Neighborhood Property based Pattern Selection For Support Vector Machines

Hyunjung Shin[1,2]
[1] Department of Industrial & Information Systems Engineering,
Ajou University, San 5, Wonchun-dong, Yeoungtong-gu,
443–749, Suwon, Korea
[2] Friedrich Miescher Laboratory, Max Planck Society,
Spemannstrasse. 37, 72076, Tübingen, Germany,
shin@tuebingen.mpg.de

Sungzoon Cho
Seoul National University,
San 56-1, Shillim-Dong, Kwanak-Gu, 151-744, Seoul, Korea,
zoon@snu.ac.kr

August 3, 2006

Abstract

Support Vector Machine (SVM) has been spotlighted in the machine learning community thanks to its theoretical soundness and practical performance. When applied to a large data set, however, it requires a large memory and long time for training. To cope with the practical difficulty, we propose a pattern selection algorithm based on neighborhood properties. The idea is to select only the patterns that are likely to be located near the decision boundary. Those patterns are expected to be more informative than the randomly selected patterns. The experimental results provide promising evidence that it is possible to successfully employ the proposed algorithm ahead of SVM training.

1 Introduction

Support Vector Machine (SVM) has been spotlighted in the machine learning community because of its theoretical soundness and practical performance. SVM has been highly successful in practical applications as diverse as face detection and recognition, handwritten character and digit recognition, text detection and categorization, *etc* (Byun & Lee, 2002; Dumais, 1998; Heisele et al., 2000; Joachims, 2002; Moghaddam & Yang, 2000; Osuna et al., 1997; Schölkopf et al., 1999). However, when applied to a large data set, SVM training can become computationally intractable. In the formulation of SVM quadratic programming (QP), the dimension of the kernel matrix ($M \times M$) is equal to the number of training patterns (M) (Vapnik, 1999). Thus, when the dataset is huge, training cannot be finished in a reasonable time, even if the kernel matrix could be loaded on the memory. Most standard SVM QP solvers have a time complexity of $\mathrm{O}(M^3)$: MINOS, CPLEX, LOQO and MATLAB QP routines. And the solvers using decomposition methods have a time complexity of $I \cdot \mathrm{O}(Mq + q^3)$, where I is the number of iterations and q is the size of the working set: Chunking, SMO, SVM$^{\mathrm{light}}$ and SOR (Hearst et al., 1997; Joachims, 2002; Schölkopf et al., 1999; Platt, 1999). Needless to say, I increases as M increases. Empirical studies have estimated the run time of common decomposition methods to be proportional to $\mathrm{O}(M^p)$ where p varies from approximately 1.7 to 3.0 depending on the problem (Hush et al., 2006; Laskov, 2002). Moreover, SVM requires heavy or repetitive computation to find a satisfactory model: for instance, which kernel is favorable over the others (among RBF, polynomial, sigmoid, and *etc.*), and additionally how to choose the kernel parameters (width of basis function, polynomial degree, offset and scale, respectively). There has been a considerable amount of work related to this topic, called by kernel learning, hyper-kernels, kernel alignment, *etc.* See Ong et al. (2005) and Sonnenburg et al. (2006), and the references therein. But the most common approaches for parameter selection are yet dependent on (cross–) validation. This implies one should train an SVM model multiple times until a proper model is found.

One way to circumvent this computational burden might be to select some of training patterns in advance. One of the distinguishable merits of SVM theory is that it clarifies which patterns are of importance to training. These patterns are distributed near the decision boundary, and fully and succinctly define the classification task at hand (Cauwenberghs & Poggio, 2001;

Pontil & Verri, 1998; Vapnik, 1999). Furthermore, the subset of support vectors (SVs) is almost identical regardless of which kernel function is opt for training (Schölkopf et al., 1995). From a computational point of view, it is therefore worth identifying a subset of *would-be* SVs in preprocessing, and then training the SVM model with the smaller set.

There have been several approaches to pattern selection that reduce the number of training patterns. Lyhyaoui et al. (1999) proposed the 1-nearest neighbor searching from the opposite class after class-wise clustering. But this method presumes no possible class overlap in the training set to properly find the patterns near the decision boundary. On the other hand, Almeida et al. (2000) conducted k-means clustering. A cluster is defined as *homogeneous* if it consists of the patterns from same class, while *heterogeneous* otherwise. All the patterns from a homogeneous cluster are replaced by a single centroid pattern, while the patterns from a heterogeneous cluster are all selected. The drawback is that it is not clear how to determine the number of clusters. Koggalage & Halgamuge (2004) also employed clustering to select the patterns from the training set. Their approach is quite similar to Almeida et al. (2000)'s: clustering is conducted on the entire training set first, and the patterns are chosen that belong to the heterogeneous clusters. For a homogeneous cluster, on the contrary, the patterns along the rim of cluster are selected instead of the centroid of Almeida et al. (2000)'s. It is relatively a safer approach since even in homogeneous clusters, the patterns near the decision boundary can exist if the cluster's boundary is almost in contact with the decision boundary. On the other hand, it has a relative shortcoming as well in that the patterns far away from the decision boundary are also picked as long as they lie along the rim. And further, the setting of the radius and defining of the width of the rim are still unclear. In the meantime, for the reduced SVM (RSVM) of Lee & Mangasarian (2001), Zheng et al. (2003) proposed to use the centroids of clusters instead of random samples. It is more reasonable since the centroids are more representative than random samples. However, these clustering-based algorithms have a common weakness: the selected patterns are fully dependent on clustering performance, which could be unstable. Liu & Nakagawa (2001) made a related performance comparison. Sohn & Dagli (2001) suggested a slightly different approach. It utilizes fuzzy class membership through k nearest neighbors. The score of fuzzy class membership is translated as a probability that how deeply a pattern belongs to a class. By the scores, the patterns having a

weak probability are eliminated from the training set. However, it overlooks the importance of the patterns near the decision boundary; they are equally treated to outliers (noisy pattern far from the decision boundary).

Active learning shares this issue of significant pattern identification with pattern selection (Brinker, 2003; Campbell et al., 2000; Schohn & Cohn, 2000). However, there are substantial differences between them. *First*, the primary motivation of active learning comes from the high cost of "obtaining labeled training patterns," not from that of training itself. For instance, in industrial process modeling, obtaining even a single training pattern may require several days. In email-filtering, obtaining training patterns is not expensive, but it takes many hours to label them. In pattern selection, on the other hand, the labeled training patterns are assumed to already exist. *Second*, active learning alternates between training with a newly introduced pattern and making queries over the next pattern. On the contrary, pattern selection runs only once before training as a preprocessor.

In this paper, we propose *neighborhood property based pattern selection algorithm* (NPPS). The practical time complexity of NPPS is vM, where v is the number of patterns in the overlap region around the decision boundary. We utilize k nearest neighbors to look around the pattern's periphery. The *first* neighborhood property is that "a pattern located near the decision boundary tends to have more heterogeneous neighbors in their class-membership." The *second* neighborhood property dictates that "a noisy pattern tends to belong to a different class from that of its neighbors." And the *third* neighborhood property is that "the neighbors of the decision boundary pattern tend to be located near the decision boundary as well." The first property is used for identifying the patterns located near the decision boundary. The second property is used for removing the patterns located on the wrong side of the decision boundary. And the third property is used for skipping unnecessary distance calculation, thus accelerating the pattern selection procedure. The paper also provides how to choose k for the proposed algorithm. Note that it has been skipped or dealt with as trivial in other pattern selection methods that employ either k-means clustering or k-nearest neighbor rule.

The remaining part of this paper is organized as follows. Section 2 briefly explains the SVM theory, in particular, the patterns critically affecting the

training. Section 3 presents the proposed method, NPPS, that selects the patterns near the decision boundary. Section 4 explains how to choose the number of neighbors, k–the parameter of the proposed method. Section 5 provides the experimental results on artificial datasets, real-world benchmarking datasets, and a real-world marketing dataset. We conclude this paper with some future work in Section 6.

2 Support Vector Machines and Critical Training Patterns

Support Vector Machines (SVMs) are a general class of statistical learning architectures that perform *structural risk minimization* on a nested set structure of separating hyperplanes (Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2002; Vapnik, 1999). Consider a binary classification problem with M patterns $(\vec{x}_i, y_i), i = 1, \cdots, M$ where $\vec{x}_i \in \Re^d$ and $y_i \in \{-1,\ 1\}$. Let us assume that patterns with $y_i = 1$ belong to class 1 while those with $y_i = -1$ belong to class 2. SVM training involves solving the following quadratic programming problem, which yields the largest margin ($\frac{2}{\|w\|}$) between classes.

$$\begin{aligned} \min \quad & \Theta\left(\vec{w}, \xi\right) = \frac{1}{2}||\vec{w}||^2 + C\sum_i^M \xi_i, \\ \text{s. t.} \quad & y_i\left(\vec{w}\cdot\Phi(\vec{x}_i) + b\right) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \ldots, M, \end{aligned} \tag{1}$$

where $\vec{w} \in \Re^d,\ b \in \Re$ (see Fig. 1), and an error tolerance parameter $C \in \Re$. Eq.(1) is the most general SVM formulation allowing both non-separable and nonlinear cases. The ξ's are nonnegative slack variables which play a role of allowing a certain level of misclassification for a non-separable case. The $\Phi(\cdot)$ is a mapping function for a nonlinear case that projects patterns from the input space into a feature space. This nonlinear mapping is performed implicitly by employing a kernel function $K(\vec{x}, \vec{x}')$ to avoid the costly calculation of inner products, $\Phi(\vec{x})\cdot\Phi(\vec{x})$. [1] There are three typical kernel functions,

[1] Aside from the computational efficiency by dot-product replacement, the kernels provide operational benefits as well: it is easy and natural to work on (or integrate) various types of data, i.e., vectors, sequences, text, images, graphs, *etc*, and to detect very general types of relations therein.

