J Inf Process Syst, Vol.13, No.5, pp.1345~1357, October 2017
https://doi.org/10.3745/JIPS.04.0045

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Nearest Neighbor Based Prototype Classification Preserving Class Regions

Doosung Hwang* and Daewon Kim**

Abstract

A prototype selection method chooses a small set of training points from a whole set of class data. As the data size increases, the selected prototypes play a significant role in covering class regions and learning a discriminate rule. This paper discusses the methods for selecting prototypes in a classification framework. We formulate a prototype selection problem into a set covering optimization problem in which the sets are composed with distance metric and predefined classes. The formulation of our problem makes us draw attention only to prototypes per class, not considering the other class points. A training point becomes a prototype by checking the number of neighbors and whether it is preselected. In this setting, we propose a greedy algorithm which chooses the most relevant points for preserving the class dominant regions. The proposed method is simple to implement, does not have parameters to adapt, and achieves better or comparable results on both artificial and real-world problems.

Keywords

Class Prototype, Dissimilarity, Greedy Method, Nearest-Neighbor Rule, Set Cover Optimization

1. Introduction

In pattern classification, the concept of dissimilarity is widely taken up as a simple and intuitive metric for assessing how different two data points are. The classes of two points are in the same class if the dissimilarity of two patterns is small or in the different classes otherwise. In classification, prototypes are a number of representatives per class from a prepared set of training data, but might be much more informative for training and predicting. A prototype classification is based on the dissimilarity among pre-collected points. The class of a test point is assigned to that of its nearest prototype. Nearest neighbor rule is one of the simplest but most effective method in building a prototype selection method [1,2]. Practically speaking, the learning capability of a classification algorithm is not increased when the amount of training data is insufficient against its model complexity. As the size of training data increases, the learning complexity of a nearest neighbor algorithm increases especially in view of learning time. Moreover, a training data might contain spurious data that is redundant or noisy, increasing the model complexity.

A prototype selection method has been studied to select a handful of informative points from the

Manuscript received October 6, 2016, first revision February 15, 2017; accepted May 18, 2017.
Corresponding Author: Daewon Kim (drdwkim@dku.edu)
* Dept. of Software, Dankook University, Yongin, Korea (dshwang@dankook.ac.kr)
** Dept. of Applied Computer Engineering, Dankook University, Yongin, Korea (drdwkim@dku.edu)

training set so that all training points are not necessary for classification and spurious data do not harm the learning capability [3-5]. A prototype selection makes an original problem a new learning task with a relatively small set of selected data. A prototype classification framework takes the two steps. In the first step, a prototype selection method is applied to select a handful of prototypes per class. We assume that the set of selected points reflect the distribution of the original data or it is enough for learning classification rules with the intended classifier. Secondly the classifier exploits discrimination rules from the new simple task in pursuit of low storage requirement, low computational cost, and equivalent generalization compared to those results of the original problem [4,5]. When the number of training data is too large, the conciseness of training data plays an important role for interpreting a predicted result. The selected prototype helps to shorten the learning time and on why a given prediction has been made [6]. Therefore, prototype learning takes advantages of data sparsity and fast learning as well as prediction explanation without degrading generalization performance. Several prototype selection methods have been reviewed with editing prototypes [7,8], sphere cover prototype [6,9-13], and boundary prototypes [14,15]. Learning vector quantization (LVQ), k-means, and k-memoir are the most widely used prototype methods which edit prototypes [1]. Restricted coulomb energy (RCE) [9,10], randomized sphere cover (RSC) [12] and set covering finding [11,16] select prototypes that cover class region by considering data hyper spheres and their classes. Another prototype classifier try to find a set of border or component patterns that are used in constituting a discriminate rule. Gaussian mixture model (GMM) [1] models the class-conditional densities as a Gaussian mixture and a class decision rule of support vector machine (SVM) depends on border patterns [14,15]. A prototype classifier suffers from some drawbacks: the unpredictable number of prototypes and the sensitivity to data order. The appropriate number of prototypes has to be determined by re-running the algorithm several times along with the evaluation of selected prototypes. A prototype selection method is sensitive to the presenting order of data points. Each run selects a different set of prototypes. In this work, we present a new deviation, based on sphere cover prototype, which chooses the most relevant points for preserving the class discrimination regions. The proposed method always selects a set of prototypes, including all the training data, and does not require any predefined parameters. This paper is organized as follows. In Section 2, some works related to prototype selection methods are discussed. Section 3 proposes our method for prototype selection and Section 4 presents the experimental results that examine the effectiveness of the proposed approach and compares it to a nearest-neighbor algorithm. Finally, in Section 5, we summarize our work and give the directions for future work.

