NEAR-BOUNDARY DATA SELECTION FOR FAST SUPPORT VECTOR MACHINES

Doosung Hwang[1], ***Daewon Kim***[2]
[1]Department of Computer Science
Dankook University, Cheonan-Si, South Korea. Email: dshwang@dankook.ac.kr
[2]Department of Multimedia Engineering
Dankook University, Cheonan-Si, South Korea. Email: dr.dwkim@gmail.com

ABSTRACT

Support Vector Machines(SVMs) have become more popular than other algorithms for pattern classification. The learning phase of a SVM involves exploring the subset of informative training examples (i.e. support vectors) that makes up a decision boundary. Those support vectors tend to lie close to the learned boundary. In view of nearest neighbor property, the neighbors of a support vector become more heterogeneous than those of a non-support vector. In this paper, we propose a data selection method that is based on the geometrical analysis of the relationship between nearest neighbors and boundary examples. With real-world problems, we evaluate the proposed data selection method in terms of generalization performance, data reduction rate, training time and the number of support vectors. The results show that the proposed method achieves a drastic reduction of both training data size and training time without significant impairment to generalization performance compared to the standard SVM.

Keywords: *Support Vector Machine, Nearest Neighbor Rule, Tomek Link, Data Selection.*

1.0 INTRODUCTION

Classification algorithm is to find an inductive rule from pre-classified data, which can predict the class of an example without known class as one among several classes. As one of the popular classification algorithms, Support Vector Machine(SVM) has gained wide acceptance due to its solid statistical foundation and reported good generalization performance in many applications [1, 2]. Training a SVM necessitates solving a constraint quadratic problem that requires large memory and large training time proportional to the number of training data. The efficiency of SVM depends on the number of support vectors during both the training and prediction phases. It has been analytically studied that as the number of training examples increase, the number of support vectors also increase [3]. The computational burden of standard SVM solvers is very largely determined by the amount of memory required to store the active part of the kernel matrix. The training time increases dramatically due to the repeated kernel computations if the memory requirement exceeds the available memory. The number of support vectors imposes a penalty on the prediction phase because the computation of the decision function requires a time proportional to the number of support vectors. Thus, the time complexity of SVM training and testing becomes a real challenge when the training size is made up of thousands of data or more [4].

In applying a SVM, the training process has $O(n^3)$ time and $O(n^2)$ space complexities, where n is the size of training dataset. These complexities may become very critical as the number of training data increases. On the other hand, the decision function of SVM depends only on a small set of support vectors. If we select a subset of training data which includes latent support vectors, we will model the same decision function by solving the quadratic problem for the reduced set of examples. The mentioned complexity depends on how to select training examples that are likely to be support vectors.

Data selection techniques for SVM learning have been studied for making the number of training examples controllable for intending not to decrease generalization capability. The key concern of this approach is that a fraction of training data might become a decision-boundary consistent set:- the decision boundary learned by the selected data is alike to that by the entire training data [5]. The time complexity of a SVM can be shortened by the reduced training set from the given training data under consideration of a little loss of generalization performance with the entire set of training data. These data selection methods are based on sampling, nearest neighbor rule, and clustering. Sampling based methods are adapted to reduce the training set, but cannot provide a way to select the training points around the

decision boundary [6,7,8].
Nearest neighbor property provides a hint on more informative examples by using the fact that training examples around a decision boundary tends to have more heterogeneous neighbors in their class labels. The SMOTE (Synthetic Minority Over-sampling Technique) rule adds synthetic data to the minority class according to nearest neighbor rule [9]. The study reported that the SMOTE+SVM outperforms the sampling+SVM in the various tests [6]. The Neighbor Property based Pattern Selection(NPPS) algorithm chooses only the samples around separation boundary by defining neighbor entropy and neighbor match based on *k*-nearest neighbor algorithm[10]. In applying the NPPS, the number of selected examples depends on *k*, entropy, and matched ratio. The FCNN(Fast Condensed Nearest Neighbor) computes a training set consistent subset from the original set using nearest neighbor property [11]. This algorithm uses the selection criterion that is guided by the decision boundary. Through various experiments, the FCNN+SVM is well scalable on large problems and outperforms the results of other condensed nearest neighbor algorithms in terms of data reduction and accuracy [5].

The confidence measure and Tomek link are used to select boundary points for SVM learning. The confidence measure of an example counts the number of homogeneous data within a sphere that is as large as possible [12]. If the confidence measure of an example is large, it is not likely to be around a boundary. The Tomek link often exists around the convex hull of the two classes. The optimal hyperplane bisects the line segment of positive and negative examples in a Tomek link. These data selection algorithms also use nearest neighbor property and can control the number of selected data.

The combination of edge detection and clustering method reduces a training size. An edge detection method points out the set of boundary examples and the *k*-means clustering algorithm adds more examples to preserve the original distribution of a problem [13]. Like the NPPS, the number of selected examples is controlled by *k*.

In this paper, we propose a data selection technique for SVM. The proposed technique is motivated by the two characteristics of SVM:- kernel mapping and large margin classifier. Through kernel methods, a SVM detects hidden regularities in the selected feature space and thus a data selection technique can be modeled in the feature space, not in the input space. The training phase of large margin classifier decides a small subset of training samples that only play significant role in constructing a decision function. Heterogeneous class samples tend to be placed around the decision hyperplane. These samples are selected through analyzing the neighborhood of a training data. Therefore, like the previous studies, we use nearest neighbor algorithm in order to provide an efficient way to reduce the training set by filtering out meaningless patterns.

The rest of this paper is organized as follows. In Section 2, we give a brief overview of kernel methods, large margin classifiers and implementations of SVM. Section 3 presents the proposed method that selects the subset of training examples for training a SVM. In Section 4, we report the experimental results on several real-world datasets. The results are compared to those of SVM and NPPS in terms of missclassification rate, data reduction rate, training time and the number of support vectors. Concluding remarks are provided in Section 5.

2.0 RELATED BACKGROUND

This section briefly discusses the mathematical background of kernel methods and large margin classifiers to implement a SVM for classification. With no loss of generality, we only consider a two-class classification problem. Assume that a classification problem $P = \{(\mathrm{x}^{(i)}, y_i) \mid \mathrm{x} \in R^d,\ y_i = \pm 1,\ i = 1, 2, \cdots, n\}$. Each $\mathrm{X}^{(i)}$ exists over input space R^d and its class is a positive example if $y_i = +1$ and a negative example if $y_i = -1$.