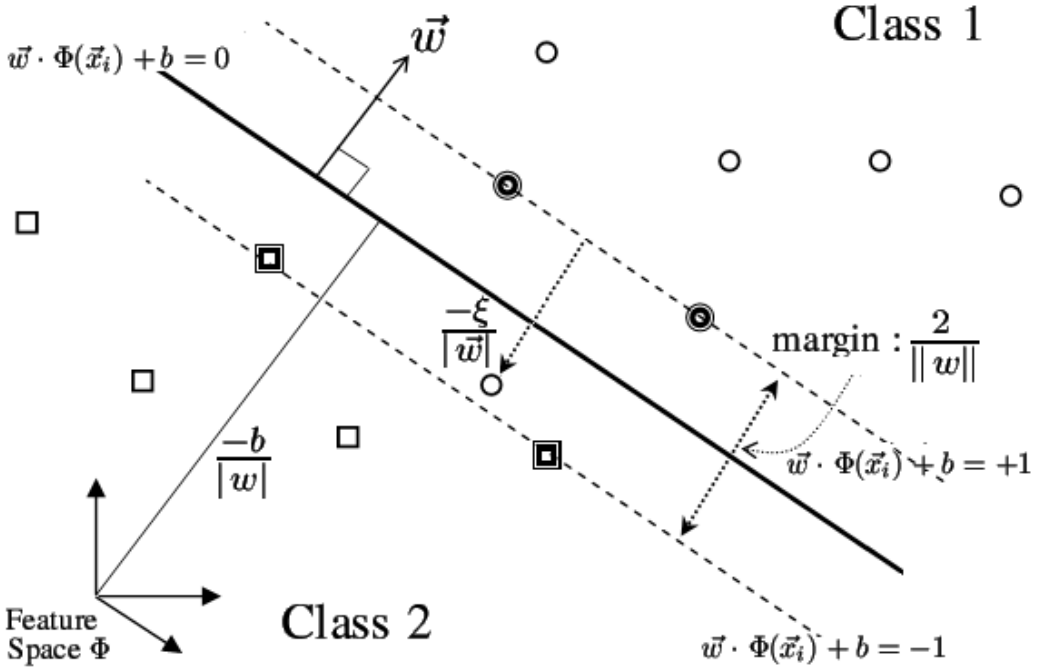


Figure 1: SVM classification problem: Through a mapping function $\Phi(\cdot)$, the class patterns are linearly separated in a feature space. The patterns determining both margin hyperplanes are outlined. The decision boundary is the half-way hyperplane between margins.

RBF, polynomial, and sigmoid in due order,

$$\begin{aligned} K(\vec{x},\vec{x}') &= exp\left(-||\vec{x}-\vec{x}'||^2/2\sigma^2\right), \\ K(\vec{x},\vec{x}') &= (\vec{x}\cdot\vec{x}'+1)^p, \\ K(\vec{x},\vec{x}') &= \tanh\left(\rho(\vec{x}\cdot\vec{x}')+\delta\right). \end{aligned} \tag{2}$$

The optimal solution of Eq.(1) yields a decision function of the following form,

$$\begin{aligned} f(\vec{x}) = sign\left(\vec{w}\cdot\Phi(\vec{x})+b\right) &= sign\left(\sum_{i=1}^{M} y_i\alpha_i\Phi(\vec{x}_i)\cdot\Phi(\vec{x})+b\right) \\ &= sign\left(\sum_{i=1}^{M} y_i\alpha_i K(\vec{x}_i,\vec{x})+b\right), \end{aligned} \tag{3}$$

where α_is are nonnegative Lagrange multipliers associated with training patterns, respectively. The solutions, α_is, are obtained from the dual problem of Eq.(1), which minimizes the convex quadratic objective function under constraints

$$\min_{0\le\alpha_i\le C} \; W(\alpha_i,b) = \frac{1}{2}\sum_{i,j=1}^{M}\alpha_i\alpha_j y_i y_j K(\vec{x}_i\cdot\vec{x}_j) - \sum_{i=1}^{M}\alpha_i + b\sum_{i=1}^{M} y_i\alpha_i.$$

The first-order conditions on $W(\alpha_i,b)$ are reduced to the Karush-Kuhn-Tucker (KKT) conditions,

$$\begin{aligned} \frac{\partial W(\alpha_i,b)}{\partial\alpha_i} &= \sum_{j=1}^{M} y_i y_j K(\vec{x}_i,\vec{x}_j)\alpha_j + y_i b - 1 = y_i\bar{f}(\vec{x}_i) - 1 = g_i, \\ \frac{\partial W(\alpha_i,b)}{\partial b} &= \sum_{j=1}^{M} y_j\alpha_j = 0, \end{aligned} \tag{4}$$

where $\bar{f}(\cdot)$ is the function inside the parentheses of *sign* in Eq.(3). The KKT complementarity condition, Eq. (4), partitions the training pattern set into three categories according to the corresponding α_is.

(a)	$g_i > 0$	$\rightarrow$	$\alpha_i = 0$	:	irrelevant patterns
(b)	$g_i = 0$	$\rightarrow$	$0 < \alpha_i < C$	:	margin support vectors
(c)	$g_i < 0$	$\rightarrow$	$\alpha_i = C$	:	error support vectors

Fig. 2 illustrates those categories (Cauwenberghs & Poggio, 2001; Pontil & Verri, 1998). The patterns belonging to (a) are out of the margins, thus irrelevant to training, while the patterns belonging to (b) and (c) are critical ones directly affecting training. They are called support vectors (SVs). The patterns of (b) are strictly on the margin hence they are called *margin SVs*. The patterns of (c) lie between two margins hence they are called *error SVs* but are not necessarily misclassified. (Note, that there is another type of SVs belonging to the category of error SVs. They are incorrectly labeled patterns that could be located very *far from the decision boundary*. We regard them as outliers which do not contribute to margin construction. We focus only on the SVs located around the decision boundary, not those located deep in the realm of the opposite class.) Going back to Eq.(3), we can now see that the decision function is a linear combination of kernels on only those critical training patterns (denoted as SVs) because the patterns corresponding to $\alpha_i = 0$ have no influence on the decision result:

$$f(\vec{x}) = sign\left(\sum_{i=1}^{M} y_i \alpha_i K(\vec{x}_i, \vec{x}) + b\right) = sign\left(\sum_{i \in \mathbf{SVs}} y_i \alpha_i K(\vec{x}_i, \vec{x}) + b\right). \quad (5)$$

Eq.(5) leads us to an attempt that reduces the whole training set to a subset of *would-be* SVs.

3 Neighborhood Property based Pattern Selection

To circumvent memory- and time-demanding SVM training, we propose a pre-processing algorithm, a neighborhood property based pattern selection algorithm (NPPS). The idea of the algorithm is to select only those patterns located around the decision boundary since they are the ones that contain the most information. Contrary to a usually employed "random sampling," this approach can be viewed as "informative or intelligent sampling." Fig. 3 conceptually shows the difference between NPPS and random sampling in selecting a subset of the training data. NPPS selects the patterns in the region around the decision boundary, while random sampling selects those from the whole input space. Obviously, no one knows how close a pattern is to the decision boundary until a classifier is built. However, we can infer the proximity ahead of training by utilizing neighborhood properties. *The first neighborhood property* is that "*a pattern located near the decision boundary*

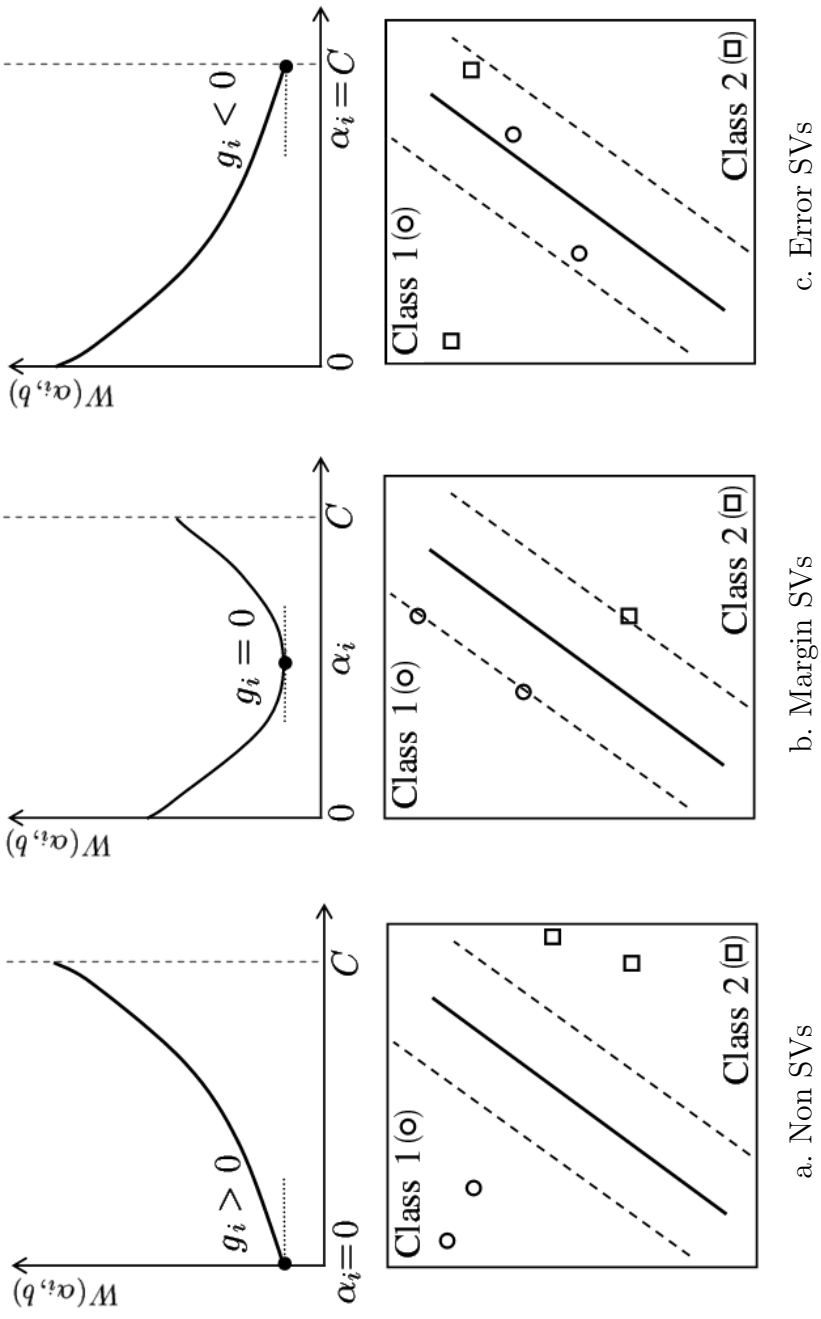
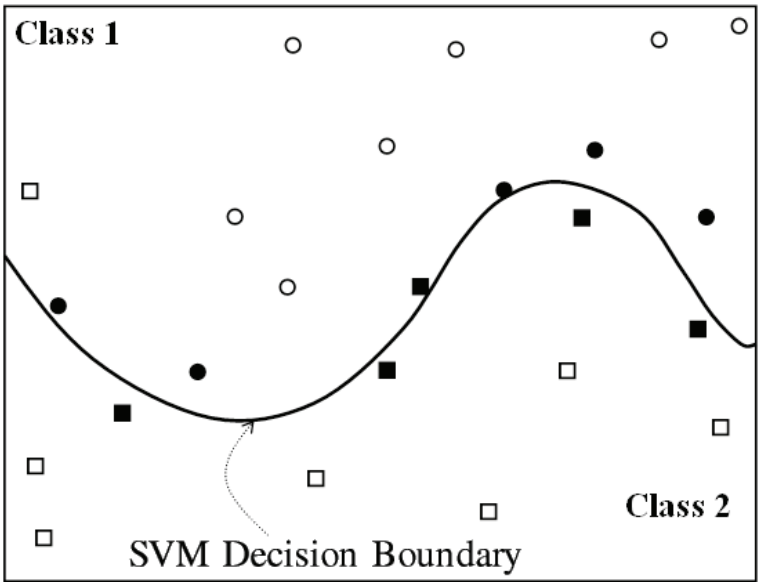
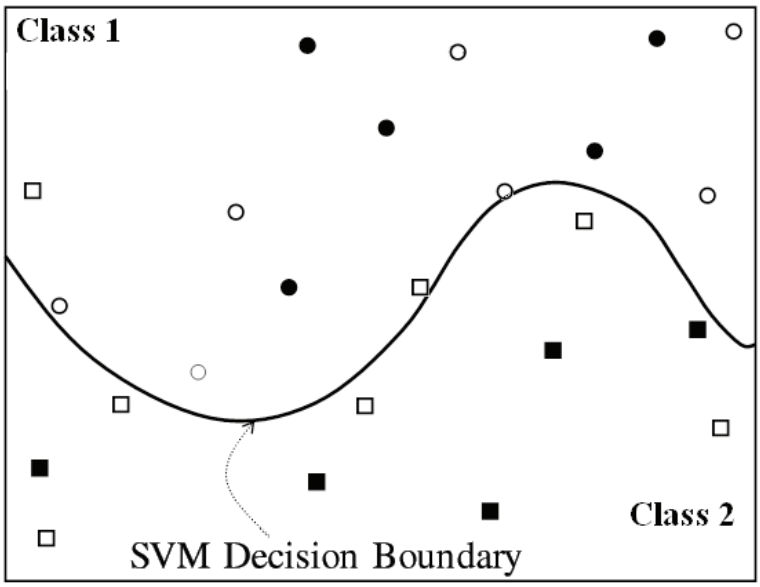


