Learning of Neural Networks with GA-Based Instance Selection

Hisao Ishibuchi, Tomoharu Nakashima, and Manabu Nii
Department of Industrial Engineering, Osaka Prefecture University
Gakuen-cho 1-1, Sakai, Osaka 599-8531, JAPAN
E-mail {hisaoi, nakashi, manabu}ie.osakafu-u.ac.jp

Abstract

We examine the effect of instance and feature selection on the generalization ability of trained neural networks for pattern classification problems. Before the learning of neural networks, a genetic-algorithm-based instance and feature selection method is applied for reducing the size of training data. Nearest neighbor classification is used for evaluating the classification ability of subsets of training data in instance and feature selection. Neural networks are trained by the selected subset (i.e., reduced training data). In this paper, we first explain our GA-based instance and feature selection method. Then we examine the effect of instance and feature selection on the generalization ability of trained neural networks through computer simulations on various artificial and real-world pattern classification problems.

1. Introduction

Nearest neighbor classification [1] is one of the most well-known methods for pattern classification. In its standard formulation, all the given instances are used as a reference set for classifying new instances. Various approaches have been proposed for decreasing the size of the reference set [2, 3, 4]. Genetic algorithms have been also used for designing compact nearest neighbor classifiers with a small reference set. For example, in [9, 10], the tasks of genetic algorithms are to maximize the classification performance of selected patterns and to minimize the number of selected instances. Genetic algorithms can be also used for selecting salient features [6] and finding an appropriate weight of each feature [7].

In our former work [5], we proposed a genetic-algorithm-based approach to the design of compact reference sets with high classifiation ability by simultaneously selecting a small number of instances and features. In this paper, we examine the effect of the instance and feature selection on the generalization ability of trained neural networks. Before the learning of neural networks, the size of training patterns is reduced by genetic algorihtms. That is, genetic algorithms are used for editing the training data for the learning of neural networks. Then we use the selected instances and features in the learning.

2. Instance and Feature Selection

2.1. Coding

Let us assume that m labeled instances $\mathbf{x}_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, 2, \ldots, m$ are given from c classes in an n-dimensional pattern space where x_{pi} is the value of the i-th feature in the p-th instance. Genetic algorithms are used to select a small number of representative instances together with a few significant features for designing a compact nearest neighbor classifier with high classification performance. Let P_{ALL} be the set of the given m instances: $P_{ALL} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$. We also denote the set of the given n features as $F_{ALL} = \{f_1, f_2, \ldots, f_n\}$ where f_i is the label of the i-th feature. The task of genetic algorithms here is to select a small number of instances from P_{ALL} and to select only significant features from F_{ALL}. Let F and P be the set of selected features and the set of selected instances, respectively, where $F \subseteq F_{ALL}$ and $P \subseteq P_{ALL}$. We denote the reference set as $S = (F, P)$. In the standard formulation of nearest neighbor classification, the reference set S is specified as $S = (F_{ALL}, P_{ALL})$ because all the given features and instances are used for classifying new instances.

For handling our instance and feature selection problem by genetic algorithms, every reference set $S = (F, P)$ is coded by a binary string of the length $(n + m)$ as

$$S = a_1 a_2 \cdots a_n s_1 s_2 \cdots s_m, \tag{1}$$

where a_i denotes the inclusion ($a_i = 1$) or the exclusion ($a_i = 0$) of the i-th feature f_i, and s_p denotes the inclusion ($s_p = 1$) or the exclusion ($s_p = 0$) of the p-th instance $\mathbf{x}_p$. The feature set F and the instance set P are obtained by decoding the string S as follows:

$$F = \{f_i | a_i = 1, i = 1, 2, \ldots, n\}, \tag{2}$$

$$P = \{\mathbf{x}_p | s_p = 1, p = 1, 2, \ldots, m\}. \tag{3}$$

0-7803-7078-3/01/$10.00 (C)2001 IEEE.