2.1 Kernel Methods

Kernel methods are used to map training examples into new high-dimensional feature data so that the set of feature data is used for finding regularities that are not detectable in the original space. A classifier is applied to find a set of classifying rules from the mapped examples in the feature space. This approach lessens the learning complexity of a classification task and make the simple classifiers more practical in learning predictive rules [13]. This is the key principle of SVM learning in the case that a training set is not linearly separable. We transform the problem *P* into the new problem $P^{new} = \{(\Phi(\mathrm{x}^{(i)}), y_i) \mid \Phi(\mathrm{x}^{(i)}) \in R^d,\ y_i = \pm 1,\ i = 1, 2, \cdots, n\}$ by $D\,(D >> d)$ dimensional feature mapping Φ.

$$\Phi : \mathrm{x} = (x_1, x_2, \cdots, x_d)^t \rightarrow \Phi(\mathrm{x}) = (\phi_1(\mathrm{x}), \phi_2(\mathrm{x}), \cdots, \phi_D(\mathrm{x}))^t \quad (1)$$

Here, $\Phi(\mathrm{x})$ is the corresponding vector in R^d and $\phi_j(\mathrm{x})$ is an arbitrary linear or nonlinear function from the input space to the higher dimensional feature space. The $\Phi(\mathrm{x})$ makes the input vectors congregated more compactly in the feature space but the computation cost is expensive. Under mapping Φ, the norm distance between $\mathrm{x}^{(i)}$ and $\mathrm{x}^{(j)}$ denotes as

$$d(\Phi(\mathrm{x}^{(i)}), \Phi(\mathrm{x}^{(j)})) = \| \Phi(\mathrm{x}^{(i)}) - \Phi(\mathrm{x}^{(j)}) \|. \quad (2)$$

A kernel function K of $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ is the inner product $\Phi(\mathbf{x}^{(i)})$ and $\Phi(\mathbf{x}^{(j)})$ in a feature space.

$$K(\mathrm{x}^{(i)}, \mathrm{x}^{(j)}) = < \Phi(\mathrm{x}^{(i)}), \Phi(\mathrm{x}^{(j)}) > \quad (3)$$

The use of kernel methods makes the problem complexity simplified by transforming the examples into the high dimensional data and, further more, can make use of infinite dimensions[1,14]. An appropriate choice of kernel function corresponds to an inner product of training data over input space and the computation of $< \Phi(\mathbf{x}^{(i)}), \Phi(\mathbf{x}^{(j)}) >$ is computed without explicit mapping $\Phi(\mathbf{x})$. The popular kernels are linear, polynomial, radial basis, and sigmoid functions shown in Table 1. The square of norm distance (2) is expressed with inner product and kernel function (4).

$$d^2(\Phi(\mathrm{x}^{(i)}), \Phi(\mathrm{x}^{(j)})) = K(\Phi(\mathrm{x}^{(i)}), \Phi(\mathrm{x}^{(i)})) - 2K(\Phi(\mathrm{x}^{(i)}), \Phi(\mathrm{x}^{(j)})) + K(\Phi(\mathrm{x}^{(j)}), \Phi(\mathrm{x}^{(j)})) \quad (4)$$

According to Mercer's theorem, the kernel or Gram matrix $G = \left[K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})\right]_{n \times n}$ must be positive semi-definite or has nonnegative eigenvalues [1,13,14]. Also, the kernel matrix must be symmetric.

$$K(\Phi(\mathrm{x}^{(i)}), \Phi(\mathrm{x}^{(j)})) = < \Phi(\mathrm{x}^{(i)}), \Phi(\mathrm{x}^{(j)}) > = < \Phi(\mathrm{x}^{(j)}), \Phi(\mathrm{x}^{(i)}) > = K(\Phi(\mathrm{x}^{(j)}), \Phi(\mathrm{x}^{(i)})) \quad (5)$$

It may imply a kernel nearest neighbor algorithm since we compute kernel norm distance and apply standard nearest neighbor algorithm in a kernel-based feature space. In [15], the kernel nearest neighbor algorithm degenerates to standard nearest neighbor algorithm when we use a kernel method. Therefore, all results of nearest neighbor algorithm can be considered as specific results of kernel nearest neighbor algorithm. This can expect that the results of kernel nearest neighbor algorithm are always no worse than those of standard nearest neighbor algorithm through a kernel method. In this paper, a kernel nearest neighbor rule is devised to pick up training data around a decision boundary. A pair of examples seems to be near a decision boundary if their norm distance is relatively small and their class labels are different.

Table 1: The examples of kernel functions with their own adjustable parameters

Kernel	Expression	Parameter
Linear	$K(\mathrm{x}^{(i)}, \mathrm{x}^{(j)}) = < \mathrm{x}^{(i)}, \mathrm{x}^{(j)} >$	-
Polynomial	$K(\mathrm{x}^{(i)}, \mathrm{x}^{(j)}) = (< \mathrm{x}^{(i)}, \mathrm{x}^{(j)} > + c)^d$	c, d
Radial basis	$K(\mathrm{x}^{(i)}, \mathrm{x}^{(j)}) = \exp(-\frac{\|\mathrm{x}^{(i)} - \mathrm{x}^{(j)}\|^2}{\sigma^2})$	σ
Sigmoid	$K(\mathrm{x}^{(i)}, \mathrm{x}^{(j)}) = \tanh(\gamma < \mathrm{x}^{(i)}, \mathrm{x}^{(j)} > + c)$	γ, c

2.2 Large Margin Classifiers

A SVM tries to learn a linear function for new problem P^{new} changed from P. The learned function becomes a hyperplane which separates a feature data into one of two classes. The learned hyperplane separates the data with the maximum margin in the feature space R^D. Given a hyperplane ($\mathbf{w}$,b) and a test data $\mathbf{x}$, the linear decision function $f(\mathbf{x})$ predicts the class of $\mathbf{x}$

$$f(\mathbf{x}) =< \mathbf{w}, \Phi(\mathbf{x}) > +b \tag{6}$$

where **w** is normal to the hyperplane and *b* is a bias. The test data **x** is a positive data if $f(\mathrm{x}) > 0$ or a a negative data otherwise. If the problem is linearly separable over the feature space, then there exists a hyperplane $< \mathrm{w}, \Phi(\mathrm{x}) >= 0$ such that the distance $y_i f(x^{(i)})$ to $\mathrm{x}^{(i)}$ from the hyperplane is greater than 0.

For a hyperplane (w, b) such that $f(\mathrm{x}) =< \mathrm{w}, \Phi(\mathrm{x}) > +b = 0$, all pairs $(\lambda \mathrm{w}, \lambda b)$ for $\lambda \in R$ are the same hyperplane. The hyperplane is rescaled so that the distance to the closest example from the hyperplane should be 1. Then, each $(\mathrm{x}^{(i)}, y_i) \in P$ satisfies $y_i f(\mathrm{x}^{(i)}) \geq 0$. The geometric margin γ_i of $(\mathrm{x}^{(i)}, y_i)$ is the distance to the hyperplane in which the minimum distance is at least $\frac{1}{\| \mathrm{w} \|}$.

$$\gamma_i = \frac{1}{\| \mathrm{w} \|} y_i (< \mathrm{w}, \Phi(\mathrm{x}^{(i)}) > +b) \tag{7}$$

Therefore, the SVM problem is to model the separating hyperplane with the large margin. The solution is accomplished by minimizing $\| \mathrm{w} \|$ under the distance constraints and formulated as the primal quadratic optimization problem (8).

$$\begin{aligned} \min_{(w,b)} \quad & \frac{1}{2} \| \mathrm{w} \|^2 \\ s.t \quad & y_i f(\mathrm{x}^{(i)}) \geq 1, \ \forall i \end{aligned} \tag{8}$$