Figure 2: Three categories of training patterns



a. NPPS



b. Random Sampling

Figure 3: NPPS and random sampling select different subsets: outlined circles and squares are the patterns belonging to class 1 and class 2, respectively. Black solid circles and squares are the selected patterns.

tends to have more heterogeneous neighbors in their class membership." A well-known entropy concept can be utilized for the measurement of heterogeneity of class labels among k-nearest neighbors. And the measure will lead us to estimate the proximity accordingly. Here, we define a measure by using (negative) entropy,

$$\text{Neighbors\_Entropy } (\vec{x}, k) = \sum_{j=1}^{J} P_j \cdot log_J \frac{1}{P_j},$$

where j indicates a particular class out of J classes, and P_j is defined as k_j/k, where k_j is the number of the neighbors belonging to class j. In most cases, a pattern with a positive value of "*Neighbors_Entropy* $(\vec{x}, k)$" is close to the decision boundary, thus selected. Those patterns are likely to be SVs, which correspond to the margin SVs in Fig. 2.b or the error SVs in Fig. 2.c. Among the patterns having a positive value of Neighbors_Entropy $(\vec{x}, k)$, noisy patterns are also present. Here, let us define an *overlap region* as a hypothetical region in feature space shared by both classes, and *overlap patterns* as the patterns in that region. [2] A set of overlap patterns contains the patterns not only located on the *right* side of the decision boundary but also on the *other* side of it. The latter denotes *noisy patterns*, which should be identified and removed as much as possible, because they are more likely to be the error SVs that would be misclassified. To remove this noisy patters, we take *the second neighborhood property.* "*An overlap pattern or an outlier tends to belong to a different class from its neighbors.*" If a pattern's own label is different from the majority label of its neighbors, it is likely to be incorrectly labelled. The measure "*Neighbors_Match* $(\vec{x}, k)$" is defined as the ratio of neighbors whose label matches that of $\vec{x}$,

$$\text{Neighbors\_Match } (\vec{x}, k) = \frac{|\{\vec{x}' | label(\vec{x}') = label(\vec{x}),\ \vec{x}' \in kNN(\vec{x})\}|}{k},$$

where $kNN(\vec{x})$ is the set of k nearest neighbors of $\vec{x}$. The patterns with a small Neighbors_Match $(\vec{x}, k)$ value are likely to be the ones incorrectly labelled. In that point of view, Neighbors_Match can be interpreted as a confidence for k-nearest neighbor classification. Only the patterns satisfying the two conditions, Neighbors_Entropy $(\vec{x}, k) > 0$ and Neighbors_Match $(\vec{x}, k) \geq \frac{1}{J}$, are selected.

[2] We defer the finer definitions for overlap region and overlap patterns until Section 4.1.

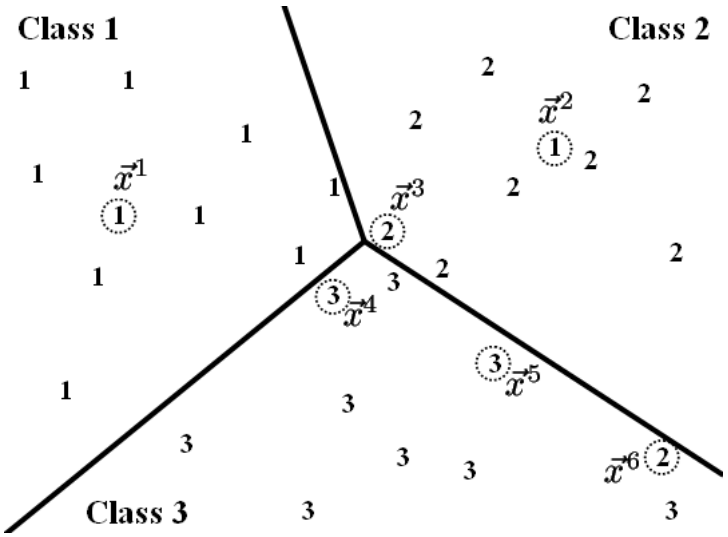


Figure 4: A toy example: The numbers scattered in the figure (i.e., 1, 2, and 3) stand for the class labels of the patterns. The parameters are assumed to be $J = 3$ and $k = 6$. For representational simplicity, we consider only six patterns (out of 29) marked by dotted circles. Table 1 shows the patterns selected by the proposed measures presenting details; the values of P_j, Neighbrs_Entropy, and Neighbors_Match.

Fig. 4 shows a toy example of how patterns are selected by the proposed measures. The numbers scattered in the figure (i.e., 1, 2, and 3) stand for the class labels of the patterns. We assume $J = 3$ and $k = 6$. For representational simplicity, we consider only six patterns marked by dotted circles. Table 1 presents the values of P_j, Neighbrs_Entropy, and Neighbors_Match for these marked patterns. Let us consider $\vec{x}^1$ first. $\vec{x}^1$ is remote from the decision boundary belonging to class 1, and is thus surrounded by the neighbors all belonging to class 1. $\vec{x}^1$ is not selected since it does not satisfy the condition of Neighbors_Entropy. Meanwhile, $\vec{x}^2$ resides in a deep region of class 2, but it is a noise pattern labelled as class 1. Since the composition of class membership of its neighbors is homogeneous (all of them belong to class 2), it also has a zero value of Neighbors_Entropy. Therefore, it is excluded from selection. However, the case of the pattern $\vec{x}^2$ is different from the case of the pattern $\vec{x}^1$; for the pattern $\vec{x}^2$, all its neighbors belong to a different class from the class of $\vec{x}^2$, while for the pattern $\vec{x}^1$, all its neighbors belong to

Table 1: A toy example: The patterns selected by Neighbors_Entropy and Neighbors_Match. j^* is the class label of pattern $\vec{x}^i$, and j^k is the label of its k^{th} nearest neighbor. Refer to Fig. 4.

	j^*	j^1	j^2	j^3	j^4	j^5	j^6	P_1	P_2	P_3	Neighbors _Entropy	Neighbors _Match	Selected Patterns
		(Class Labels of Neighbors)											
$\vec{x}^1$	1	1	1	1	1	1	1	6/6	0/6	0/6	0	6/6	X
$\vec{x}^2$	1	2	2	2	2	2	2	0/6	6/6	0/6	0	0/6	X
$\vec{x}^3$	2	1	1	2	2	3	3	2/6	2/6	2/6	1	2/6	◯
$\vec{x}^4$	3	3	3	2	2	3	1	1/6	2/6	3/6	0.9227	3/6	◯
$\vec{x}^5$	3	3	3	3	3	3	2	0/6	1/6	5/6	0.4100	5/6	◯
$\vec{x}^6$	2	3	3	3	3	3	2	0/6	1/6	5/6	0.4100	1/6	X

the same class that $\vec{x}^1$ belongs to. We can differentiate two cases by Neighbors_Match, which is 1 for $\vec{x}^1$ and 0 for $\vec{x}^2$. In any case, the remote patterns from the decision boundary are screened out by the proposed measures. On the other hand, the pattern $\vec{x}^3$ is close to the decision boundary, and the composition of the neighbors shows heterogeneous class membership; two of them belong to class 1, another two belong to class 2, and the rest belong to class 3. This leads to Neighbors_Entropy = 1 and Neighbors_Match = 1/3, and consequently satisfies the conditions. Similarly, the patterns $\vec{x}^4$ and $\vec{x}^5$ are chosen. From the example we can see that the selected patterns are distributed near the decision boundary and almost correctly labeled.