2. Related Background

The RCE network is a classification algorithm using a set of region covering prototypes into which the feature space is divided [9]. The algorithm selects a subset of training points to cover regions of different classes in the training phase. A training point becomes a new prototype if it is not chosen and its covering region doesn't have the instances of other classes. The training procedure stops and outputs the set of prototypes where their covering regions contain all the training points. A radius of a prototype is controlled between preselected minimum and maximum values in the training process. Depending on the preselected minimum radius, the new prototype might include other class points. The greedy sphere covering (GSC) algorithm extends the RCE network in view of adapting the radii of

prototypes and minimizing the number of prototypes [10]. The choice of prototypes is based on a greedy method using the largest number of uncovered points in the hope of minimizing the size of prototypes so that the union of the chosen sphere regions covers all the training examples. The α randomized sphere cover (αRSC) classifier computes the radius of a prototype in a similar way to the GSC method, but controls the number of instances covered by a potential prototype with parameter α [12]. In addition, the parameter α has a smoothing effect around the decision boundary by filtering out noises or excluding outliers. The GSC and αRSC methods do not have the training points of other classes in constituting sphere covering regions. The set covering machine (SCM) uses data-dependent balls to select prototypes from the training examples and constructs a Boolean function that is the smallest conjunction (or disjunction) of positive prototypes (or negative prototypes) [16]. The center instances of selected balls become prototypes and are used for a Boolean classification rule. The radius of a prototype is controlled with a very small value so that the ball of a prototype doesn't include the instances of other class. Since the SCM is applicable only to 2-class tasks, we must adopt the algorithm in one-versus-one or one-versus-all fashion for k-class problems (k>2). In [17], a class cover problem is formulated by finding a small number of sphere prototypes covering all the training examples in one class without having other class examples. Based on a class cover problem, the class cover catch digraph (CCCD) of instance x is defined by including the homogeneous instances within the minimum distance to other class instances but centered at x [13]. The CCCDs of all training points divide the feature space into covering regions and are represented by a directed graph. The prototype selection follows a greedy method to find a near-minimal dominating set. The set of prototypes forms a CCCD classifier using a nearest neighbor rule that scales distances by the calculated radii. In [6], they adapt a prototype selection problem to a set covering problem in optimization theory. Depending on a pre-chosen radius, the feature space is divided into sphere covering regions centered at training points. The sphere of a training point allows for having or not having the other class points. Based on a set covering optimization, the algorithm finds a feasible solution that partially covers the training points. Their algorithm has the following properties: covering as many training data of the same class as possible, covering as few other class data as possible, and minimizing a number of prototypes [6,17]. They embody these properties to a set covering optimization and propose a greedy algorithm for solving the constrained optimization problem. The set of sphere covering regions makes up of the set of prototypes by minimizing the number of prototypes and the number of other class points within covering spheres among prototypes. We summarize the discussed studies with three factors: how to define a sphere covering region of prototype, how to compute the radii of prototypes, and how to select prototypes. A sphere covering region depends on the inclusion of other class points and the radius of a training point is easily estimated through the nearest neighbor rule. The RCE network and the prototype selection with a fixed radius allow sphere covering regions to include training points with other classes. The αRSC and GSC, SCM, and CCCD approaches do not allows for including other class points in building sphere covering regions. The reviewed methods select prototypes in brute-force or greedy strategy [18]. The RCE, αRSC and the CCCD approach randomly choose a potential prototype, estimate its radius and, in brute-force manner, consider if it becomes a prototype or not. A greedy approach is preferable for the GSC, SCM and the prototype selection with a fixed radius. The sphere covering algorithm we propose get motivated by the prototype selection with a fixed radius, but differs in that a sphere covering region does not allow for the embodiment of training points with other classes and controls the radius of a prototype. This change in constituting sphere covering region takes advantages of self-

adapting radii of training points, class-wise prototype selection step and the direct design of our greedy algorithm.