2.2. Fitness Function

In the nearest neighbor classification with the reference set $S = (F, P)$, the nearest neighbor $\mathbf{x}_{p*}$ of a new instance $\mathbf{x}$ is found from the instance set P as

$$d_F(\mathbf{x}_{p*}, \mathbf{x}) = \min\{d_F(\mathbf{x}_p, \mathbf{x}) | \mathbf{x}_p \in P\}, \quad (4)$$

where $d_F(\mathbf{x}_p, \mathbf{x})$ is the distance between $\mathbf{x}_p$ and $\mathbf{x}$, which is defined by the feature set F as

$$d_F(\mathbf{x}_p, \mathbf{x}) = \sqrt{\sum_{i \in F} (x_{pi} - x_i)^2}. \quad (5)$$

The new instance $\mathbf{x}$ is classified by its nearest neighbor $\mathbf{x}_{p*}$. If the class of the nearest neighbor $\mathbf{x}_{p*}$ is the same as that of the new instance $\mathbf{x}$, $\mathbf{x}$ is correctly classified. Otherwise the new instance $\mathbf{x}$ is misclassified.

In the instance and feature selection in this paper, the number of selected instances and the number of selected features are to be minimized, and the classification performance of the reference set $S = (F, P)$ is to be maximized. Thus our problem is written as follows:

$$\begin{aligned} &\text{Minimize } |F|, \text{ minimize } |P|, \\ &\text{and maximize } Perf(S), \end{aligned} \quad (6)$$

where $|F|$ is the number of features in F (i.e., the cardinality of F), $|P|$ is the number of instances in P, and $Perf(S)$ is a performance measure of the reference set $S = (F, P)$. The performance measure is defined based on the classification results of the given m instances by the reference set $S = (F, P)$.

In our former work [5], we defined the performance measure $Perf(S)$ by the number of correctly classified instances by $S = (F, P)$. Each instance $\mathbf{x}_q$ $(q = 1, 2, \ldots, m)$ was classified by its nearest neighbor $\mathbf{x}_{p*}$, which is defined as

$$d_F(\mathbf{x}_{p*}, \mathbf{x}_q) \min\{d_F(\mathbf{x}_p, \mathbf{x}_q) | \mathbf{x}_p \in P\}. \quad (7)$$

In the definition of the performance measure in [9, 10] for the instance selection, when an instance $\mathbf{x}_q$ was included in the reference set, $\mathbf{x}_q$ was not selected as its own nearest neighbor. In the context of the instance and feature selection, this means that the nearest neighbor $\mathbf{x}_{p*}$ of $\mathbf{x}_q$ is selected as follows:

$$d_F(\mathbf{x}_{p*}, \mathbf{x}_q) = \begin{cases} \min\{d_F(\mathbf{x}_p, \mathbf{x}_q) | \mathbf{x}_p \in P\}. \\ \qquad\qquad \text{if } \mathbf{x}_q \notin P, \\ \min\{d_f(|\mathbf{x}_p, \mathbf{x}_q) | \mathbf{x}_p \in P - \{\mathbf{x}_q\}\}, \\ \qquad\qquad \text{if } \mathbf{x}_q \in P. \end{cases} \quad (8)$$

That is, $Perf(S)$ is the number of correctly classified instances when the nearest neighbor of each instance is defined by (8). We use this performance measeure in our instance and feature selection problem. In our former work [5], we showed that the generalization ability is higher when we use this performance measure than when we use a simple performance measure where the nearest neighbor $\mathbf{x}_{p*}$ to an instance $\mathbf{x}_q$ $(q = 1, 2, \ldots, m)$ is defined as in (7).

The fitness value of the reference set $S = (F, P)$ is defined by the three objectives of our instance and feature selection problem in (6) as

$$\begin{aligned} &fitness(S) \\ &= W_{Perf} \cdot Perf(S) - W_F \cdot |F| - W_P \cdot |P|, \end{aligned} \quad (9)$$

where W_{Perf}, W_F, and W_P are user definable non-negative weights.