However, NPPS takes $\mathrm{O}(M^2)$ to evaluate kNNs for M patterns, so the pattern selection process itself can be time-consuming. To accelerate the pattern selection procedure, we consider *the third neighborhood property*. "*The neighbors of a pattern located near the decision boundary tend to be located near the decision boundary as well.*" Assuming the property, one may narrow the scope of computation from all the training patterns to the patterns near the decision boundary. Only the neighbors of a pattern satisfying Neighbors_Entropy $(\vec{x}, k) > 0$ are evaluated in the next step. This lazy evaluation can reduce the practical time complexity from $\mathrm{O}(M^2)$ to $\mathrm{O}(vM)$ where v is the number of patterns in the overlap region. In most practical problems, $v < M$ holds. In addition, any algorithm on efficient nearest neighbor searching can be incorporated into the proposed method in order for further

Table 2: Notation

Symbol	Meaning
$\mathbf{D}$	the original training set whose cardinality is M
$\mathbf{D}_{\mathbf{e}}^{i}$	the evaluation set at i^{th} step
$\mathbf{D}_{\mathbf{o}}^{i}$	a subset of $\mathbf{D}_{\mathbf{e}}^{i}$, the set of patterns to be "expanded" from $\mathbf{D}_{\mathbf{e}}^{i}$ each element of which will compute its k nearest neighbors to constitute the next evaluation set, $\mathbf{D}_{\mathbf{e}}^{i+1}$
$\mathbf{D}_{\mathbf{x}}^{i}$	a subset of $\mathbf{D}_{\mathbf{e}}^{i}$, the set of patterns "not to be expanded" from $\mathbf{D}_{\mathbf{e}}^{i}$, or $\mathbf{D}_{\mathbf{x}}^{i} = \mathbf{D}_{\mathbf{e}}^{i} - \mathbf{D}_{\mathbf{o}}^{i}$
$\mathbf{D}_{\mathbf{s}}^{i}$	the set of "selected" patterns from $\mathbf{D}_{\mathbf{o}}^{i}$ at i^{th} step
$\mathbf{S}_{\mathbf{o}}^{i}$	the accumulated set of expanded patterns, $\bigcup_{j=0}^{i-1} \mathbf{D}_{\mathbf{o}}^{j}$
$\mathbf{S}_{\mathbf{x}}^{i}$	the accumulated set of non-expanded patterns, $\bigcup_{j=0}^{i-1} \mathbf{D}_{\mathbf{x}}^{j}$
$\mathbf{S}^{i}$	the accumulated set of selected patterns, $\bigcup_{j=0}^{i-1} \mathbf{D}_{\mathbf{s}}^{j}$ the last of which $\mathbf{S}^{N}$ is the reduced training pattern set
$k\mathbf{NN}(\vec{x})$	the set of k nearest neighbors of $\vec{x}$

reduction of the time complexity. There is a considerable amount of literature about efficient searching for nearest neighbors. Some of them attempt to save distance computation time (Grother et al., 1997; Short & Fukunaga, 1981), while others attempt to avoid redundant searching time (Bentley, 1975; Friedman et al., 1977; Guttman, 1984; Masuyama et al., 1999). Apart from them, there are many sophisticated NN classification algorithms such as *approximated* (Arya et al., 1998; Indyk, 1998), *condensed* (Hart, 1968), and *reduced* (Gates, 1972). See also (Berchtold et al., 1997; Borodin et al., 1999; Ferri et al., 1999; Johnson et al., 2000; Kleingberg, 1997; Tsaparas, 1999).

The time complexity analysis for the fast NPPS can be found in `http://www.kyb.mpg.de/publication.html?user=shin` and Shin & Cho (2003a), and a brief empirical result in Appendix A. The algorithm and related notations are shown in Fig. 5 and Table 2.

NPPS (D, k) {

[0] Initialize $\mathbf{D}_\mathbf{e}^0$ with randomly chosen patterns from $\mathbf{D}$.
Constants, k and J, are given. Initialize i and various sets as follows:
$i \leftarrow 0, \quad \mathbf{S}_\mathbf{o}^0 \leftarrow \emptyset, \quad \mathbf{S}_\mathbf{x}^0 \leftarrow \emptyset, \quad \mathbf{S}^0 \leftarrow \emptyset.$

while $\mathbf{D}_\mathbf{e}^i \neq \emptyset$ **do**

[1] Choose $\vec{x}$ satisfying [Expanding Criteria].
$\mathbf{D}_\mathbf{o}^i \leftarrow \{\vec{x} \mid Neighbors\_Entropy\,(\vec{x}, k) > 0,\ \vec{x} \in \mathbf{D}_\mathbf{e}^i\}.$
$\mathbf{D}_\mathbf{x}^i \leftarrow \mathbf{D}_\mathbf{e}^i - \mathbf{D}_\mathbf{o}^i.$

[2] Select $\vec{x}$ satisfying [Selecting Criteria].
$\mathbf{D}_\mathbf{s}^i \leftarrow \{\vec{x} \mid Neighbors\_Match\,(\vec{x}, k) \geq 1/J,\ \vec{x} \in \mathbf{D}_\mathbf{o}^i\}.$

[3] Update the pattern sets.
$\mathbf{S}_\mathbf{o}^{i+1} \leftarrow \mathbf{S}_\mathbf{o}^i \cup \mathbf{D}_\mathbf{o}^i$: the expanded,
$\mathbf{S}_\mathbf{x}^{i+1} \leftarrow \mathbf{S}_\mathbf{x}^i \cup \mathbf{D}_\mathbf{x}^i$: the non-expanded,
$\mathbf{S}^{i+1} \leftarrow \mathbf{S}^i \cup \mathbf{D}_\mathbf{s}^i$: the selected.

[4] Compute the next evaluation set $\mathbf{D}_\mathbf{e}^{i+1}$.
$\mathbf{D}_\mathbf{e}^{i+1} \leftarrow \bigcup_{\vec{x} \in \mathbf{D}_\mathbf{o}^i} k\mathbf{NN}(\vec{x}) - (\mathbf{S}_\mathbf{o}^{i+1} \cup \mathbf{S}_\mathbf{x}^{i+1}).$

[5] $i \leftarrow i + 1.$

end

return $\mathbf{S}^i$

}

Figure 5: NPPS

4 How to Determine the Number of Neighbors

In this section, we briefly introduce a heuristic for determining the value of k. Too large a value of k results in too many patterns being selected. Consequently, we will achieve little effect of pattern selection. Too small a value of k, on the other hand, leads to few patterns selected. However, it may degrade SVM accuracy since there are more chances to miss important patterns, i.e., the would-be support vectors. The dilemma about k will not be symmetrical because a serious loss of accuracy is not likely to be well compensated for by a benefit of down-sized training set. This point relates to our idea; the selected pattern set should be large enough at least to contain the patterns in the overlap region. Therefore, it is first required to estimate how many training patterns reside in the overlap region. Based on the estimation, the next step is to enlarge the value of k until the size of the corresponding set covers the estimate. Under this condition, the minimum value of k will be regarded as optimal unless SVM accuracy degrades.

In the following sub-sections, we will first identify the overlap region $\mathbf{R}$ and the overlap set $\mathbf{V}$. Second, an estimate of the lower bound on the cardinality of $\mathbf{V}$ will be given. Third, we will define Neighbors_Entropy set $\mathbf{B}_k$ with respect to the value of k, and some related properties as well. Finally, a systematic procedure for determining the value of k will be provided.

4.1 Overlap Region and Overlap Set

Consider a two-class classification problem (see Fig. 6),

$$f(\vec{x}) = \begin{cases} \vec{x} \rightarrow C_1 & \text{if} \quad f(\vec{x}) > 0, \\ \vec{x} \rightarrow C_2 & \text{if} \quad f(\vec{x}) < 0, \end{cases} \tag{6}$$

where $f(\vec{x})$ is a classifier and $f(\vec{x}) = 0$ is the decision boundary. Let us define *noisy overlap patterns* (NOPs) as the patterns that are located on the *wrong* side of the decision boundary. They are shown in Fig. 6 as squares located above $f(\vec{x}) = 0$ and circles located below $f(\vec{x}) = 0$. Let $\mathbf{R}$ denote *overlap region*– a hypothetical region where NOPs reside, the convex-hull of NOPs, enclosed by the dotted lines in Fig. 6. Similarly, let us define *correct overlap patterns* (COPs) as the patterns that are enveloped by the convex hull, yet in

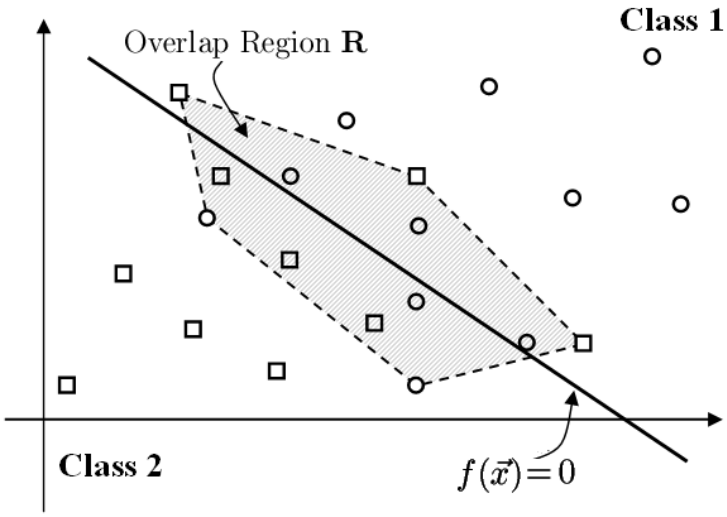


Figure 6: Two class classification problem where the circles belong to class 1 while the squares belong to class 2. The area enclosed by the dotted lines is defined as overlap region **R**. The area comprises the overlap set **V** of NOPs and COPs.

the *right* class-side. Note that **R** is defined by NOPs, but it contains not only NOPs but also COPs. Let **V** denote the *overlap set* defined as the intersection of **D** and **R**, i.e. the subset of **D** which comprises NOPs and COPs. There are six NOPs and six COPs in Fig. 6. The cardinality of **V** is denoted as v.

4.2 Size Estimation of Overlap Set

Now, we estimate v. Let $P_{\mathbf{R}}(\vec{x})$ denote the probability that a pattern $\vec{x}$ falls in the region **R**. Then, we can calculate the expected value of v from the given training set **D** as

$$v = MP_{\mathbf{R}}(\vec{x}), \tag{7}$$

where M is the number of the training patterns, say, $M = |\mathbf{D}|$. The probability $P_{\mathbf{R}}(\vec{x})$ can be dissected class-wise such as

$$P_{\mathbf{R}}(\vec{x}) = \sum_{j=1}^{2} P(\vec{x} \in \mathbf{R}, C_j), \tag{8}$$

where $P(\vec{x} \in \mathbf{R}, C_j)$ is the joint probability of $\vec{x}$ belonging to class C_j and lying in $\mathbf{R}$. Note that $P_{\mathbf{R}}(\vec{x})$ also can be interpreted as the probability of $\vec{x}$ being COPs or NOPs. Further, if the region $\mathbf{R}$ is divided into

$$\mathbf{R}_1 = \{\vec{x} \in \mathbf{R} \mid f(\vec{x}) \geq 0\} \text{ and } \mathbf{R}_2 = \{\vec{x} \in \mathbf{R} \mid f(\vec{x}) < 0\}, \tag{9}$$

Eq. (8) can be rewritten as

$$\begin{aligned} P_{\mathbf{R}}(\vec{x}) &= P(\vec{x} \in \mathbf{R}, C_1) + P(\vec{x} \in \mathbf{R}, C_2) \\ &= P(\vec{x} \in \mathbf{R}_1 \cup \mathbf{R}_2, C_1) + P(\vec{x} \in \mathbf{R}_1 \cup \mathbf{R}_2, C_2) \\ &= \underbrace{\{P(\vec{x} \in \mathbf{R}_1, C_2) + P(\vec{x} \in \mathbf{R}_2, C_1)\}}_{\textbf{(a)}} \\ &+ \underbrace{\{P(\vec{x} \in \mathbf{R}_1, C_1) + P(\vec{x} \in \mathbf{R}_2, C_2)\}}_{\textbf{(b)}}. \end{aligned} \tag{10}$$

The parentheses (a) and (b) in the last row denote the probabilities of the patterns located in $\mathbf{R}$ which are incorrectly and correctly classified– NOPs and COPs, respectively. Since all NOPs are the incorrectly classified patterns, (a) can be estimated from the misclassification error rate P_{error} of the classifier $f(\vec{x})$. On the contrary, it is not easy to estimate (b).[3] Therefore, what we can do in practice is to infer the lower bound of it. Generally, the following inequality holds,

$$P(\vec{x} \in \mathbf{R}_j, C_j) \geq P(\vec{x} \in \mathbf{R}_j, C_i), \quad j \neq i. \tag{11}$$

Based on that, Eq. (10) can be simplified as

$$P_{\mathbf{R}}(\vec{x}) \geq 2P_{error} \tag{12}$$

and the lower bound of v becomes

$$v \ \geq \ v^{LOW} = 2MP_{error} \tag{13}$$

from Eq. (7).