3. Prototype Selection Method

We consider a prototype selection method as a set covering problem from a given classification problem. Every training example can be a prototype covering the partial area of its own class but not covering the points of different classes. The prototype selection method finds the smallest number of class covering points from one class, which leads to take advantages of storage and computational requirement in practice. In this setting, a set of class prototypes should capture the whole structure of class distribution so that the generalization error should close or equal to that of the original problem. In a typical classification problem, we are given a training set consisting of pre-categorized points. Suppose that we have a classification problem $X=\{(x_i, y_i)|i = 1, \dots, n\}$ with $x_i \in R^d$ and their class labels $y_i \in \{1, \dots, c\}$ and the data of c class sets is described by $X_l = \{(x_i, l)|i = 1, \dots, n_l\}$ where the size of X_l is $n_l = |X_l|$ and $\sum_{l=1}^{c} n_l = n$. The output of a prototype selection method is a prototype set $P = \{P_l|l = 1, \dots, c\}$ where $P_l \subseteq X_l$ for class l and $|P_l| \ll |X_l|$. We begin with the notation of neighborhood by considering a sphere *S(x)* centered at a training data (*x, y*). A desirable prototype of class l constitutes a sphere which covers as many training examples of the same class as possible, and does not include training examples of other classes. The covering area of a prototype is decided with its own radius that is computed by the class labels of nearest neighbors. The radius of x is adopted from the margin concept of SVM, reminding that the optimal boundary of two points with different classes is a midpoint between them. Given a classification problem X and any distance metric $d(\cdot,\cdot)$ in a vector space R^d, we constitute the sphere $S(x)$ for any $(x, y) \in X$ with radius γ_x.

$$S(x) = \{z|d(x, z) \leq \gamma_x\}$$

Here, the radius γ_x of x is computed by the followings:

$$d(x, z_1) \leq d(x, z_1) \leq \cdots \leq d(x, z_n)$$

$$i_1 = \text{argmax}_{i,l(x)=l(z_i)} d(x, z_i)$$

$$i_2 = \text{argmin}_{i,l(x)\neq l(z_i)} d(x, z_i)$$

$$\gamma_x = d(x, z_{mid}), \text{ where } z_{mid}=\frac{z_{i_1} + z_{i_2}}{2}$$

Thus $S(x)$ constitutes a subset of training points with the same class that are covered within γ_x. The goal of our method is to find the smallest subset of prototypes $P \subset X$ such that P can cover X: every $x_i \in X$ is included within γ_{x_j} of some point x_j in P. The problem becomes an integer problem by introducing indicator variables $\propto = (\propto_1, \propto_2, \cdots, \propto_n)$ for each $x_i \in X$.

$$\propto_i = \begin{cases} 1, & \text{if } x_i \in P \\ 0, & \text{otherwise} \end{cases}$$

Therefore, $\sum_{j,x_i \in S(x_j)} \alpha_j$ is the number of occurrences that x_i is covered by $S(x_j)$. Requiring that this sum be positive for each $x_i \in X$ enforces that P induces a cover of X. Our problem is equivalent to the following integer problem.

$$\begin{matrix} \text{Min}_{\alpha} & \sum_{j=1}^{n} \alpha_j \\ \text{s.t} & \sum_{j,x_i \in S(x_j)} \alpha_j \geq 1, \forall\, x_i \in X \end{matrix} \tag{1}$$

A feasible solution of (1) is one of a set covering problem which chooses a subset of class data points. Set covering problem is NP-hard in the strong sense and there is no polynomial time approximation method unless *NP* = *P*[18]. Since $S(x_j)$ doesn't include the data points of other classes, the problem of (1) can be solved independently for each class $l \in \{1, \dots, c\}$. Thus, the problem leads to c separate integer sub-problems. Let $\alpha_j^l \in \{0,1\}$ indicate whether x_j^l is chosen in P_l to be a prototype for class l. By covering a training space with selected prototypes, the sum $\sum_{j,x_i^l \in S(x_j^l)} \alpha_j^l$ counts the number of spheres in which x_i^l is included. We then solve the following integer problem for class l.

$$\begin{matrix} \text{Min}_{\alpha^l} & \sum_{j=1}^{n_l} \propto_j^l \\ \text{s.t} & \sum_{j=1}^{n_l} \sum_{i=1}^{n_l} \alpha_j^l 1(\, x_i^l \in S(x_j^l)) \geq 1, \end{matrix} \tag{2}$$

where, $1(b)$ is 1 if b is true and 0 otherwise. The solution of (2) is to find a minimal set of sphere covering prototypes. The normal way of its solution requires a linear problem (LP) solver like GLPK package [19], but we have checked that the use of an LP solver can be slow and memory-intensive for large data. Thus, our method is based on greedy approach that has been applied for solving set covering problems [18]. The proposed method is deterministic and computationally simpler than the case of an LP. In our greedy method, a prototype is selected with the ratio of cost to the number of points that newly covered. The solution of (2) is easily found by selecting the larger size of $S(x_j^l)$ and, simultaneously, maximizing the number of uncovered points in $x_i^l \in X^l$. After computing all the class spheres for $x \in X$, we try to solve the solution of $X^l, l = 1,2, \dots, c$. For $x_i^l \in X^l$, x_i^l is added to P_l if it has the maximum number of uncovered points in X^l. The step from $\{P_1, \dots, P_l, \dots, P_c\}$ to $\{P_1, \dots, P_l \cup \{x_i^l\}, \dots, P_c\}$ is proceeded by maximizing $\Delta\text{obj}(x_j^l)$.