2.3. Basic Algorithm

We use a genetic algorithm to maximize the fitness function in (9) by appropriately selecting instances and features. Our genetic algorithm is similar to [9, 10] where only the instance selection was discussed. In our genetic algorithm, first a number of binary strings (say, N_{pop} strings) of the length $(n + m)$ are randomly generated to form an initial population. Next a pair of strings are randomly selected from the current population to generate two strings by crossover and mutation. The selection, crossover, and mutation are iterated to generate N_{pop} strings. The newly generated N_{pop} strings are added to the current population to form an enlarged population of the size $2 \cdot N_{\text{pop}}$. The next population is constructed by selecting the best N_{pop} strings from the enlarged population. The population update is iterated until a pre-specified stopping condition is satisfied. Our genetic algorithm is written as follows:

[GA-based Instance and Feature Selection]

Step 1 (Initialization): Randomly generate N_{pop} strings of the length $(n + m)$.

Step 2 (Genetic Operations): Iterate the following procedures $N_{\text{pop}}/2$ times to generate N_{pop} strings.

1. Randomly select a pair of strings from the current population.
2. Apply a crossover operation to the selected pair of strings to generate two offspring. In computer simulations of this paper, we use the uniform crossover to avoid the dependency of the performance on the order of n features and m instances in the string.

3. Apply a mutation operation to each bit value of the two stings generated by the crossover operation. The mutation operation changes the bit value from 1 to 0 or from 0 to 1. For the muatation on the substring for instance selection, we bias the mutation probability. That is, we use different probability for the bit change from 1 to 0 and from 0 to 1. The number of selected instances can be efficiently decreased by the biased mutatation.

Step 3 (Generation Update): Add the newly generated N_{pop} strings in Step 2 to the current population of the N_{pop} strings to form an enlarged population of the size $2 \cdot N_{\text{pop}}$. Select the N_{pop} best strings with the largest fitness values from the enlarged population to form the next population.

Step 4 (Termination Test): If a pre-specified stopping condition is not satisfied, return to Step 2. Otherwise decode the best string with the largest fitness value to generate a reference set as shown in (2) and (3). In our computer simulations, we use the total number of generations (i.e., iterations of Step 2 and Step 3) as the stopping condition of our genetic algorithm.

Our genetic algorithm is different from the standard implementation [11] in the selection and generation update procedures. In our algorithm, the selection of parent strings for the crossover is performed randomly. The selection of good strings is actually performed in the generation update procedure. In this sense, the generation update procedure of our genetic algorithm can be viewed as a selection procedure for generating a mating pool from which parent strings are randomly selected. We adopted this implementation according to the first attempt of the application of genetic algorithms to the instance selection in [9, 10]. We also examined a more standard implementation based on the roulette wheel selection with the linear scaling and a single elite strategy. Simulation results of these two implementations were almost the same. So we only report simulation results by the above implementation.

2.4. Numerical example

In this section, we show the result of the instance and feature selection by our genetic-algorithm-based approach. We apply the genetic algorithm to artificial two-dimensional data. We generated 30 instances for each class in a unit square $[0, 1] \times [0, 1]$.

Our genetic algorithm was appled to the two-dimensional data for selecting a small number of instances and features. The following parameter specifications were used:

Population size: 50,
Crossover probability: 1.0,
Mutation probability: 0.01,
Stopping condition: 1000 generations,
$(W_{Perf}, W_F, W_P) = (5, 1, 1)$.

Our genetic algorithm selected six instances (three for each class) and two features (i.e., no feature was removed). The selected instances are shown in Fig.1 along with the classification boundary generated by them. From Fig.1, we can see that all training data are correctly classified.

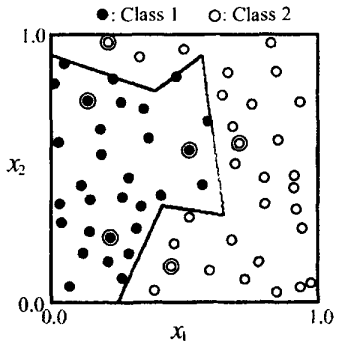

Figure 1. Selected patterns.