When P^{new} is not linearly separable due to class overlapping or noisy data, SVM allows for example dependent loss. By introducing a slack variable ξ_i for each training data $\mathrm{x}^{(i)}$ and relaxing distance constraints

$$y_i f(\mathrm{x}^{(i)}) = y_i (< \mathrm{w}, \Phi(\mathrm{x}^{(i)}) > +b) \geq 1 - \xi_i, \ \forall i \text{ and } \xi_i \geq 0 \tag{9}$$

the primal problem (8) is modified as (10).

$$\begin{aligned} \min_{(\mathrm{w},b)} \quad & \frac{1}{2} \| \mathrm{w} \|^2 + C\xi \\ s.t \quad & y_i f(\mathrm{x}^{(i)}) \geq 1 - \xi_i \\ & \xi_i \geq 0, \ \forall i \end{aligned} \tag{10}$$

The parameter *C* is a positive penalty constant that controls the trade-off between large margins and margin violations. The slack variable ξ_i is the loss of $\mathrm{x}^{(i)}$ that measures how much its distance constraint is satisfied and allowed to be non-separable in finding a solution. Using the Lagrange multiplier method, the dual quadratic programming (11) of (10) is preferred to find a solution due to simplified constraints and implicit feature mapping.

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \alpha_i \alpha_j y_j < \Phi(\mathrm{x}^{(i)}), \Phi(\mathrm{x}^{(j)}) > \\ s.t \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq C, \ \forall i \end{aligned} \tag{11}$$

The constraints of (11) are much more simplified than those of (10) and easier to handle. By simply replacing the inner product of (11) with a certain kernel function (3), a SVM can implicitly map the training data $\mathrm{x}^{(i)}$ to the feature data $\Phi(\mathrm{x})$ in R^D. According to the Karush-Kuhn-Tucker(KKT) optimality condition, the solution of (11) groups the training examples into one of three categories depending on :- non-support vector, ordinary support vector, and error support vectors.

$$\alpha_i = 0 \quad \Rightarrow y_i f(\mathrm{x}^{(i)}) > 1 \text{ (non-support vector)}$$
$$0 < \alpha_i < C \quad \Rightarrow y_i f(\mathrm{x}^{(i)}) = 1 \text{ (support vector)}$$
$$\alpha_i = C \quad \Rightarrow y_i f(\mathrm{x}^{(i)}) < 1 \text{ (error support vector)}$$

Non-support vectors have no influence in building the decision function. The $\mathrm{x}^{(i)}$'s($\alpha_i > 0$), lies either on the margin or inside the margin area. The $\mathrm{x}^{(i)}$'s($\alpha_i = C$) lie on the opposite class area. Only the ordinary and error support vectors build the decision boundary. The weight vector and bias (w, b) is computed with support vectors.

$$\mathrm{w} = \sum_{\alpha_i \neq 0} \alpha_i y_i \Phi(\mathrm{x}^{(i)}) \tag{12}$$

The bias *b* is estimated with a pair of $(\mathrm{x}^{(i)}, +1)$ and $(\mathrm{x}^{(j)}, -1)$ such that $0 < \alpha_i, \alpha_j \leq C$.

$$b = -\frac{1}{2}(<\mathrm{w}, \Phi(\mathrm{x}^{(i)})> + <\mathrm{w}, \Phi(\mathrm{x}^{(i)})>) \tag{13}$$

Knowing the parameter (w, b), the decision function (6) of a test example x becomes

$$f(\mathrm{x}) = \sum_{\alpha_i \neq 0} \alpha_i y_i K(\mathrm{x}^{(i)}, \mathrm{x}) + b. \tag{14}$$

2.3 Implementation Issues of SVMs

Training a SVM classifier is equal to solve the dual problem (11) under the constraints. The most straightforward way to train a SVM is to feed to a quadratic programming(QP) solver. Standard QP solvers, such as CPLEX, LOQO, LINDO, MINOS, and Matlab QP solver, are used for finding the optimal solution [16]. For the large-scale data, these standard solvers are unlikely to find the solution due to the limitation of large memory and enormous training time. In order to overcome this difficulty, various approaches have been studied by utilizing the sparsity of the SVM solution and the KKT optimality conditions.

The chunking strategy [17] make use of the sparseness of a SVM solution and a generic QP solver. It begins to learn a SVM with an arbitrary subset or chunk of data. The algorithm continues to learn a SVM of a new chunk consisting of the support vectors from the previous chunk and some of the non-chunked data violating the KKT conditions. This procedure is iterated until the entire set of nonzero Lagrange multipliers are identified. Even though this approach is applicable for large-scale data, it requires a QP optimizer to solve the smaller QP problems in sequence.

The decomposition algorithm [18,19] breaks a large problem into a series of smaller QP subproblems and finds the solution of the subproblems. The algorithm solves a sequence of QP subproblems that always contain at least one training example violating the KKT conditions. Consequently, each learning step keeps the size of the QP subproblem fixed and updates one training example for the next iteration. The decomposition approach is effective for training large-scale problem and benefits from fast learning, compared to the chunking algorithm. However a QP solver is still required for implementing this algorithm.

The key idea of Sequential Minimal Optimization(SMO) [20] is that a QP subproblem of size two is solved analytically without a QP solver. Various heuristics are based on the KKT conditions to choose two training examples to jointly optimize at each iteration [16]. Through experiments, the SMO algorithm allows to be several order of magnitude faster and to exhibit better scaling properties than the classical chunking algorithm.

According to the KKT optimality condition, the learned hyperplane depends only on the small set of support vectors. The number of support vectors is much smaller than the total number of training examples. In most cases, support vectors lie either on the margin or inside the margin area. A SVM model will be learned fast if the set of training examples consists only of examples that might become support vectors. The decision hyperplane will be similar to or the same as the learned boundary with the whole training set. In theory, the QP solver seeks to identify the support

vectors from the given training examples. The support vectors are the training examples around the learning hyperplane and more likely to be misclassified. If there exist a computational way to select training examples which might be support vectors, then the speed of SVM training can be improved without degrading the generalization performance. Even though the number of the reduced training set is larger than that of the set of desired support vectors, the SVM training speed will be significantly improved since the algorithm scales quadratically to the number of examples on many problem. In the next section, we propose a data reduction method that selects a subset of training examples for fast SVM training.

3.0 DATA SELECTION FOR TRAINING SVMs

The proposed data selection strategy is based on the geometrical analysis that a SVM learns an optimal hyperplane bisecting the two convex sets so that the margin distances of the closest positive and negative examples to the decision boundary are equidistant. Support vectors tend to be the training examples closer to the decision boundary. Even though the final decision function is not built until a SVM finishes learning, the neighborhood property can be utilized to select training samples around a decision boundary. The nearest neighbor members of a training example are easily computed for any distance metric.

The neighbors of boundary or noisy samples are much more heterogeneous than those of centered samples. Tomek links have been employed in order to improve the learning performance by removing noisy or borderline data [12, 20, 21]. Tomek links are pairs of samples with different classes. In a 2-class classification, the simple modification of nearest neighbor algorithm can implement a Tomek link detection method. If the classes of an example and its neareat neighbor are different, the pair will be a Tomek link.