$$\Delta\text{obj}(x_j^l) = |X^l \cap S(x_j^l) \backslash \cup_{x_i^l \in P_l} S(x_i^l)| \tag{3}$$

If the number of $S(x_j)$ is relatively small, x_j is positioned around boundary or within class overlapped region. We manage to control those with parameter τ. This parameter has an effect on controlling the number of prototypes and removing the prototypes of small sphere on the decision boundary. The proposed algorithm is presented in Algorithm 1.

Algorithm 1. Select the set of class prototypes $P_l, l = 1,2, \dots, c$

SelectPrototype(*X, S, c,* τ)
$P = \phi$
for *l=1* to *c* do
$P_l = \phi$
while $\Delta\text{obj}(x_j^l) > 0$ and $S(x_j^l) > \tau$ do
j = $\text{argmin}_{x_i^l \in X_l} \Delta\text{obj}(x_j^l)$
$P_l = P_l \cup \{x_j^l\}$
for $x_j^l \in X_l$ do
$S(x_i^l) = S(x_i^l) \backslash \{x_j^l\}$
end for
end while
$P = P \cup P_l$
end for
return *P*

If the prediction rule follows the 1-nearest neighbor, the class of test point x is predicted with the class of the nearest prototype.

$$\hat{y} = \text{argmin}_l \text{min}_{z \epsilon P_l} d(x, z)$$

The nearest prototype z provides a clue that can explain why a given prediction has been made.

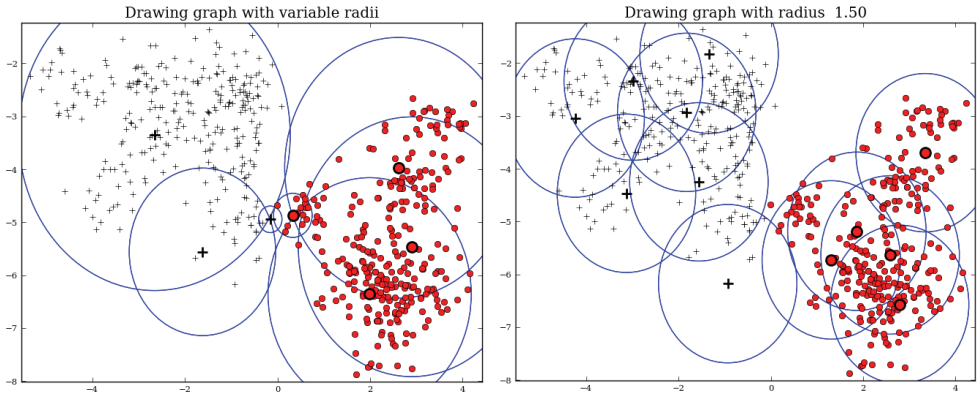

Fig. 1. The examples of greedy algorithms for a 2-class toy problem. (a) A case of Prop with $\tau = 0$. (b) A case of the Fixed with $\gamma = 1.5$.

Fig. 1 shows the illustrative examples of the proposed algorithm (Prop), and the prototype classifier with a fixed radius (Fixed) [6] in 2-dimensional cases. The toy problem is artificially generated with 2 classes. In Fig. 1, (a) shows a case of Prop with $\tau = 0$ while (b) is that of the Fixed with $\gamma = 1.5$. The

chosen prototypes are marked in bold circle or cross point with their covered spheres. The Prop method selects 7 prototypes, not including other class points, and the Fixed selects 19 prototypes.