[3]Of course, if $\mathbf{R}_1$ and $\mathbf{R}_2$ contain roughly the same number of correct and incorrect patterns, the probabilities of (a) and (b) become similar. But this assumption will only work when both class distributions follow the uniform distribution.

4.3 Pattern Set with Positive Neighbors_Entropy

Let us define $\mathbf{B}_k$ given a specified value of k,

$$\mathbf{B}_k = \{\vec{x} \mid \text{Neighbors\_Entropy}(\vec{x}, k) > 0,\ \vec{x} \in \mathbf{D}\}. \tag{14}$$

Note that $\mathbf{B}_k$ is the union of $\mathbf{D}_{\mathbf{o}}^i$'s, $\mathbf{B}_k = \bigcup_{i=0}^{N} \mathbf{D}_{\mathbf{o}}$ where N is the total number of iterations in the algorithm NPPS (see Fig.5). The following property of $\mathbf{B}_k$ leads to a simple procedure.

Lemma 1 *A Neighbors_Entropy set* $\mathbf{B}_k$ *is a subset of* $\mathbf{B}_{k+1}$*:*

$$\mathbf{B}_k \subseteq \mathbf{B}_{k+1} \qquad 2 \leq k \leq M-2. \tag{15}$$

Proof: Denote P_j^k as the probability that k_j out of k nearest neighbors belong to class C_j. If $\vec{x} \in \mathbf{B}_k$, then it means Neighbors_Entropy $(\vec{x}, k) > 0$. A positive Neighbors_Entropy is always accompanied with $P_j^k = \frac{k_j}{k} < 1,\ \forall j$. Therefore,

$$k_j < k,\ \forall j.$$

Adding 1 to both sides yields

$$(k_j + 1) < (k + 1),\ \forall j.$$

Suppose the $(k+1)^{th}$ nearest neighbor of $\vec{x}$ belongs to C_{j^*}. Then, $(k_{j^*} + 1) < (k + 1)$ holds for j^* while $k_j < (k + 1)$ for $j \neq j^*$. The inequalities lead to $P_{j^*}^{k+1} < 1$ and $P_{j \neq j^*}^{k+1} < 1$, respectively. As a consequence, we have a positive Neighbors_Entropy of $\vec{x}$ in the case of the $(k + 1)^{th}$ nearest neighbor as well. Therefore, Neighbors_Entropy $(\vec{x}, k+1) > 0$ which indicates $\vec{x} \in \mathbf{B}_{k+1}$.

From Lemma 1, it follows that b_k, the cardinality of $\mathbf{B}_k$, is an increasing function of k.

4.4 Procedure for Determining the Number of Neighbors

$\mathbf{B}_k$ larger than $\mathbf{V}$ merely increases the SVM training time by introducing redundant training patterns. In contrast, $\mathbf{B}_k$ smaller than $\mathbf{V}$ could degrade

[1] Estimate P_{error}:
$\hat{P}_{error}$ is calculated based on the training error rate of *1*-NN rule.

[2] Calculate the lower bound of v according to Eq. (13):
$v \geq \ v^{LOW} = 2M\hat{P}_{error}$.

[3] Find k^* according to Eq. (16):
$k^* = \arg\min \{ \ |P^{svm}_{error}(k) - P^{svm}_{error}(k+1)| \leq \epsilon, \ k \geq k^{\min}\}$.
where $k^{\min} = \min \{k \mid b_k \geq v^{LOW}, \ \ k \geq 2 \ \}$
and ϵ is a trivial value.

Figure 7: Procedure to determine the value of k.

the SVM accuracy. Therefore, our objective is to find the smallest $\mathbf{B}_k$ that covers $\mathbf{V}$.

Let us define $k^{\min}$ using the lower bound of v from Eq.(13),

$$k^{\min} = \min \{k \mid b_k \geq v^{LOW}, \ \ k \geq 2 \ \}.$$

From Lemma 1, we know that b_k is an increasing function of k. Therefore, it is not necessary to evaluate the values of k less than $k^{\min}$. Instead, we check the stabilization of SVM training error rate P^{svm}_{error} only for the k's larger than $k^{\min}$ by increasing the value little by little. The optimal value of k is then chosen as

$$k^* = \arg\min \{ \ |P^{svm}_{error}(k) - P^{svm}_{error}(k+1)| \leq \epsilon, \ k \geq k^{\min}\}, \qquad (16)$$

where ϵ is set to a trivial value. Eq.(16) requires several dry-runs of SVM training. However, the runs are not likely to impose heavy computational burden since the strong inequality $b_k << M$ holds for the first smallest k's and hence the size of the training set (the selected pattern set) is small. The procedure is summarized in Fig. 7. [4] The main contribution of the proposed procedure is to limit the search space of k by means of the lower bound estimation of the size of overlap set. More details can be found in Shin & Cho (2003b) and some empirical results in Appendix B.

[4]There are several benefits of using *1*-NN rule as a P_{error} estimator in step 1 since it is a local learning algorithm, simple, and computationally efficient. One can skip the step 1 if an approximate P_{error} of the problem is given *a priori*.

5 Experiments

We applied NPPS to various kinds of datasets: artificial datasets, benchmarking datasets, and a real-world marketing dataset. The following three sub-sections present the experimental results in order.

5.1 Artificial Datasets

The first problem was drawn from four Gaussian densities: the Continuous XOR problem. A total of 600 training patterns, 300 from each class, were generated. The classes, C_1 and C_2, were defined as

$$C_1 = \left\{\vec{x} \mid \vec{x} \in N_{1A} \cup N_{1B}, \begin{bmatrix} -3 \\ -3 \end{bmatrix} \leq \vec{x} \leq \begin{bmatrix} 3 \\ 3 \end{bmatrix}\right\},$$
$$C_2 = \left\{\vec{x} \mid \vec{x} \in N_{2A} \cup N_{2B}, \begin{bmatrix} -3 \\ -3 \end{bmatrix} \leq \vec{x} \leq \begin{bmatrix} 3 \\ 3 \end{bmatrix}\right\},$$

where

$$N_{1A} = \left\{\vec{x}|N\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}\right)\right\}, \quad N_{1B} = \left\{\vec{x}|N\left(\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}\right)\right\},$$
$$N_{2A} = \left\{\vec{x}|N\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}\right)\right\}, \quad N_{2B} = \left\{\vec{x}|N\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}\right)\right\}.$$

See also Fig.11.1.a. The second problem is the Sine Function problem. The input patterns were generated from a two-dimensional uniform distribution, and then the class labels were determined by whether the pattern was located above or below a sine decision function:

$$C_1 = \left\{\vec{x} \mid x_2 > \sin\left(3x_1 + 0.8\right)^2, \begin{bmatrix} 0 \\ -2.5 \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 2.5 \end{bmatrix}\right\},$$
$$C_2 = \left\{\vec{x} \mid x_2 \leq \sin\left(3x_1 + 0.8\right)^2, \begin{bmatrix} 0 \\ -2.5 \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 2.5 \end{bmatrix}\right\}.$$

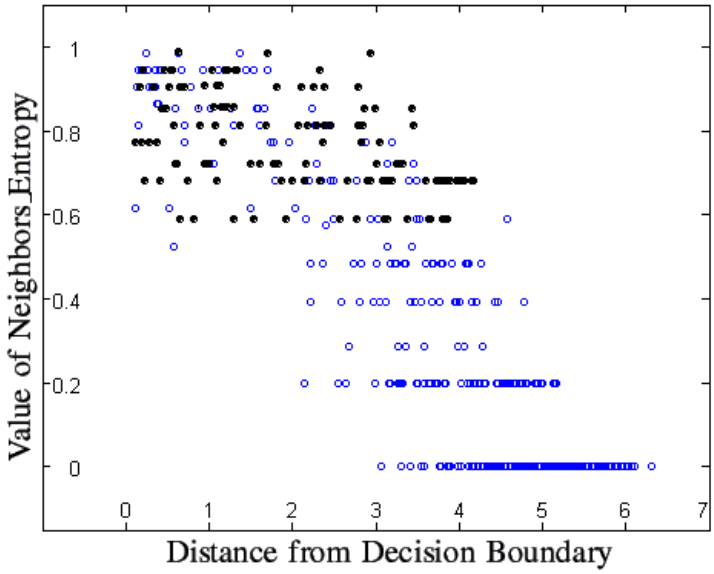
To make the density near the decision boundary thicker, four different Gaussian noises were added along the decision boundary, i.e., $N(\vec{\mu}, s^2 I)$ where $\vec{\mu}$ is an arbitrary point on the decision boundary and s is a Gaussian width parameter ($s = 0.1,\ 0.3,\ 0.8,\ 1.0$). A total of 500 training patterns were generated including noises. Fig.11.1.b shows the problem.

For both problems, Fig. 8 presents the relationship between distance from decision boundary to a pattern and its value of Neighbors_Entropy.