The consideration of Tomek links and distance metric leads to the data selection algorithm. The algorithm is based on set operations and shown in Fig 1. Given a training set $T=(X,Y)$ and a kernel mapping $K(\cdot,\cdot)$, the algorithm returns the selected subset S by using the distance matrix and the set of Tomek links. The distance matrix $d(\cdot,\cdot)$, and class vector Y is looked up for Tomek links set in ascending order. Let H be a set of Tomek link pairs (p_i, q_i) where $Y[p_i] \neq Y[q_i]$. The shorter pairs of H become more informative than the longer pairs in consideration of boundary examples. The nearest neighbors of examples in T are checked for the reduced training set S only including boundary points. Boundary points are inferred from closest pairs with different class labels.

SelectExamples(X[1:n],Y[1:n],K(·,·)) initializes the set S of class members of the shortest distance (p_1, q_1). The iterative step tests whether the nearest neighbors of the next pair in H are augmented to S and continues until all members of T are checked. The boolean variable $V[p]$ is marked by *visited* if the p-th example is tested.

The detail operations of the data selection algorithm are geometrically described in Fig 2. A circle belongs to one class and a square belongs to the other class. The hollowed points are selected as the members of S by the previous iterations. The decision boundary is drawn by dotted line and a Tomek link pair is connected with solid line. The neighbors n_p and n_q are the nearest neighbors of p and q and marked with the dashed arrows. The next pair (p,q) of H and the neighbors n_p and n_q are examined for being new candidates of S if p or $q \notin S$. According to which neighbor of p or q is a member of S or not in S, the possible case can be one of three conditions:- $q \in S$ and $n_p \in S$, $q \in S$ and $n_p \notin S$, or $n_p \notin S$ and $n_p \notin S$. When q is in S and $n_p \in S$, the neighbor n_p is much closer than q because n_p is preselected as a member of S. When q is in S and $n_p \notin S$, the class label of n_p is the same as that of p, the distances to p and n_p from q is compared in order to pick up a near sample to the decision boundary. If $d(q, n_p) < d(q, p)$, then n_p is a new member of S because this point is much closer than p to the decision boundary. Otherwise, the algorithm selects p as a member of S. The distance measure provides a way to select which of Tomek link members or their neighbors are closer to the decision boundary. If n_p and n_q are not in S, the algorithm simply selects the closer point of p or q by comparing the distances $d(n_p, q)$ and $d(n_q, p)$.

SelectExamples$(X[1:n], Y[1:n], K(\cdot,\cdot))$ {X and Y are the training examples and classes. $K(\cdot,\cdot)$ is a kernel function. This functions returns the set S of indexes on the selected examples in X and Y.}
$T = \{1,2,3,\cdots,n\}; S = \emptyset; V[1..n]$ = *None*
Calculate the kernel matrix $G = [d(i,j)]_{n\times n}$ by Equation (4)
Compute Tomek ordering $H = ((p_1,q_1),(p_2,q_2),\cdots,(p_n,q_n))$ such that $Y[p_i] \neq Y[q_i]$ and $d(p_i,q_i) \leq d(p_{i+1},q_{i+1})$ for all i and j, $i \neq j$
$S = S \cup \{p_1,q_1\}; V[p_1] = visited; V[q_1] = visited$
repeat
  for all each $(p_i,q_i) \in H$ **do**
    $(n_p, cls_p) = NearestNeighbor(p_i, T)$
    $(n_q, cls_q) = NearestNeighbor(q_i, T)$
    if $p_i \in S$ **then**
      $q = p_i;\ n_q = n_p; p = q_i$
    end if
    if $q \in S$ and $n_p \in S$ **then**
      $V[p]$ = *visited*
    end if
    if $q \in S$ and $n_p \notin S$ and $d(q,p) < d(q,n_p)$ **then**
      $S = S \cup \{p\}; V[p] = visited$
    else
      $S = S \cup \{n_p\}; V[n_p] = visited$
    end if
    if $n_q \notin S$ and $n_p \notin S$ and $d(n_p,q) < d(n_q,p)$ **then**
      $S = S \cup \{q,n_p\}; V[q] = V[n_p] = visited$
    else
      $S = S \cup \{p,n_q\}; V[p] = V[n_q] = visited$
    end if
  end for
until $V[i] == visited\ \forall i$
return S

Fig. 1. Data selection algorithm for SVM learning

In the iteration step, the algorithm finds the nearest neighbors n_p and n_q of p_i and q_i for $(p_i,q_i) \in H$. If $p_i \in S$, the algorithm sets the indices of p, q, n_p and n_q. The nearest neighbors n_p and n_q are checked for their membership in S.

When $q \in S$ and $n_p \in S$ (Fig 2 (a)), the algorithm skips this case after assigning *visited* to *V[p]*. If $q \in S$ and $n_p \notin S$(Fig 2 (b)), the algorithm chooses the closer point of p and n_p from q. The point p is added to S if $d(p,q) < d(q,n_p)$ or n_p otherwise. If the class labels of nearest neighbors are the same but not in S, then it is considered that p or q is farther away from the boundary but their closest members can become alternatives for new members in S. However, if n_p is not in S, the distance from n_q to q is checked for closeness to the boundary. In the case of $n_q \notin S$ and $n_p \notin S$(Fig 2 (c)), the algorithm adds $\{q,n_p\}$ to S if $d(n_p,q) < d(n_q,p)$ or $\{p,n_q\}$ otherwise. When n_p and n_q are in S, the algorithm does not check this case because n_p and n_q are more informative than p and q. Except these three cases, other possible cases do not happen because only ordered Tomek links pairs are checked as new members in S. The algorithm terminstes after all members in T are checked.

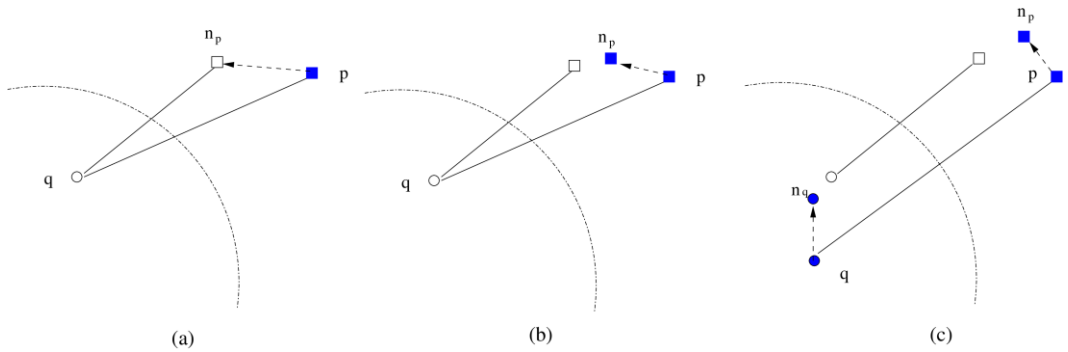

Fig. 2. The possible cases for selecting boundary examples

When applying the data selection algorithm for classification problem *P*, we take the three steps. Firstly, we utilize a kernel method with necessary parameter settings for *P*. Secondly, a new problem P^{new} is made by selecting the training set *S* with the algorithm $SelectExamples(X, Y, K(\cdot,\cdot))$. Finally, we model and evaluate a linear decision function of a SVM on P^{new} with the same kernel function at the first step.