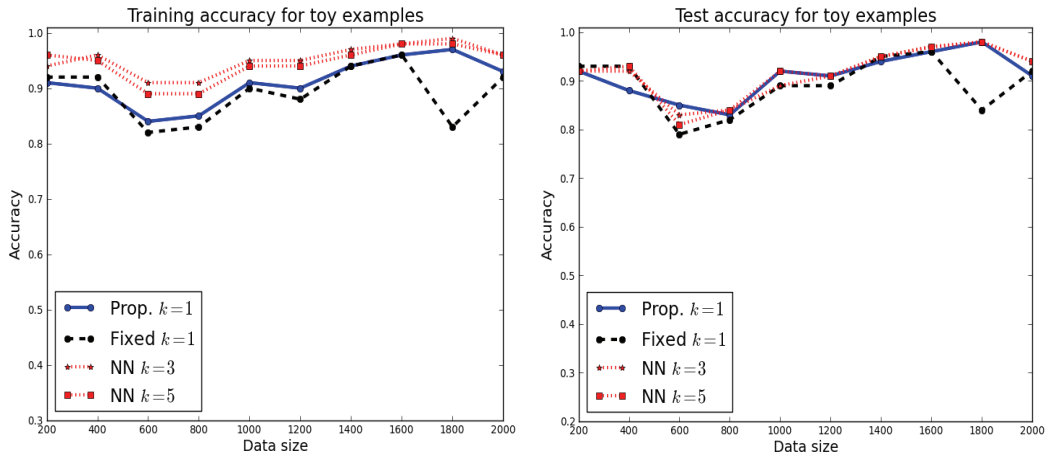

Fig. 2. The comparison of greedy algorithms for the 2-class toy problems. (a) Training accuracy. (b) Testing accuracy.

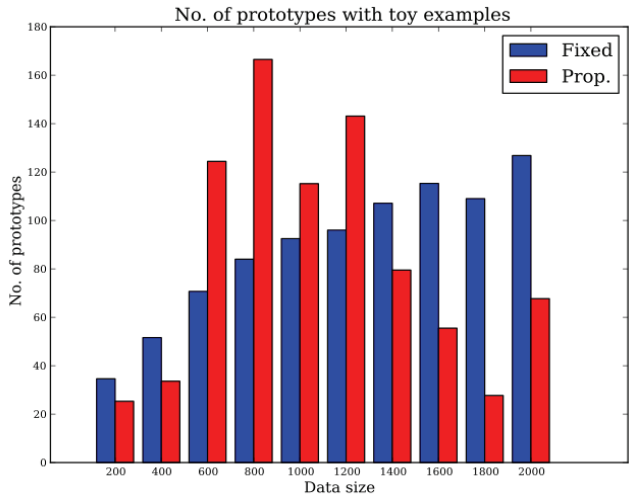

Fig. 3. The comparison of greedy algorithms for the number of prototypes.

Figs. 2 and 3 compares the two methods of Prop and Fixed in terms of generalization capability and prototype number. The training data was artificially generated for 2-class classification problems varying the sizes from 200 to 2000. The separable complexity of a problem depends on about 10% to 30% overlapped points among two classes. The *k*-nearest neighbor algorithm was used for comparison. Each trial was run 5 times by a stratified 10-fold cross validation along with k = 3, 5. The average accuracy and the number of prototypes are shown in Figs. 2 and 3. The Fixed algorithm tends to depend on the predefined radius and the selected prototype tends to embed other class points. Also, it couldn't select the prototypes covering all the training points. Our method Prop selects the prototypes

with all points by self-adapting the radii. It tends that the radii of boundary prototypes is much smaller than those of within-class prototypes. The simulation results are compared to those of the nearest neighbor without prototype selection.

Fig. 2(a) shows that the training accuracy of Prop is between that of the nearest neighbor and that of Fixed. While the training accuracy of the nearest neighbor is higher than those of the prototype methods, the testing results of all three approaches are similar in Fig. 2(b). The standard nearest neighbor shows a large difference in training and testing. The use of a prototype selection method can reduce a gap in training and testing. The number of prototypes varies due to the data distribution along with learning complexity and the way to constitute class spheres in Fig. 3.

4. Experimental Results

In order to assess the performance of the proposed algorithm Prop and Bien's algorithm Fixed, we run them on about twenty problems available from the UCI Machine Learning Repository [20] and the KEEL data set repository [21]. There are 3 problems only with nominal data, one problem of nominal and numeric data, and the other problems only with numeric values. The simulation results are compared to those of k-nearest-neighbor (k = 3, 5). The experiments of the nearest-neighbor classifier retain all the training points as prototypes. In Table 1, the main characteristics of the chosen problems are summarized in terms of task name (Problem), data size (No), data type (numerical [Num]) and nominal [Nom]), the total number of attributes (Attr), and the number of classes (Cls). We adopt a stratified 10-fold cross validation for each task and select 2/3 of each data set for training and use the remainders for testing. The Euclidean distance is chosen for dissimilarity metric. Each problem is compared by use of accuracy and prototype rate. Tables 2 and 3 report the training and testing accuracy of the experiments. In Table 3, the radius r of Fixed is chosen for maximizing the accuracy of the testing set and r is stepped by 0.2 from the shortest distance to the longest distance.