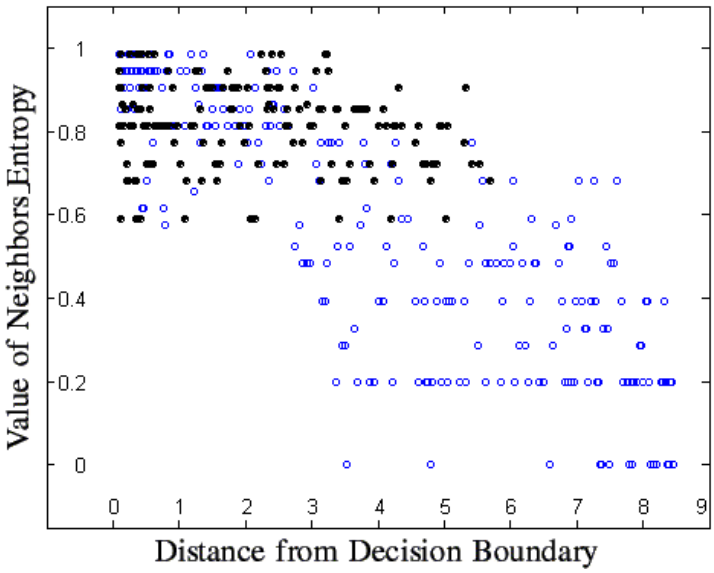
The closer to the decision boundary a pattern is, the higher value of Neighbors_Entropy it has. Both sub-figures of Fig. 8 assure that Neighbors_Entropy is pertinent to estimating the proximity to the decision boundary. Among the patterns with positive Neighbors_Entropy values, the ones which meet the Neighbors_Match condition are selected. They are depicted as solid circles against outlined ones. Note that the solid circles are distributed nearer the decision boundary. The distribution of the selected patterns is shown in Fig.11.3.a and Fig.11.3.b.

NPPS changes the size as well as the distribution of the training set. Such changes may change the optimal value of hyper-parameter that brings the best result. Thus, we first observed the effect of change of training set; The SVM performance was measured under various combinations of hyper-parameters such as $(C, \sigma) \in \{0.1, 1.0, 10, 100, 1000\} \times \{0.25, 0.5, 1, 2, 3\}$ where C and σ indicate misclassification tolerance and Gaussian kernel width, respectively. For each of the artificial problems, a total of 1000 test patterns were generated from the statistically identical distributions to its original training set. We compared the test error rates (%) of SVM when *trained with all patterns* (ALL), *trained with random patterns* (RAN), and *trained with the selected patterns* (SEL).

Fig. 9 depicts the test error rate over the hyper-parameter variation for Continuous XOR problem. The most pronounced feature is the higher sensitivity of SEL to hyper-parameter variation than before (ALL). It may be caused by the fact that the patterns selected by NPPS are mostly distributed in the "narrow" region along the decision boundary. In Fig.9.a, for instance, SEL shows a sudden rise of the test error rate for σ larger than a certain value when C is fixed, and similarly, a sharp descent after a certain value of C when σ is fixed. An interesting point is that SEL can always reach a performance comparable to that of ALL by adjusting the value of the hyper-parameters. Now, we compare SEL and RAN from Fig.9.b. We used the same number of random samples as that of the patterns selected by NPPS. Therefore, there is no difference in the size of training set but only a difference in data distribution. When compared with SEL, RAN is less sensitive to hyper-parameter variation because the patterns are distributed all over the region. However, note that RAN never performs as well as ALL since random sampling inevitably misses many patterns of importance to SVM training. See the best result of RAN in Table 3. Fig.10 shows similar results for Sine

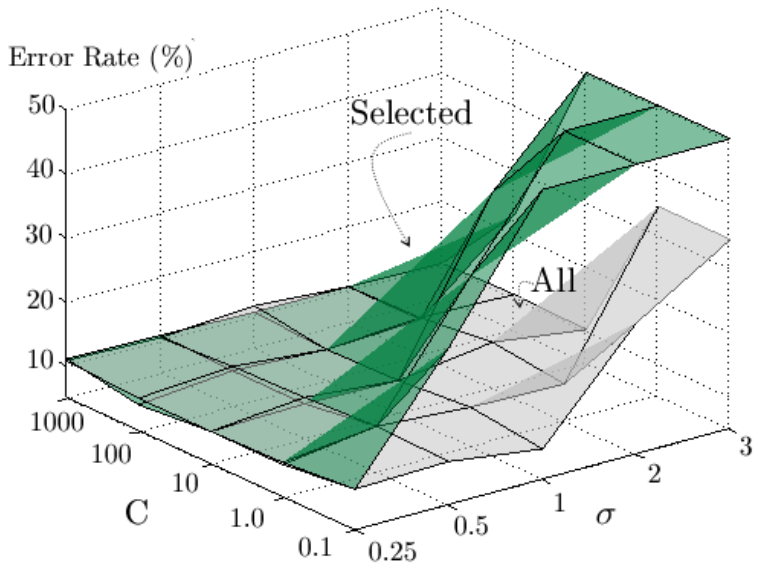


a. Continuous XOR problem

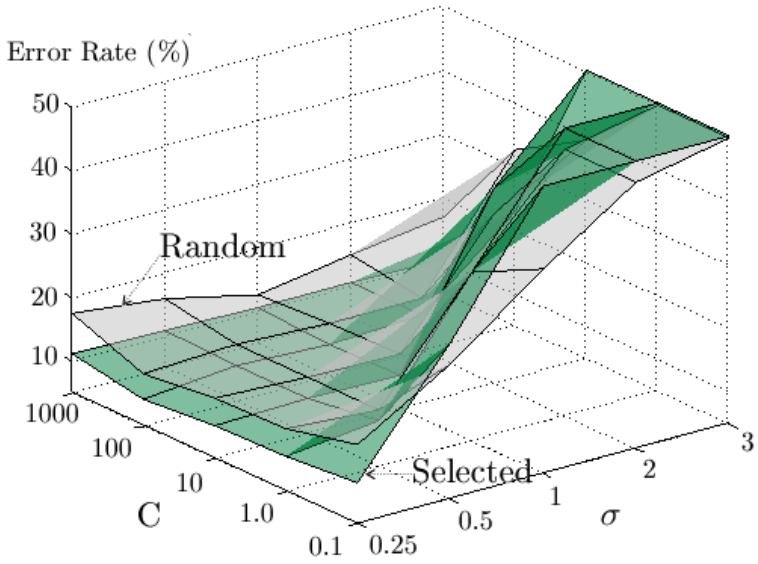


b. Sine Function problem

Figure 8: The relationship between the distance from decision boundary and Neighbors_Entropy: The closer to the decision boundary a pattern is, the higher value of Neighbors_Entropy it has. The selected patterns are depicted as solid circles against outlined ones.



a. **SEL** VS. **ALL**



b. **SEL** VS. **RAN**

Figure 9: Performance comparison over hyper-parameter variation: Continuous XOR problem

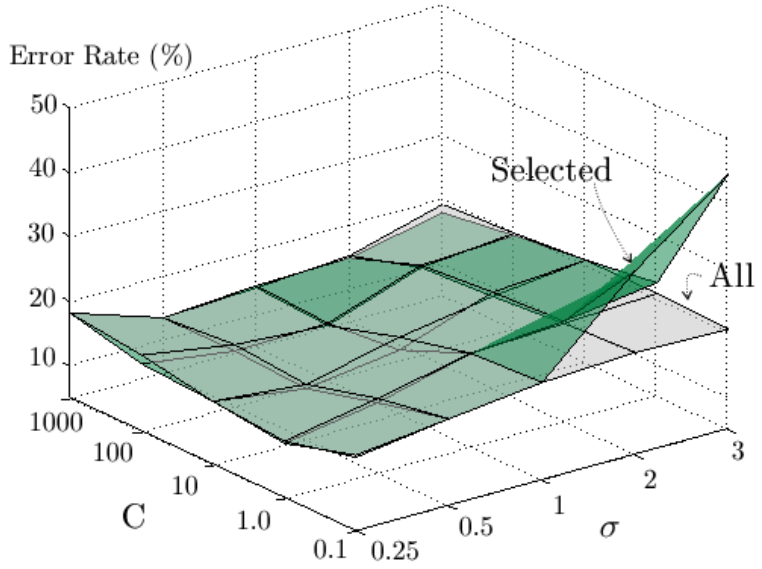
Table 3: Best result comparison: **ALL** vs. **RAN** vs. **SEL**

	Continuous XOR			Sine Function		
	ALL	**RAN**	**SEL**	**ALL**	**RAN**	**SEL**
(C, σ)	(10, 0.5)	(100, 1)	(100, 0.25)	(10, 0.5)	(0.1, 0.25)	(10, 0.5)
Execution Time (sec)	454.83	3.02	3.85	267.76	8.97	8.79
Num. of Training Patterns	600	180	180	500	264	264
Num. of Support Vectors	167	68	84	250	129	136
Test Error (%)	9.67	12.33	9.67	13.33	16.34	12.67
McNemar's Test (p-value)	–	0.08	1.00	–	0.12	0.89

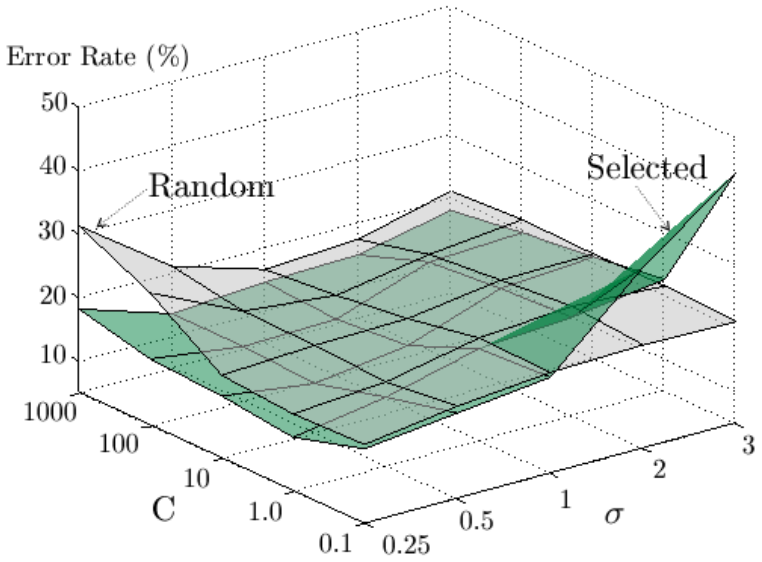
Function problem.

Table 3 summarizes the best results of **ALL**, **SEL**, and **RAN** for both problems. First, compared with the training time of **ALL**, that of **SEL** is little more than trivial because of the reduced size of training set. For both artificial problems, a standard QP solver, i.e., Gunn's SVM MATLAB toolbox was used. Considering that model selection always involves multiple trials, an individual's reduction in training time can amount to huge time-saving. Second, compared with **RAN**, **SEL** achieved an accuracy on a similar level to the best model of **ALL** while **RAN** could not. To show that there was no significant difference between **ALL** and **SEL**, we conducted the McNemar's test (Dietterich, 1998). The p-values between **ALL** and **RAN**, and also between **ALL** and **SEL** are presented in the last row of the table. In principle, the McNemar's test determines whether classifier A is better than classifier B. A p-value of zero indicates a significant difference between A and B, while a value of one indicates no significant difference. Although the p-value between **ALL** and **RAN** is not less than 5% in each problem, one can still compare its degree of difference from the p-value between **ALL** and **SEL** in statistical terms.