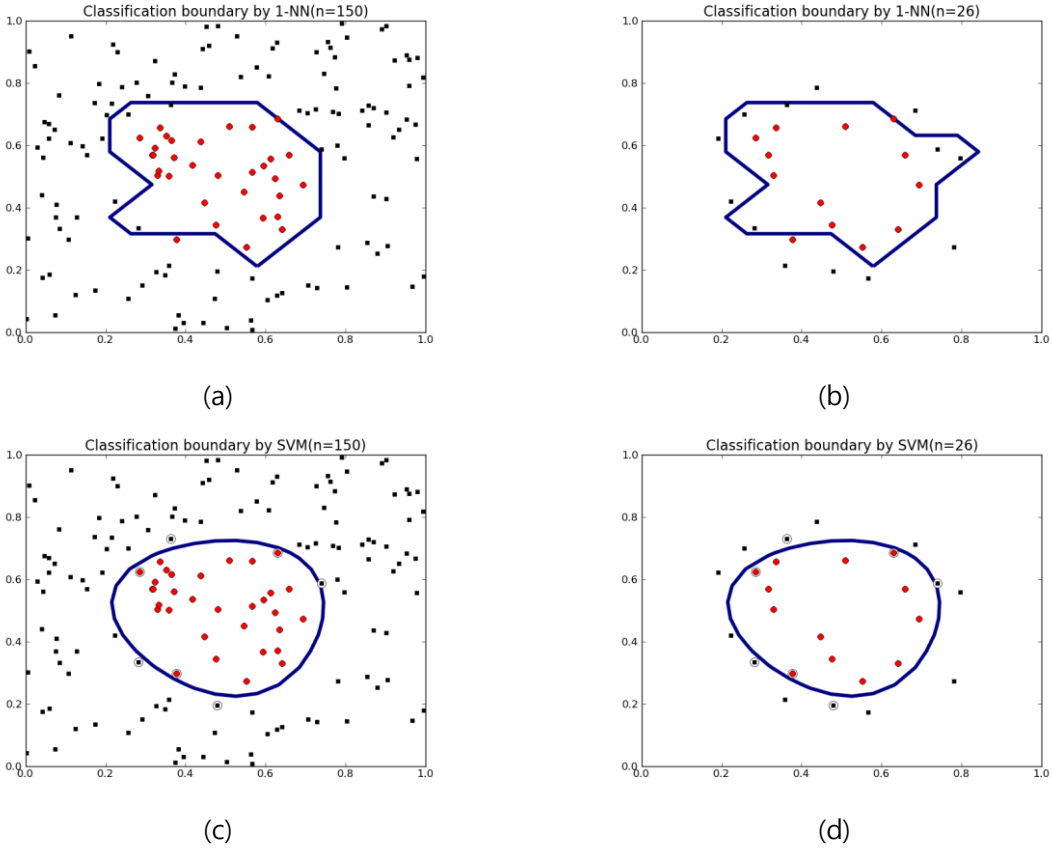

Fig. 3. The simulation results of nearest neighbor and SVM with an artificial problem

Fig 3 shows the learning effect on an artificial problem in the plane. The artificial problem consists of 150 samples with 118 negatives and 32 positives generated by random numbers. The proposed algorithm selects the 26 examples(17%) which have the equal number of positives and negatives. Fig 3 (a) and (b) are the simulation results of nearest neighbor algorithm without and with data selection. The modeled decision boundaries are different and there exit some missclassified examples. Fig 3 (c) and (d) are the decision boundaries that a SVM learns with *C*=300 and chooses a RBF kernel with $\sigma^2 = 0.5$. The support vectors are highlighted with the decision boundary. The SVM model of the reduced data set constructs the same decision boundary as that of the original problem. Also, their numbers of support vectors are exactly same. There are 7 support vectors in which 3 positives and 4 negatives are chosen. The result of nearest neighbor algorithm has 4 points predicted wrong but the SVM correctly classifies all

points. In both cases, the class boundary of SVM is smoother and more accurate than that of nearest neighbor **algorithm.**

4.0 EXPERIMENTAL RESULTS

In this section, we show the experimental results on various real-world problems [19, 22]. The problem size varies from a few hundreds to ten thousands. Each task uses a radial basis kernel with $\sigma^2 = 0.5$ and the performance metrics are measured through a 5-fold cross-validation way. The metrics is classification error, data reduction rate, the number of iterations and the number of support vectors. Table 2 describes all the selected problems in terms of data size and the number of attributes. The problems are processed for having two classes through merging or extracting some class examples. The penalty parameter *C* is 0.5, 1, 5, 10, 100. LIBSVM is chosen as the implementation of the SVM algorithm [22]. The results of our learning approach are compared to those of SVMs with the original problems. In order to show the efficiency of a data selection approach in SVM learning, we test the NPPS algorithm [10] for the same tasks. The value *k* of nearest neighbors is 3(NPPS-3) and 5(NPPS-5).

Table 2: The selected problems in the experiments

Problem	Size		# Features
	Training	Testing	
Breast Cancer(BC)	546	137	10
Australian(AU)	551	139	14
Pima(PI)	614	154	8
Liver Disorder(LD)	276	69	6
Ionosphere(IO)	280	71	34
Vowel(VO)	528	462	10
Letter-BG(BG)	759	425	16
Car Evaluation(CE)	1,382	346	6
Mushroom(MU)	6,498	1,626	112
Adult5a(AU)	6,414	26,147	123
Svmguide1(SV)	3,089	4,000	4
Web4a(WE)	7,366	4,383	300
Shuttle(SH)	14,500	43,500	9

Table 3 presents the misclassification errors in training and testing. In most of the selected problems, the error results of the proposed approach are competitive to those of the SVM. In Fig 4 and 5, the proposed approach outperforms the NPPS-3 and NPPS-5 except Australian, Pima, and Adult5a. The training error of SVM is more accurate than that of the data selection approaches in Australian and Inosphere. With Breast Cancer, Australian, Svmguide1 and Shuttle, the testing error of SVM is higher than that of the data selection methods (Fig 5). Comparing the results of the NPPS method, the NPPS-3 might not select enough near-boundary samples to learn a SVM decision function. When the NPPS applies to Mushroom, the algorithm cannot construct a reduced training set for SVM. This is because the NPPS could not select any examples in the data selection step which is shown in Table 4.