Table 1. The benchmark problems for the empirical evaluation

Problem	No	Attr	Num	Nom	Cls	Problem	No	Attr	Num	Nom	Cls
Abalone	4177	8	8	0	3	Ionosphere	351	34	34	0	2
Adult a1a	1605	123	0	123	2	Liver disorder	345	6	6	0	2
Australian	690	7	7	0	2	Mammographic	961	5	2	3	2
Breast cancer	683	9	0	9	2	Pendigits	7494	16	16	0	10
Car evaluation	1728	6	0	6	4	Segment	2310	19	19	0	7
Diabetes (Pima)	768	8	8	0	2	Svmguide1	3089	4	4	0	2
DNA	2000	180	0	180	3	Vehicle	846	18	18	0	4
Fourclass	862	2	2	0	2	Vowel	990	13	13	0	11
Glass	214	9	9	0	7	Wine	178	13	13	0	3
Heart	270	13	13	0	2	W1a	2477	300	0	300	2

Table 2. The 10-fold cross-validation accuracy in training

Problem	3-NN		5-NN		Fixed			Prop	
	Mean	SD	Mean	SD	Mean	SD	r	Mean	SD
Abalone	0.74	0.01	0.68	0.01	0.49	0.02	1.0	0.84	0.00
Adult a1a	0.89	0.01	0.87	0.01	0.67	0.13	0.3	0.81	0.01
Australian	0.90	0.00	0.88	0.01	0.81	0.02	0.5	0.85	0.01
Breast cancer	0.98	0.00	0.98	0.00	0.95	0.02	0.3	0.96	0.00
Car evaluation	0.99	0.00	0.99	0.00	0.84	0.01	1.2	0.86	0.01
Diabetes (Pima)	0.90	0.01	0.80	0.01	0.69	0.01	0.4	0.78	0.02
DNA	0.88	0.01	0.86	0.02	0.60	0.02	0.3	0.93	0.00
Fourclass	1.00	0.00	1.00	0.00	0.72	0.07	1.0	0.75	0.06
Glass	0.81	0.02	0.76	0.02	0.56	0.01	0.3	0.83	0.02
Heart	0.89	0.01	0.84	0.02	0.71	0.08	1.0	0.81	0.02
Ionosphere	0.91	0.01	0.87	0.01	0.79	0.07	0.7	0.81	0.06
Liver disorder	0.83	0.02	0.77	0.02	0.58	0.03	0.2	0.86	0.02
Mammographic	0.91	0.00	0.88	0.01	0.81	0.01	0.5	0.85	0.01
Pendigits	1.00	0.00	1.00	0.00	0.98	0.00	1.0	0.97	0.00
Segment	0.98	0.00	0.97	0.00	0.92	0.00	3.0	0.93	0.01
Svmguide1	0.98	0.00	0.97	0.00	0.94	0.01	0.5	0.94	0.01
Vehicle	0.85	0.01	0.82	0.01	0.65	0.02	0.1	0.86	0.00
Vowel	0.99	0.00	0.97	0.01	0.74	0.03	0.3	0.87	0.03
Wine	0.98	0.01	0.97	0.01	0.92	0.03	1.0	0.96	0.01
W1a	0.98	0.00	0.98	0.00	0.96	0.03	1.5	0.96	0.02

Table 3. The 10-fold cross-validation accuracy in testing

Problem	3-NN		5-NN		Fixed		Prop	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Abalone	0.74	0.01	0.68	0.01	0.48	0.02	0.84	0.01
Adult a1a	0.89	0.01	0.87	0.01	0.67	0.12	0.80	0.02
Australian	0.83	0.03	0.85	0.03	0.81	0.03	0.85	0.03
Breast cancer	0.96	0.01	0.97	0.02	0.95	0.02	0.96	0.02
Car evaluation	0.96	0.01	0.98	0.01	0.84	0.02	0.86	0.02
Diabetes (Pima)	0.73	0.04	0.73	0.03	0.71	0.04	0.79	0.02
DNA	0.76	0.02	0.78	0.02	0.60	0.04	0.93	0.00
Fourclass	1.00	0.00	1.00	0.00	0.72	0.07	0.75	0.07
Glass	0.66	0.07	0.64	0.06	0.56	0.10	0.83	0.05
Heart	0.78	0.05	0.81	0.05	0.72	0.09	0.81	0.05
Ionosphere	0.85	0.03	0.85	0.04	0.81	0.08	0.81	0.56
Liver disorder	0.62	0.04	0.61	0.05	0.60	0.04	0.85	0.05
Mammographic	0.83	0.02	0.84	0.02	0.86	0.02	0.85	0.02
Pendigits	1.00	0.00	1.00	0.00	0.98	0.00	0.97	0.01
Segment	0.96	0.01	0.95	0.01	0.92	0.01	0.93	0.01
Svmguide1	0.96	0.01	0.96	0.01	0.94	0.01	0.94	0.01
Vehicle	0.70	0.03	0.70	0.03	0.65	0.04	0.86	0.03
Vowel	0.95	0.03	0.90	0.03	0.75	0.03	0.86	0.04
Wine	0.96	0.03	0.96	0.03	0.92	0.05	0.96	0.03
W1a	0.98	0.00	0.98	0.00	0.96	0.04	0.96	0.02