Fig.11 visualizes the experimental results of Table 3 on both problems. The sub-figures, 1, 2 and 3, indicate the results of **ALL**, **RAN**, and **SEL** in order. The decision boundary is depicted as a solid line and the margin as a dotted line. Support vectors are outlined. The decision boundary of **ALL** looks more alike to that of **SEL** than to that of **RAN**. That explains why **SEL** produced



a. **SEL** VS. **ALL**



b. **SEL** VS. **RAN**

Figure 10: Performance comparison over hyper-parameter variation: Sine Function problem

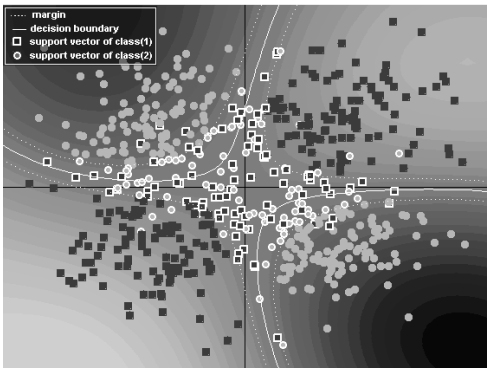
similar results to ALL.

5.2 Bench-Marking Datasets

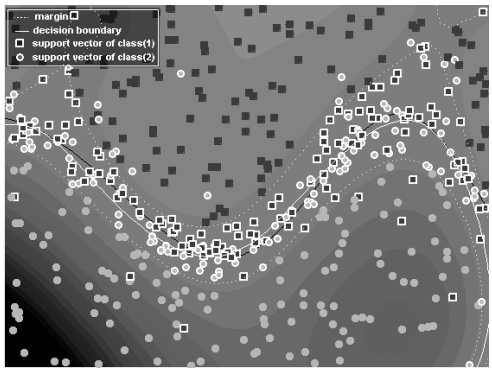
The following Table 5.2 presents the summary of the comparison results on various real-world bench-marking datasets (MNIST database; UCI repository) including the artificial datasets in the previous section. Again, SEL is compared with ALL and RAN. And another method is added as a competitor, SVM-KM (Almeida et al., 2000), which is a preprocessing method that reduces the training set based on k-means clustering algorithm.

The hyper-parameters of SVM were chosen based on the best results via validation: C, σ, and p, indicate the misclassification tolerance, the width of RBF kernel, and the order of polynomial kernel, respectively. The parameter of SEL (NPPS), the number of neighbors k, was determined by the procedure in Section 4 and Appendix B. See also Shin & Cho (2003b). And the parameter of SVM-KM, the number of clusters k, was set to 10% of the number of training patterns as recommended in Almeida et al. (2000). For the MNIST dataset, we chose three binary classification problems; they are known as most difficult due to the similar shapes of the digits, for instance, digit '3' looks similar with digit '8'. Most of the experiments were run on Pentium 4 with 1.9 GHz 512 RAM, whereas Pentium 4 with 3.0 GHz 1024 RAM for the MNIST problems. A standard QP solver lacks of memory-capacity to handle the real-world datasets. So we chose an iterative SVM solver, OSU SVM Classifier Toolbox (Kernel Machines Organization) after brief comparison about scalability with another candidate RSVM (Lee & Mangasarian, 2001). Refer to Appendix C for the comparison on OSU SVM Classifier vs. RSVM.

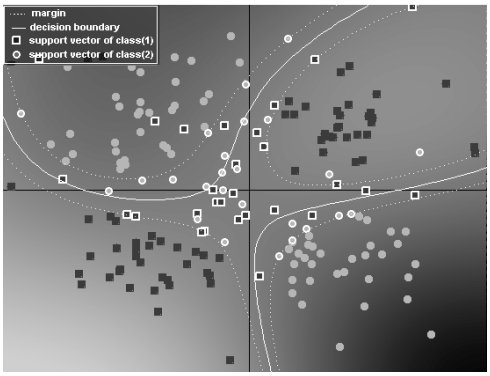
In the table, N/A denotes 'not available' results. For ease of comparison, the computational time, preprocessing or SVM training, is shown as a ratio to the SVM training time of ALL. The best two results are represented as boldface in test error rate. Similarly, the statistically most similar result with ALL is represented as boldface in McNemar's p-value. The results show that the preprocessing either by SEL or by SVM-KM is of great benefit to SVM training in reducing the training time. Particularly, if multiple running of SVM is required one can take the benefit as multiple times as required. However, SVM-KM took longer than SEL, and even worse it run out of memory



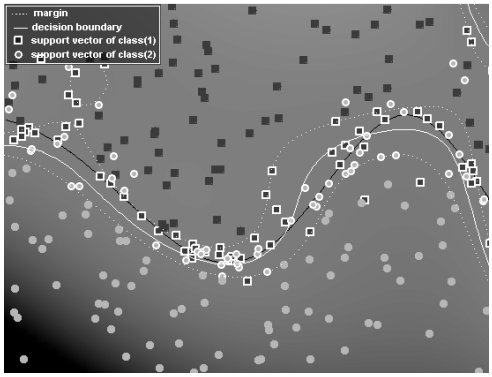
1.a. Continuous XOR: **all**



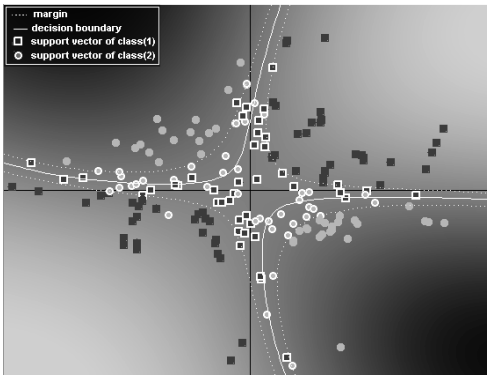
1.b. Sine Function: **all**



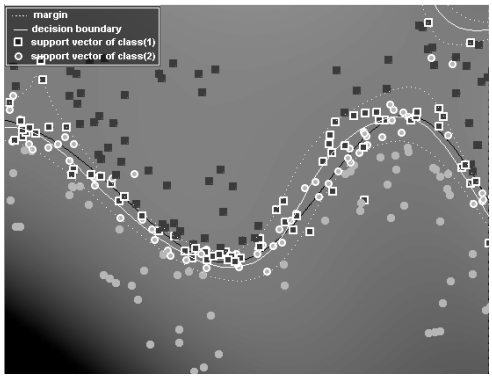
2.a. Continuous XOR: **ran**



2.b. Sine Function: **ran**



3.a. Continuous XOR: **sel**



3.b. Sine Function: **sel**

Figure 11: Patterns and SVM decision boundaries: decision boundary is depicted as a solid line and the margins as the dotted lines. Support vectors are outlined. The decision boundary of **all** looks more alike to that of **sel** than to that of **ran**. This explains why **sel** performed similarly to **all**.

in large-sized problems. Also note that the pairwise test show that SEL shows more similarity to ALL than any other method.

5.3 Real-world Marketing Dataset

The proposed algorithm was also applied to a marketing dataset from the Direct Marketing Educational Foundation (DMEF: The Direct Marketing Association). The dataset DMEF4 has been used in various researches (Ha et al., 2005; Malthouse, 2001, 2002). It is concerned with an up-scale gift business that mails general or specialized catalogs to customers. The task predicts a customer will response to the offering during the test period, 09/1992–12/1992. If the customer who has received the catalog buys the product, then he/she is labelled '+1'; otherwise, he/she is labelled '-1'. The training set is given based on the period, 12/1971–06/1992. There are 101,532 patterns in the dataset, each of which represents the purchase history information of a customer. We derived 17 input variables out of 91 original ones just as in (Ha et al., 2005; Malthouse, 2001).

To show the effectiveness of SEL, we compared it with seven RANs because random sampling has most commonly been employed when researchers in this field attempt to reduce the size of the training set. Table 5 shows the models: RAN* denotes an SVM trained with random samples, where '*' indicates the ratio of random samples drawn without replacement. Each model was trained and evaluated by using five-fold cross-validation. The hyperparameters of SVM were determined from $(C, \sigma) = \{0.1, 1, 10, 100, 1000\} \times \{0.25, 0.5, 1, 2, 3\}$. The number of neighbors (k) for SEL was set to 4. The OSU SVM Classifier was used as an SVM solver (Kernel Machines Organization). Typically, a DMEF dataset has a severe class imbalance problem because the customers' response rate for the retailer's offer is very low, i.e., 9.4% for DMEF4. In that case, an ordinary accuracy measure tends to mislead us about the results by giving more weight to the heavily-represented class. Thus, we used another accuracy measure, Balanced Correct-classification Rate (BCR), defined as

$$\text{Balanced Correct-classification Rate (BCR)} \quad = (\tfrac{m_{11}}{m_1}) \cdot (\tfrac{m_{22}}{m_2}),$$

where m_i denotes the size of class i and m_{ii} is the number of patterns correctly classified into class i. Now, the performance measure is balanced between