3. Learning of Neural Networks

Ishibuchi et al. [5] demonstrated that the instance and feature selection can improve the generalization ability of nearest neighbor classifiers. In this section, we use selected reference sets as training data for the learning of neural networks. The purpose of this paper is to examine the effect of the instance and feature selection described in the previous section on the generalization ability of the neural networks. We use standard three-layered neural networks. In this paper we use the back-propagation algorithm to the learning of the neural networks.

When we do not reduce the size of the training data, the number of input units is specified as n where n is the number of attributes in the given data set. All the m instances are used in the backpropagation algorithm. On the other hand, when we use the reduced training data by our GA-based instance and feature selection method in Section 2, the number of input units n_I of neural networks is smaller than n. That is, we use neural networks with simpler architecture. The number of instances is also smaller than m. Thus the instance and feature selection decreases both the complexity of neural networks and the size of training data. Its straightforward positive effect on the learning of neural networks using less instances leads to less computation time than the

case of the entire training data. The instance and feature selection may have two different effects on the learning of neural networks. One is a negative effect by decreasing the number of training instances. In general, it is difficult to obtain good generalization by the learning of neural networks from a small number of training instances. The other is a positive effect by removing outliers or noisy instances. Such instances deteriorate the generalization ability of trained neural networks. Another merit of the instance and feature selection is to speed up the learning of neural networks.

It is not clear whether the learning of simpler neural networks using less instances leads to higher generalization ability than the case of the entire training data. In the next section, we examine the effect of the instance and feature selection on the generalization ability of trained neural networks through computer simulations.

4. Computer Simulations

We used two artificial data sets and four real-life data sets. In our computer simulations, we applied our GA-based method to each data set after normalizing attribute values to real numbers in the unit interval [0,1]. Each data set is briefly described in the following.

Data Set I with Small Overlap: We generated a two-class pattern classification problem in the unit square $[0,1] \times [0,1]$. For each class, we generated 50 instances using the normal distribution $N(\mu_k, \sum_k)$ where μ_k and $\sum_k$ were specified as follows:

$$\mu_1 = (0,1), \mu_2 = (1,0), \quad \sum_1 = \sum_2 = \begin{pmatrix} 0.4^2 & 0 \\ 0 & 0.4^2 \end{pmatrix}. \tag{10}$$

Data Set II with Large Overlap: We generated a two-class pattern classification problem in the same manner as in the above data set using larger variances. We specified the normal distribution of each class as follows:

$$\mu_1 = (0,1), \mu_2 = (1,0), \quad \sum_1 = \sum_2 = \begin{pmatrix} 0.6^2 & 0 \\ 0 & 0.6^2 \end{pmatrix}. \tag{11}$$

Iris Data: The iris data consist of 150 instances with four features from three classes (50 instances from each class).

Appendicitis Data: The appendicitis data consist of 106 instances with eight features from two classes. Since one feature has some missing values, we used seven features as in [13].

Cancer Data: The cancer data consist of 286 instances with nine features from two classes. This data set was also used in [13].

Wine Data: The wine data consist of 178 instances with 13 features from three classes, which is available from the machine learning database in UC Irvine.

In our computer simulations, we generated 100 training data and 1000 test data according to the normal distributions of Data I and Data II. For the other data sets, we divided each data set into two groups: 2/3 for training data, and 1/3 for test data. Our GA-based instance and feature selection method was applied to the training data for selecting a compact reference set with a small number of instances and features. The selected instances and features were then used to train neural networks. The test data were classified by the trained neural networks for evaluating its generalization ability. These procedures were iterated 30 times using different partitions into training data and test data for calculating the average classification rate on the test data.

In Table 1 we show simulation results of our GA-based instance and feature selection method over 30 trials for each data set. This table shows the average number of the selected instances and features and the performance on test data by the nearest neighbor classifier with the selected instances and features. The number of the training data for each data set is shown in parentheses in this table.

Table 1. Simulation results of instance and feature selection by the GA-based method.