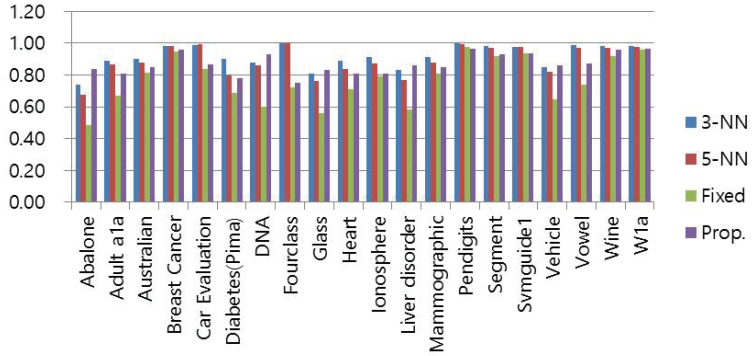

Fig. 4. The training accuracy comparison of Prop. vs. Fixed.

In Fig. 4, the training accuracy results of Prop are between 3-nn and 5-nn. The test result of Fig. 5 shows that our method Prop is competitive or better than *k*-nearest neighbor but with significant reduction in prototypes. No single method is the best for all the data sets due to the inherent data distribution. The Prop algorithm largely depends on data distribution, class overlapping, and boundary points against the Fixed algorithm. The method Fixed uses a predefined radius and select the subset of prototypes which covers all the training points.

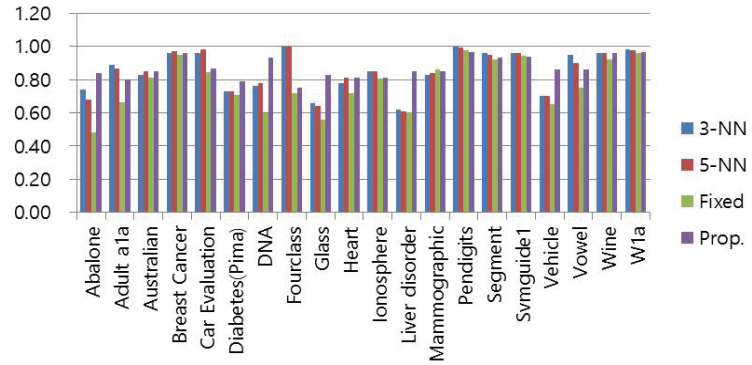

Fig. 5. The testing accuracy comparison of Prop. vs. Fixed.

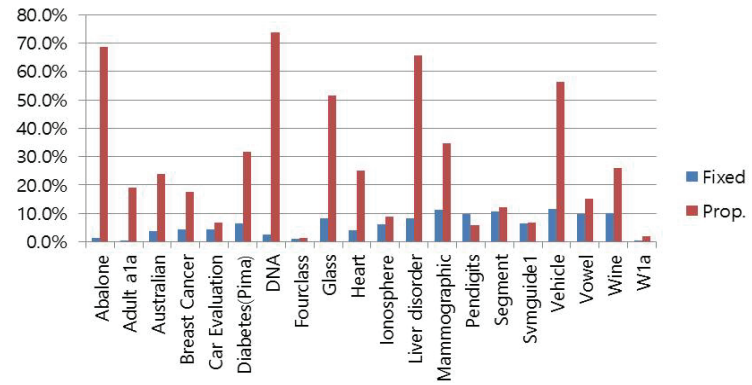

Fig. 6. The comparison in the number of prototypes.

In comparison of the number of prototypes, the results of Fixed are smaller than those of Prop because the prototype of Prop constitutes a sphere-covering region which can include as many points within the same class as possible. In Fig. 6, the prototype rates of Fixed are under 13.0% but those of Prop are widely divergent. However, in Fig. 5, the testing results of Prop are much higher than those of the nearest neighbor in Abalone, Australian, Pima, DNA, Glass, Liver disorder, and Vehicle. We analyze that our method is preserving class regions along with the data distribution of a problem. In addition, the large reduction rate cannot be scalable to the generalization ability of an original problem.

