating general-
ization loss is obtained as follows, where we are supposed
to obtain the generalization error for the numbers of units of the
CAN2 (or a predictor) for \( N_1, N_2, \cdots, N_s \);

1: Generate a dataset \( D^n \) from the given dataset \( D^n \), and
let \( D^{n-m} = D^n \setminus D^m \).

2: For every predictor with the number of units \( N_i \) (\( i = 1,2,\cdots,s \)), apply a resampling method with a certain
value of \( \alpha \) to the dataset \( D^n \), and obtain \( L_{\text{train}}^{an,m} \)
and \( L_{\text{valid}}^{an,m} \). Further, with the dataset \( D^{n-m} \), obtain
\( L_{\text{valid}}^{n-m,m} \).
method. From the above results, we can say that the estimation precision for each number of units validation is shown Fig. 5.

Other resampling methods can be also applicable, and a result using 2-fold cross-validation shows a good result, especially it changes smoothly with the change of the number of units. We have also presented a method for estimating generalization error. From the result of numerical experiments, the method seems to work pretty well on average, but the precision does not seem to be so high for selecting optimal number of units, except the modified bootstrap using big $\alpha$ ($= 2.3$).

V. CONCLUDING REMARKS

We have examined several resampling methods for selecting optimal number of units of the CAN2. By means of numerical experiments, we have observed that the best model selection (selecting the number of units of the CAN2) has been achieved by the modified bootstrap loss $L_{MBS}^B$, but it sometimes does not change smoothly with the change of the number of units. The Lendasse’s bootstrap error function $L_{LBS}^B$ shows a good result, especially it changes smoothly with the change of the number of units. We have also presented a method for estimating generalization error. From the result of numerical experiments, the method seems to work pretty well on average, but the precision does not seem to be so high for selecting optimal number of units.

In order to confirm the validity of the present method, we have to examine the method for many datasets as well as analyze the method mathematically or stochastically. Incidentally, Shao[17] has already introduced a modified bootstrap using variable sample size $m = \alpha n$ for linear model selection and
he has clarified the asymptotic consistency in model selection. So, we may be able to analyze asymptotic properties of the present method. Further, stochastic approach as the Bayesian learning approach which estimates predictive distribution is another possibility for analyzing the present method.

This work was partially supported by the Grant-in-Aid for Scientific Research (B) 16300070 of the Japanese Ministry of Education, Science, Sports and Culture.

REFERENCES