Table 3: Misclassification error in training and testing

Problem	Training				Testing			
	SVM	NPPS-3	NPPS-5	Proposed	SVM	NPPS-3	NPPS-5	Proposed
Breast Cancer(BC)	0.03	0.11	0.12	0.03	0.05	0.31	0.13	0.08
Australian(AU)	0.11	0.26	0.22	0.28	0.14	0.14	0.15	0.24
Pima(PI)	0.22	0.26	0.26	0.32	0.24	0.22	0.21	0.24
Liver Disorder(LD)	0.27	0.31	0.26	0.22	0.26	0.34	0.26	0.05
Ionosphere(IO)	0.05	0.00	0.00	0.03	0.08	0.30	0.23	0.10
Vowel(VO)	0.09	0.25	0.01	0.10	0.09	0.64	0.27	0.09
Letter-BG(BG)	0.02	0.21	0.03	0.01	0.03	0.33	0.11	0.02
Car Evaluation(CE)	0.01	0.05	0.04	0.02	0.03	0.11	0.11	0.03
Mushroom(MU)	0.00	-	-	0.00	0.00	-	-	0.00
Adult5a(AU)	0.13	0.19	0.20	0.27	0.16	0.16	0.16	0.15
Svmguide1(SV)	0.03	0.34	0.26	0.06	0.04	0.40	0.09	0.15
Web4a(WE)	0.01	0.04	0.05	0.01	0.01	0.04	0.05	0.01
Shuttle(SH)	0.07	0.35	0.28	0.03	0.03	0.34	0.28	0.13

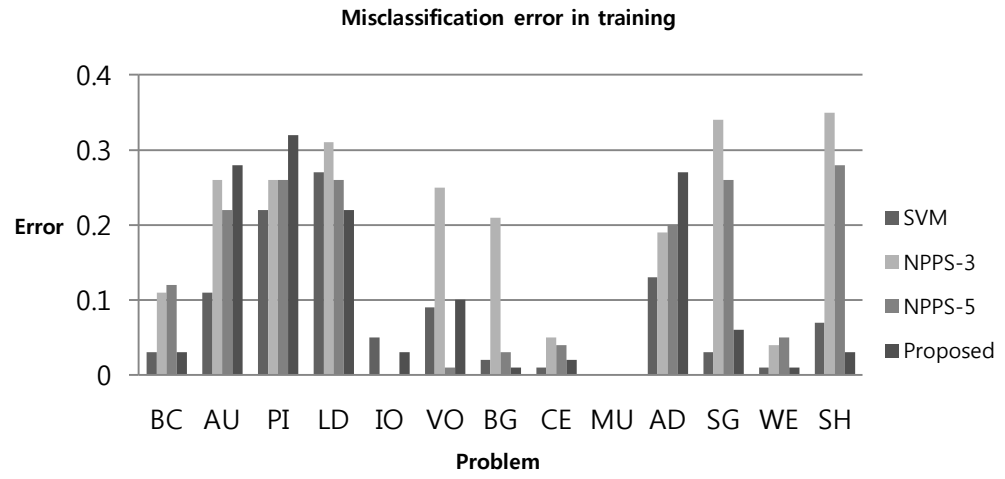

Fig. 4. Misclassification error in training

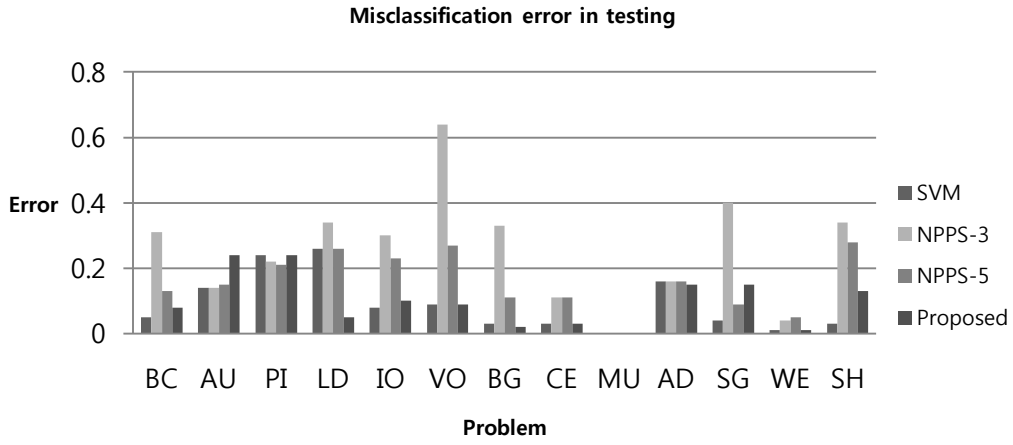

Fig. 5. Misclassification error in testing

Table 4 and 5 analyze the effects of data reduction and learning time. Fig 6 and 7 show their ratios in proportion to those results in SVMs. The reduction rate is estimated by the number of selected samples over the original training size. The reduction rate of the proposed approach varies from at least 2.6% (Web4a) to at most 56.2% (Liver Disorder). This is an evidence that a great number of examples can be removed from the original problems, but the SVM learning might make up for a little loss of the generalization capability of the whole training set (Fig 4 and 5). In most problems, the reduction rate of the proposed data selection algorithm is between that of NPPS-3 and that of NPPS-5. The rate of NPPS-5 is higher than that of NPPS-3, depending on the number k of neighbors.

Table 4: Comparisons of the numbers of selected examples

Problem	SVM	NPPS-3		NPPS-5		Proposed	
		No	%	No	%	No	%
Breast Cancer(BC)	546.0	54.8	10.0	123.5	22.6	60.0	11.0
Australian(AU)	551.0	152.0	27.6	278.8	50.6	163.4	29.7
Pima(PI)	614.0	304.6	49.6	433.8	70.7	254.6	41.5
Liver Disorder(LD)	276.0	199.0	72.1	246.4	89.3	155.2	56.2
Ionosphere(IO)	280.0	40.8	14.6	62.8	22.4	72.2	25.8
Vowel(VO)	528.0	8.6	1.6	52.0	9.8	184.0	34.8
Letter-BG(BG)	759.0	23.6	3.1	54.0	7.1	265.6	35.0
Car Evaluation(CE)	1,382.0	296.4	21.4	480.8	34.8	663.2	48.0
Mushroom(MU)	6,498.0	-	-	-	-	2779.0	42.8
Adult5a(AU)	6,414.0	1,584.6	24.7	2,338.4	36.5	1571.0	24.5
Svmguide1(SV)	3,089.0	333.0	10.8	441.0	14.1	1183.4	38.3
Web4a(WE)	7,366.0	181.4	2.5	298.6	4.1	190.6	2.6
Shuttle(SH)	14,500.0	22.0	0.2	34.6	0.2	1842.8	12.7