Data set	Features	Instances
Data set I	2.0 (2)	7.1 (100)
Data set II	1.7 (2)	12.9 (100)
Iris	2.5 (4)	6.2 (100)
Appendicitis	2.4 (7)	4.7 (71)
Cancer	3.7 (9)	22.5 (191)
Wine	5.6 (13)	6.4(119)

We used the reduced raining data to train neural networks. The BP (back-propagation) algorithm [12] was iterated 5000 times (i.e., 5000 epochs). The learning rate and the momentum rate are 0.1 and 0.9, respectively. The number of hidden units is five for the iris and the appendicitis data, 15 for the cancer data, and two for the wine data.

Table 2 shows average simulation results. From Table 2, we can see that the generalization ability of neural networks was improved by using the reduced training data for the appendicitis data and the cancer data. However, the generalization ability was not improved for the iris data and the wine data. It is well-known that the iris data and the wine data are not challenging as classification problems. That is, there are only small overlaps between classes in the iris data and the wine data. On the other hand, in the appendicitis data and the cancer data, two classes are overlapping with each other.

In that case, higher generalization is obtained by using simple classification boundaries than by using complex classification boundaries.

Table 2. Simulation results of neural networks using the original and reduced data sets.

Data set	Original	Reduced
Data set I	70.2%	70.0%
Data set II	61.2%	64.3%
iris	94.5%	93.2%
appe	83.9%	88.4%
cancer	68.8%	73.8%
wine	97.7%	95.8%

In our computer simulations, simple classification boundaries were produced by trained neural networks with the redued training data. To clearly demonstrate this, we show the classification boundaries made by trained neural networks in Fig. 2 and Fig. 3. In Fig. 2, we show the classification boundaries by the neural networks by using the original training data (i.e., before the instance and feature selection). On the other hand, the classification boundaries by the reduced training data were shown in Fig. 3. From Fig. 2 and Fig. 3, we can see that simpler classification boundaries can be obtained by using the reduced training data set, and overfitting is ovservbed by using the original data set. From these simulation results, we can say that instance and feature selection prevented the overfitting of neural networks to training data.

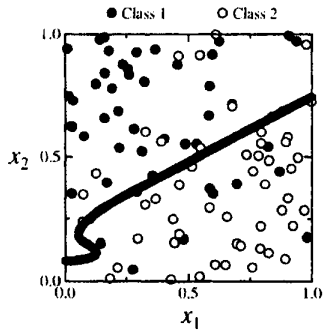

Figure 2. Classification boundaries by using the original data set.

To examine how the generalization ability of the trained neural networks improves during the course of the learning, we monitored the classification rates of the trained neural networks for training data and test data. Fig. 4 and Fig. 5 show the classification rates of the trained neural networks by using the original training data and the reduced training data, respectively. From Fig. 4 and Fig. 5, we can see that the generalization

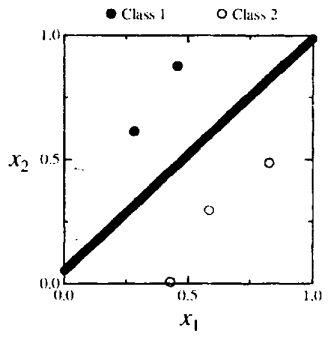

Figure 3. Classification boundaries by using the reduced data set.

ability of the trained neural networks first increased in the beggining of the learning in the case of both the original training data and the reduced training data. However, the generalization ability decreased after a certain number of iteration of the learning in the case of the original training data. On the contrary, we did not observe any degradation of the generalization ability in the case of the reduced training data. From Fig. 4 and Fig. 5, we can see that the neural networks were likely to be overfit to the training data when we did not use the reduced training data.

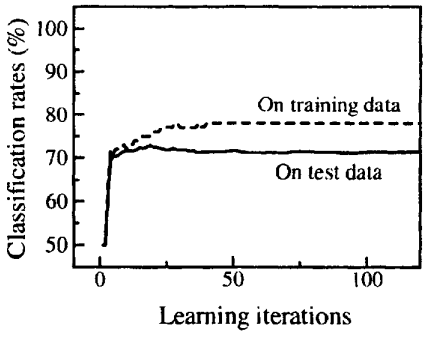

Figure 4. Classification rates by using the original training data.