5. Conclusions

We have presented a prototype reduction procedure by selecting representative samples from a data set. The proposed method provides a way to summarize a data set in pursuits of fast learning and equable generalization capability when compared to the non-prototype based results of the problems. Purposefully, we defined our notion of a desired prototype set and reformulated the problem as a minimum set cover problem. A set cover problem has been known as one of NP-complete problem, but there had better design an approximate algorithm. We designated the greedy algorithm that can be implemented in ease and simplicity, but it can find a full set cover solution, not partial cover set one. Our method chooses a suitable number of prototypes for each class through adapting the radii of plausible prototypes automatically. The self-adapting way of prototype radii constitutes the same class points of a prototype by nearest neighbor search. Any selected prototypes can summarize a large set of training points and can enhance the explanation ability of test data in such practical applications as gene data analysis, disease prediction, bank credit analysis etc. This feature will be useful to domain experts for exploiting a set of concise knowledge in mining a large data set. The proposed data reduction method is applicable for any classification and has the expandability to any distance metrics.

Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.20170010550012002, Next Generation EDR Technology for Effective Response of Intelligent APT Attacks).

References

[1] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer, 2001.

[2] X. Wu and V. Kumar, *The Top Ten Algorithms in Data Mining*. Boca Raton, FL: CRC Press, 2009.

[3] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257-286, 2000.

[4] J. A. Olvera-Lopez, J. A. Carrasco-Ochoa, J. F. Martinez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133-143, 2010.

[5] S. Garcia, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417-435, 2012.

[6] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, vol. 5, no. 4, pp. 2403-2424, 2011.

[7] H. A. Fayed, S. R. Hashem, and A. F. Atiya, "Self-generating prototypes for pattern classification," *Pattern Recognition*, vol. 40, no. 5, pp. 1498-1509, 2007.

[8] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *IEEE Transactions on Systems, Man, and Cybernetics Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 86-100, 2012.

[9] M. J. Hudak, "RCE classifiers: theory and practice," *Cybernetics and Systems*, vol. 23, no. 5, pp. 483-515, 1992.

[10] J. Wang, P. Neskovic, and L. N. Cooper, "Learning class regions by sphere covering," Brown University, Providence, RI, *IBNS Technical Report 2006-02*, 2006.

[11] I. Takigawa, M. Kudo, and A. Nakamura, "Convex sets as prototypes for classifying patterns," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 1, pp. 101-108, 2009.

[12] R. Younsi and A. Bagnall, "A randomized sphere cover classifier," in *Proceedings of the 11th International Conference on Intelligent Data Engineering and Automated Learning*, Paisley, UK, 2010, pp. 234-241.

[13] D. Marchette, "Class cover catch digraphs," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 2, pp. 171-177, 2010.

[14] F. Angiulli, "Fast nearest neighbor condensation for large data sets classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, pp. 1450-1464, 2007.

[15] D. Hwang and D. Kim, "Near-boundary data selection for fast support vector machines," *Malaysian Journal of Computer Science*, vol. 25, no. 1, pp. 23-37, 2013.

[16] M. Marchand and J. S. Taylor, "The set covering machine," *Journal of Machine Learning*, vol. 3, pp. 723-746, 2003.

[17] A. H. Cannon and L. J. Cowen, "Approximation algorithms for the class cover problem," *Annals of Mathematics and Artificial Intelligence*, vol. 40, no. 3-4, pp. 215-223, 2004.

[18] V. V. Vazirani, *Approximation Algorithms*. New York, NY: Springer, 2001.

[19] The GLU Linear Programming Kit Package [Online]. Available: https://www.gnu.org/software/glpk/.

[20] UCI Machine Learning Repository [Online]. Available: http://archive.ics.uci.edu/ml/.

[21] KEEL-Dataset Repository [Online]. Available: http://sci2s.ugr.es/keel/datasets.php.

Doosung Hwang

He received a M.S. (1990) from Chungnam National University, Daejeon, Korea and Ph.D. (2003) in Computer Science from Wayne State University, Detroit, MI, USA. He worked as a senior researcher for ETRI (Electronics and Telecommunications Research Institute), Korea (1991–1998). He is currently a professor in Department of Software at Dankook University, Korea. His research interests include data mining, parallel processing and database design.

Daewon Kim

He received a M.S. (1996) from the University of Southern California, Los Angeles, CA, USA and Ph.D. (2002) in Electrical and Computer Engineering from Iowa State University, Ames, IA, USA. He worked as a senior researcher at Samsung Electronics Co. Ltd., Suwon, Korea (2002–2004). He is currently a professor in Department of Applied Computer Engineering at Dankook University, Korea. His research interests include signal processing, mobile applications, and nondestructive evaluation.