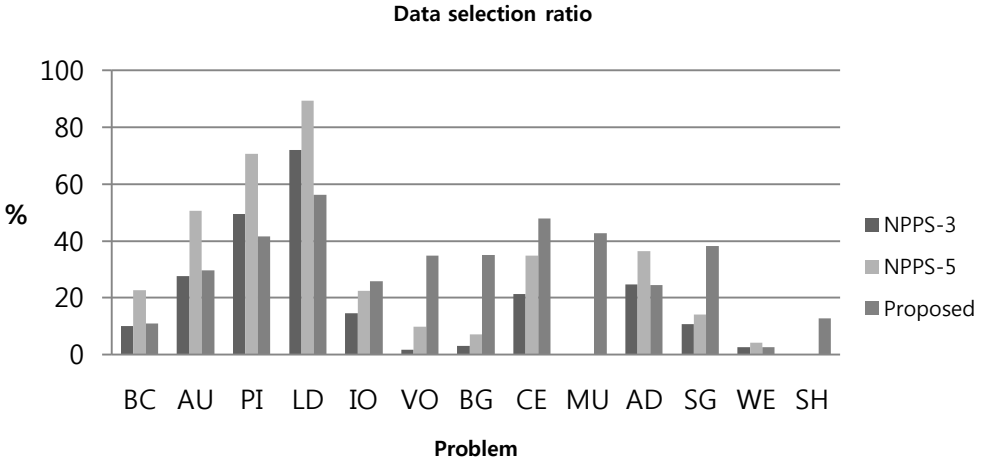

Fig. 6. Data selection ratio

Table 5 and Fig 7 presents the total of iterations and the relative rates of the data selection approaches to the SVM learning. The learning time can be estimated by the number of iterations. In the data selection approaches, the number of iterations becomes relatively small compared to that of a SVM. Comparing Table 4 and Table 5, the data selection approach helps to curtail the SVM learning time. In most experiments, a large number of iterations are reduced. The proposed method takes 2 times as long as the SVM for Letter and the NPPS-5 takes 3 times for Vowel. Generally a data selection method can speed up the training process by reducing the time complexity of a given training set and thus, the time taken for the whole learning process might be smaller than that by a standard SVM. Let n_s be the size of the selected examples. The time complexity for the data selection and a SVM is $O(n^2 + n_s^3)$, where the cost of the data selection approach is and the cost of the SVM is $O(n_s^3)$. The value n_s depends on the characteristics of the training set, but usually it holds that $n_s << n$, with n_s corresponding to a small part of the whole trainging set.

Table 5: Trainingns of the number of iterations

Problem	SVM	NPPS-3		NPPS-5		Proposed	
		No	%	No	%	No	%
Breast Cancer(BC)	153.7	29.8	19.4	74.4	48.4	59.8	38.9
Australian(AU)	681.4	174.6	25.6	278.8	40.9	181.8	26.7
Pima(PI)	493.8	274	55.5	433.8	87.8	332.4	67.3
Liver Disorder(LD)	246.6	183.2	74.3	317.8	128.9	147.6	59.9
Ionosphere(IO)	602.2	94.6	15.7	121	20.1	282.6	46.9
Vowel(VO)	116.6	5.0	4.3	363.4	311.7	56.4	48.4
Letter-BG(BG)	404.6	33.4	8.3	362.8	89.7	870.2	215.1
Car Evaluation(CE)	806.0	398.2	49.4	523.8	65.0	698.8	86.7
Mushroom(MU)	3,691.0	-	-	-	-	2,003.6	54.3
Adult5a(AU)	26,727.2	11,122.6	41.6	162,14.2	60.7	316.0	1.2
Svmguide1(SV)	414.6	168.0	40.5	252.0	60.8	249.4	60.2
Web4a(WE)	4,500.0	528.8	11.8	925.8	20.6	522.8	11.6
Shuttle(SH)	980.2	23.4	2.4	55.4	5.7	393.6	40.2

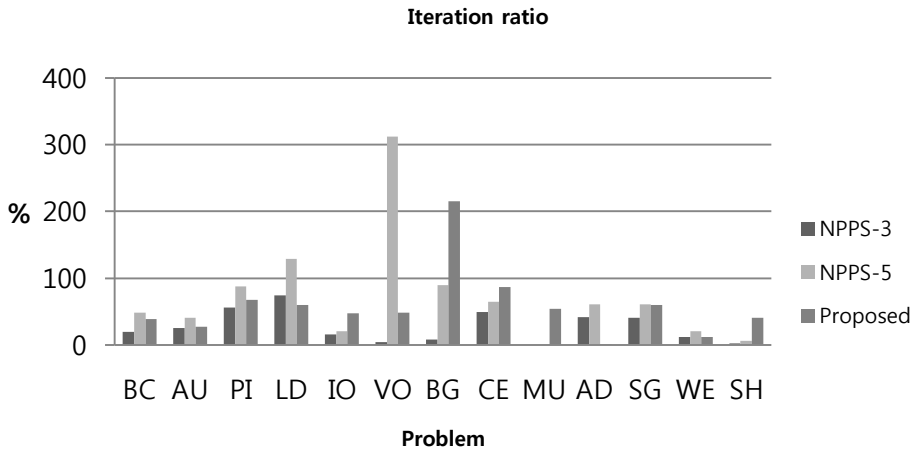

Fig. 7. Iteration ratio in SVM learning

Table 6 shows the rate that a selected sample becomes a support vector. The rate of the number of selected data over the training size is much higher than in the results of the SVM in Fig 8. In SVM learning, a data selection approach can make to increase the rate that a selected data becomes a support vector and speeds up a SVM learning by reducing a significant amount of training data. In both the NPPS and the proposed approach, the rate of being support vector is higher than that of the SVM. Comparing to the rate of the NPPS, the proposed method scores high in Breast Cancer, Pima, Liver Disorder, Adults5a, Web4a. By directly comparing these rates, we are unable to provide a significant difference between the proposed method and the NPPS. The comparative studies on these methods need to be done for practical problems. Considering the NPPS method, the rate of the NPPS-5 is higher than that of the NPPS-3 because the number of nearest neighbors is large.

Table 6: Comparisons of the numbers of support vectors

Problem	SVM		NPPS-3		NPPS-5		Proposed	
	No	%	No	%	No	%	No	%
Breast Cancer(BC)	49.6	9.1	19.0	34.7	30.6	24.8	29.6	49.3
Australian(AU)	189.0	34.3	90.8	59.7	136.6	49.0	96.4	59.0
Pima(PI)	335.4	54.6	210.6	69.1	278.2	64.1	234.2	92.0
Liver Disorder(LD)	223.2	80.9	169.2	85.0	194.2	78.8	139.4	89.8
Ionosphere(IO)	52.2	18.6	16.6	40.7	21.4	34.1	32.8	45.4
Vowel(VO)	83.6	15.8	6.6	76.7	22.4	43.1	47.4	25.8
Letter-BG(BG)	132.8	17.5	21.8	92.4	28.6	53.0	45.2	17.0
Car Evaluation(CE)	300.0	21.7	197.4	66.6	268	55.7	265.2	40.0
Mushroom(MU)	287.6	4.4	-	-	-	-	203.2	7.3
Adult5a(AU)	2,103.8	32.8	1,010	63.7	14,36.4	61.4	1,288.0	82.0
Svmguide1(SV)	631.8	20.5	248.0	74.5	382	86.6	386.4	32.7
Web4a(WE)	307.6	4.2	1,03.4	57.0	126.8	42.5	148.4	77.9
Shuttle(SH)	1,532.6	10.6	16.8	76.4	21.6	62.4	484.6	26.3