Another positive effect of using the reduced training data is the decrease in the learning time of neural networks. In Table 3, we show the average CPU time for training the neural networks with the original training data and the reduced training data. From this table, we can see that the CPU time for the learning was drastically decreased by the reduction of training data.

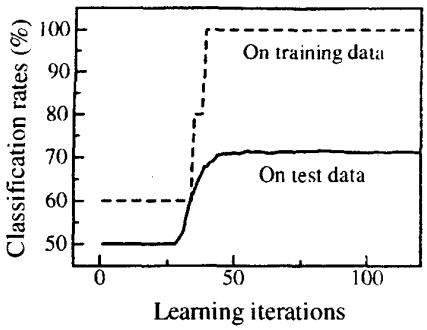

Figure 5. Classification rates by using the reduced data set.

Table 3. CPU time. (sec.)

Data set	Original	Reduced
Data set I	3.6	0.3
Data set II	3.6	0.4
Iris	3.8	0.2
Appendicitis	2.8	0.2
Cancer	16.4	1.6
Wine	3.3	0.2

5. Conclusions

In this paper, we examined the effect of the instane and feature selection on the generalization ability of trained neural networks. We used a genetic-algorithm-based method for selecting a small number of instances and features. The reduced training data with a small number of instances and features were used for the learning of neural networks by the backpropagation algorithm.

Through the computer simulations on artificial and real-world data sets, we showed that the generalization ability of trained neural networks were improved for difficult problems with large overlaps. On the other hand, the generalization ability was not improved for simple problems with small overlaps. Instance and feature selection prevented the overfitting of neural networks to training data on almost all data sets in our computer simulations.

We also showed another positive effect of using reduced training data. The learning time of neural networks was shorter when we use the reduced training data than when the original training data were used.

References

[1] T. M. Cover, and P. E. Hart, "Nearest neighbor pattern classification", *IEEE Transaction of Information Theory*, Vol. 13, 1967, pp. 21-27.

[2] P. Hart, "The condensed nearest neighbor rule", *IEEE Trans. on Information Theory*, Vol. 14, 1968, pp.515-516.

[3] B. V. Dasarathy, "Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 24, 1994, pp. 511-517.

[4] D. Chaudhuri, C. A. Murthy, and B. B. Chaudhuri, "Finding a subset of representative points in a data set", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 24, 1994, pp. 1416-1424.

[5] H. Ishibuchi, T. Nakashima, and M. Nii, "Genetic-algorithm-based instance and feature selection", *Instance Selection and Construction for Data Mining*, Norwell, MA, Kluwer Academic Publishers, 2001, pp. 95-112.

[6] W. Siedlecki, and J. Sklansky, "A note on genetic algorithms for large-scale feature selection", *Pattern Recognition Letters*, Vol. 10, 1989, pp. 335-347.

[7] J. D. Kelly Jr., and L. Davis, "Hybridizing the genetic algorithms and the k nearest neighbors classification algorithm", *Proc. of 4th International Conference on Genetic Algorithms*, 1991, pp. 377-383.

[8] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 2, 1972, pp. 408-420.

[9] L. I. Kuncheva, "Editing for the k-nearest neighbors rule by a genetic algorithm", *Pattern Recognition Letters*, Vol. 16, 1995, pp. 809-814.

[10] L. I. Kuncheva, "Fitness functions in editing k-NN reference set by genetic algorithms", *Pattern Recognition*, Vol. 30, 1997, pp. 1041-1049.

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA, Addison-Wesley, 1989.

[12] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing*, MIT Press, Cambridge, USA, 1986.

[13] S. M. Weiss, and C. A. Kulikowski, *Computer Systems that Learn*, San Mateo, CA, Morgan Kaufmann, 1991.