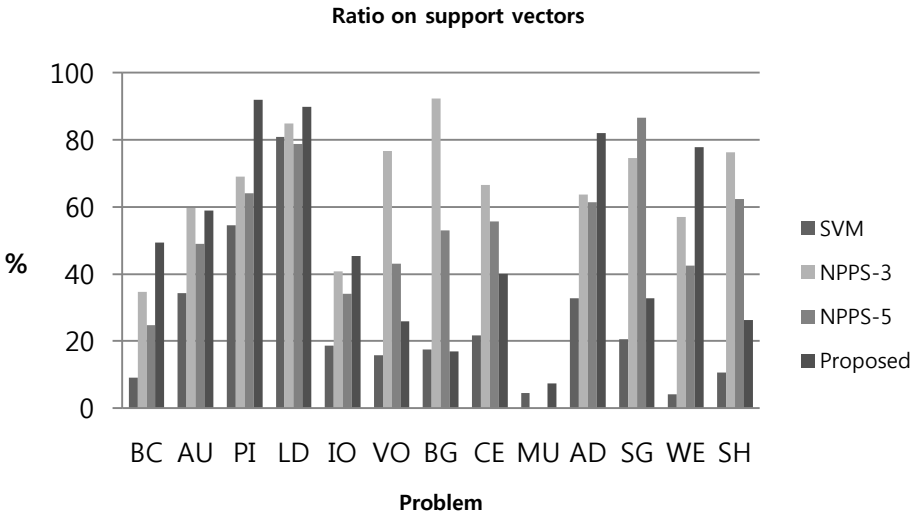

Fig. 8. Ratio on the number of selected support vectors

5.0 CONCLUSIONS

This paper presents a data selection method for SVM training. The proposed data selection is based on Tomek links and distance measure. Tomek link is a pair with different classes and can be found by modifying the 1-nearest neighbor algorithm. Distance measure is considered in feature space, not in input space, and used for choosing more informative points by analyzing the relationships between Tomek link pairs and their distances. According to the geometrical relationship between Tomek link pairs and neighbors, the possible cases are checked for reconstructing the original training set into a new problem, lessening the time complexity of the given problem. The proposed data selection method was tested on several real-world problems in order to analyze its effectiveness in terms of generalization performance, data reduction rate, learning time. Specifically, we compared the SVM results with the entire training set and the reduced dataset. From our experiments, the significant data reduction was accomplished with a little loss of generalization performance, compared to the results of the SVM. This experimental evidence substantiates that a SVM model depends on the small subset of training examples and also the use of a data selection method would be effective in aiming at fast SVM learning. The data reduction technique provides the benefits of SVM training and testing due to the low learning complexity. As for the further works, there might be the comparative study on a significant difference between the proposed algorithm and others. In addition, the data selection approach is able to balance the equal number of examples per class for the skewed classes.

ACKNOWLEDGEMENT

This work was supported by the research program of Dankook University, South Korea(2009). Dr. Daewon Kim is the corresponding author of this paper.

REFERENCES

[1] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines (and other kernel-based learning methods)*. Cambridge University Press, 2000.
[2] J. M. Moguerza and A. Muoz, "Support vector machines with Applications". *Statistical Science*, No. 4, 2006, pp. 322–336.
[3] I. Steinwart, "Sparseness of support vector machines". *Journal of Machine Learning Research*, Vol. 4, 2003.
[4] X. Wu et al., *The top ten algorithms in data mining*, CRC Press, 2009.
[5] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets". *Lecture Notes in Computer Science*, Vol. 3201, 2004, pp. 39–50.
[6] V. Garca et al., "The class imbalance problem in pattern classification and learning". *Data Engineering*, 2007, pp. 283–291.
[7] F. Vilariño et al., "Experiments with svm and stratified sampling with an imbalanced problem: Detection of intestinal contractions". *Pattern Recognition and Image Analysis*, *Lecture Notes in Computer Science*, Vol. 3687, 2005, pp. 783–791.
[8] N. V. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique". *Journal of Artificial Intelligence Research*, Vol. 16 , 2002, pp. 321–357.
[9] H. J. Shin and S. Z. Cho, "Response modeling with support vector machines". *Expert Systems with Applications*, Vol. 30, No. 4, 2006, pp. 746 – 760.
[10] F. Angiulli, "Fast nearest neighbor condensation for large data sets classification". *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, 2007, pp. 1450–1464.
[11] J. Wang, P. Neskovic, and L. N. Cooper, "Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence". *Pattern Recognition*, Vol. 39, No. 3, 2006, pp. 417–423.
[12] B. Li, Q. Wang, and J. Hu, " A fast svm training method for very large datasets", *In Proceedings of the 2009 international joint conference on Neural Networks*, 2009, pp. 1784 –1789.
[13] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge University Press, 2004.
[14] K. Yu, L. Ji, and X. Zhang, "Kernel nearest-neighbor algorithm". *Neural Processing Letters*, Vol. 15, 2002, pp. 147–156.
[15] L. Bottou and C. J. Lin, *Support machine solvers*, in Large scale kernel machines, The MIT Press, 2007, pp. 1–27.
[16] B. E. Boser et al., "A training algorithm for optimal margin classifiers", *in Proceedings of the fifth annual workshop on Computational learning theory*, ACM, 1992, pp. 144–152.
[17] E. Osuna, R. Freund, and F. Girosit, "Training support vector machines: an application to face detection, Computer Vision and Pattern Recognition", *in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 130 –136.
[18] T. Joachims, *Making large-scale support vector machine learning practical*. The MIT Press, 1999, pp. 169–184.
[19] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*, in Advances in Kernel Methods - Support Vector Learning, The MIT Press, 1998.
[20] Gustavo E. A. P. A. Batista et al., "A study of the behavior of several methods for balancing machine learning training data". SIGKDD Explor., Newsl. 6, 2004, pp. 20–29.
[21] P. Jeatrakul, K.W. Wong and C.C. Fung, "Data cleaning for classification using misclassification analysis". *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 14, No. 3, 2010, pp. 297–302.
[22] UCI machine learning repository, http://archive.ics.uci.edu/ml/.
[23] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines". *ACM Transactions on Intelligent Systems and Technology*, Vol. 2, 2011.

BIOGRAPHY

Doosung Hwang received a M.S.(1990) from Chungnam National University, Taejeon, South Korea, and Ph.D. (2003) in Computer Science from Wayne State University, Detroit, Michigan, USA. He worked as a senior researcher for ETRI(Electronics and Telecommunications Research Institute), South Korea (1991-1998). He is currently an associate professor in Computer Science at Dankook Univeristy, South Korea. His research interests include datamining, parallel processing and database design.

Daewon Kim received a M.S. (1996) from the University of Southern California, LA, California, USA, and a Ph. D. (2002) in Electrical and Computer Engineering from Iowa State University, Ames, Iowa, USA. He worked as a senior researcher at Samsung Electronics Co. Ltd., Suwon, South Korea (2002-2004). He is currently an associate professor in Multimedia Engineering department at Dankook University, South Korea. His research interests include signal processing, mobile applications, and nondestructive evaluation.