Table 4: Empirical result for bench-marking datasets

		Num. of Training Patterns	Num. of Support Vectors	Preprocessing Time (Ratio)	Training Time (Ratio)	Test Error Rate (%)	McNemar's p-value
Continuous XOR: 1000 test patterns							
ALL	$C = 10, \sigma = 0.5$	600	167	–	1.00	**9.67**	–
RAN	$C = 100, \sigma = 1$	180	68	–	0.10	12.33	0.08
SEL	$C = 100, \sigma = 0.25, k = 5$	180	84	0.20	0.12	**9.67**	**1.00**
SVM-KM	$C = 1, \sigma = 0.25, k = 60$	308	149	0.78	0.16	10.00	0.69
Sine Function: 1000 test patterns							
ALL	$C = 10, \sigma = 0.50$	500	250	–	1.00	**13.33**	–
RAN	$C = 0.1, \sigma = 0.25$	264	129	–	0.31	16.34	0.12
SEL	$C = 10, \sigma = 0.5, k = 5$	264	136	0.26	0.29	**12.67**	**0.89**
SVM-KM	$C = 50, \sigma = 0.25, k = 50$	335	237	2.04	0.27	14.00	0.81
4x4 Checkerboard: 10000 test patterns							
ALL	$C = 20, \sigma = 0.25$	1000	172	–	1.00	**4.03**	–
RAN	$C = 50, \sigma = 0.5$	275	75	–	0.42	8.44	0.00
SEL	$C = 50, \sigma = 0.25, k = 4$	275	148	0.08	0.40	**4.66**	**0.76**
SVM-KM	$C = 20, \sigma = 0.25, k = 100$	492	159	49.13	0.33	5.20	0.00
Pima Indian Diabetes: 153 test patterns (5-cv out of 768 patterns)							
ALL	$C = 100, p = 2$	615	330	–	1.00	**29.90**	–
RAN	$C = 1, p = 1$	311	175	–	0.47	31.11	0.61
SEL	$C = 100, p = 2, k = 4$	311	216	0.13	0.58	30.30	**0.87**
SVM-KM	$C = 100, p = 2, k = 62$	561	352	0.48	0.98	**28.75**	0.72
Wisconsin Breast Cancer: 136 test patterns (5-cv out of 683 patterns)							
ALL	$C = 5, p = 3$	546	87	–	1.00	**6.80**	–
RAN	$C = 1, p = 1$	96	27	–	0.05	12.33	0.60
SEL	$C = 10, p = 3, k = 6$	96	41	0.01	0.06	**6.70**	**0.94**
SVM-KM	$C = 10, p = 3, k = 55$	418	85	17.08	0.62	11.02	0.53
MNIST: 3–8: 1984 test patterns							
ALL	$C = 10, p = 5$	11982	1253	–	1.00	**0.50**	–
RAN	$C = 10, p = 5$	4089	1024	–	0.15	0.86	0.19
SEL	$C = 50, p = 5, k = 50$	4089	1024	0.21	0.10	**0.45**	**0.42**
SVM-KM	$k = 1198$	N/A	N/A	N/A	N/A	N/A	N/A
MNIST: 6–8: 1932 test patterns							
ALL	$C = 10, p = 5$	11769	594	–	1.00	**0.25**	–
RAN	$C = 10, p = 5$	1135	185	–	0.03	0.83	0.01
SEL	$C = 20, p = 5, k = 50$	1135	421	0.20	0.07	**0.25**	**0.67**
SVM-KM	$k = 1177$	N/A	N/A	N/A	N/A	N/A	N/A
MNIST: 9–8: 1983 test patterns							
ALL	$C = 10, p = 5$	11800	823	–	1.00	**0.41**	–
RAN	$C = 10, p = 5$	1997	323	–	0.07	0.99	0.01
SEL	$C = 10, p = 5, k = 40$	1997	631	0.19	0.09	**0.43**	**0.57**
SVM-KM	$k = 1180$	N/A	N/A	N/A	N/A	N/A	N/A

Table 5: SVM models: RAN* denotes an SVM trained with random samples, where '*' indicates the '%' of random sampling. And note that RAN100 corresponds to ALL . The number of patterns of SEL slightly varies with the given set of each fold of 5-CV, and thus, it is represented as an average.

Model	RAN05	RAN10	RAN20	RAN40	RAN60	RAN80	RAN100	SEL
(C, σ)	$(100, 1)$	$(100, 1)$	$(100, 0.5)$	$(100, 0.5)$	$(10, 1)$	$(10, 0.5)$	$(10, 0.5)$	$(10, 0.5)$
Number of patterns	4060 (5%)	8121 (10%)	16244 (20%)	32490 (40%)	48734 (60%)	64980 (80%)	81226 (100%)	8871 avg.

two different class-sizes.

Fig. 12 shows the BCRs of the eight SVM models and Table 6 presents the details. More patterns result in higher BCR among RAN*'s. However, the training time also increases proportionally to the number of training patterns, with the peak of 4,820 (sec) for RAN100. On the other hand, SEL takes only 68 (sec), and 129 (sec) if the NPPS running time is included. Note that one should perform SVM training several times to find a set of optimal hyper-parameters, but only once for NPPS ahead of the whole procedure of training. In the last column of the table, the p-value of McNemar's test is listed when SEL is compared with an individual RAN*. There is a statistically significant difference between SEL and RAN up to RAN60 in accuracy, but no difference between SEL and RAN80 or RAN100. Overall, SEL achieves almost the same accuracy as RAN80 or RAN100 only with the amount of training patterns comparable to RAN10 or RAN20.

6 Conclusions and Discussions

In this paper, we introduced an informative sampling method for SVM classification task. By pre-selecting the patterns near the decision boundary, one can relieve the computational burden during SVM training. In the experiments, we compared the performance of the selected pattern set (SEL) by NPPS with that of a random sample set (RAN) and that the original training set (ALL). And also we compared the proposed method with a competing

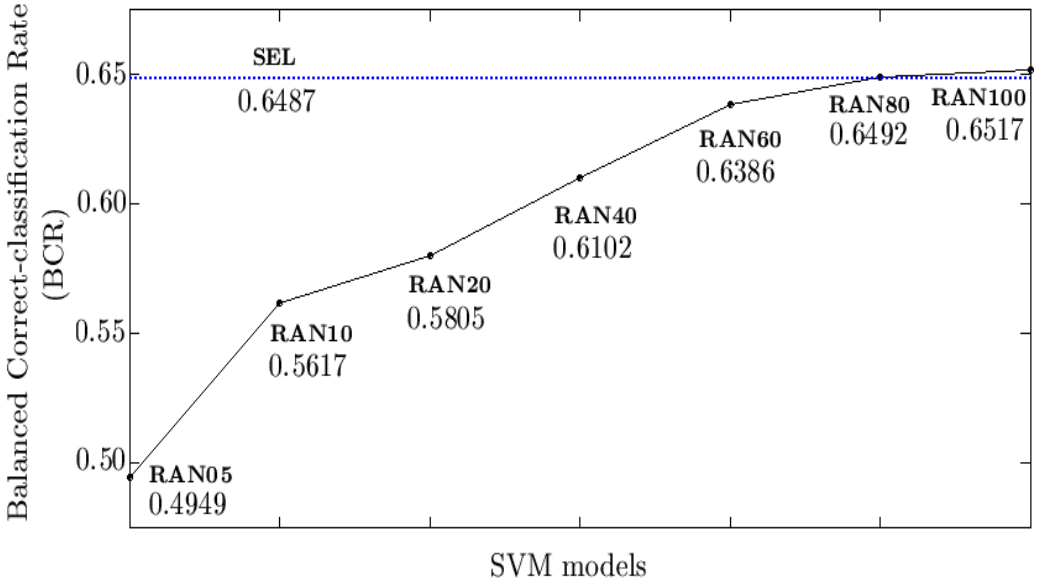


Figure 12: Balanced Correct-classification Rate (BCR): **RAN*** is depicted as a solid circle while **SEL** is represented as a dotted reference line.

Table 6: Empirical result for a real-world marketing dataset–DMEF4

	Num. of Training Patterns	Num. of Support Vectors	Training Time (sec)	Correct Rate BCR (%)	McNemar's Test (p-value)
RAN05	4,060	1,975	13.72	49.49	0.00
RAN10	8,121	4,194	56.67	56.17	0.00
RAN20	16,244	7,463	149.42	58.05	0.00
RAN40	32,490	14,967	652.11	61.02	0.04
RAN60	48,734	22,193	1,622.06	63.86	0.08
RAN80	64,980	28,968	2,906.97	64.92	0.64
RAN100	81,226	35,529	4,820.06	65.17	0.87
SEL	8,871	6,624	68.29	64.92	–

method, SVM-KM (Almeida et al., 2000). Through the comparison on synthetic and real-world problems, we empirically validated the efficiency of the proposed algorithm. SEL achieved an accuracy similar to that of ALL, while the computational cost was still similar to that of RAN with 10% or 20% of the samples. When compared with SVM-KM, SEL achieved better computational efficiency.

Here, we would like to address some future works. *First*, SVM solves a multi-class problem by divide-and-combine strategy, which divides the multi-class problem into several binary sub-problems (e.g., one-versus-others or one-versus-one), and then, combines the outputs. This has led to the application of NPPS to binary class problems. However, NPPS can readily be extended to multi-class problems without major correction. *Second*, NPPS can also be utilized to reduce the lengthy training time of neural network classifiers. But it is necessary to add extra correct patterns to the selected pattern set to enhance the overlap region near the decision boundary (Choi & Rockett, 2002; Hara & Nakayama, 2000). The rationale is that "overlap patterns" located on the "wrong" side of the decision boundary would lengthen the MLP training time. The derivatives of the back-propagated errors will be very small when evaluated at those patterns since they are grouped in a narrow region on either side of the decision boundary. By means of adding extra correct patterns, however, the network training will converge faster. In the meantime, the idea of adding some randomly selected patterns to SEL may also relieve the higher sensitivity of SEL to hyper-parameter variation (see Fig.9). If the high sensitivity is caused by "narrow" distribution of the selected patterns, it can be relaxed by some random patterns from "overall" input space. But the mixing ratio of the patterns from SEL and from RAN requires further study. *Third*, the current version of NPPS works for classification problems only, and thus, is not applicable to regression problems. A straightforward idea for regression would be to use the mean (μ) and variance (Σ) of k nearest neighbors' outputs. A pattern having a small value of Σ can be replaced by μ of its neighbors including itself. That is, $k+1$ patterns can be replaced by one pattern. On the contrary, a pattern having a large value of Σ can be totally eliminated since in a regression problem, the patterns located away from major group, such as outliers, are less important to learning. But its neighbors should be used to explore the next pattern. A similar research based on ensemble neural network was conducted by Shin & Cho (2001). *Fourth*, one of interesting future directions is the effort to

extend NPPS for data with *concept drift* (Aha et al., 1991; Bartlett et al., 2000; Helmbold & Long, 1994; Klinkenberg, 2004; Pratt & Tschapek, 2003; Stanley, 2003; Widmer & Kubat, 1996). In concept drift, the decision boundary changes as data arrive in the form of a stream. A naive approach would be to employ a moving window over the data stream and then run NPPS repeatedly. For better results, NPPS should be substantially modified and rigorously tested.

Acknowledgements

HS was partially supported by the grant for Post Brain Korea 21. SC was supported by grant No. R01-2005-000-103900-0 from Basic Research Program of the Korea Science and Engineering Foundation, Brain Korea 21, and Engineering Research Institute of SNU.

Appendix

A Empirical complexity analysis

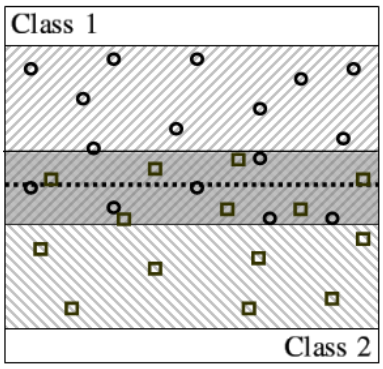
We empirically show that NPPS runs in approximately vM where M is the number of training patterns and v is the number of overlap patterns (a full theoretical analysis is available in Shin & Cho (2003a)). A total of M patterns, half from each class, were randomly generated from a pair of two-dimensional uniform distributions:

$$C_1 = \left\{\vec{x} \mid U\left(\begin{bmatrix} -1 \\ (\ 0 - \frac{1}{2}\frac{v}{M}) \end{bmatrix} < \vec{x} < \begin{bmatrix} 1 \\ (1 - \frac{1}{2}\frac{v}{M}) \end{bmatrix}\right)\right\},$$
$$C_2 = \left\{\vec{x} \mid U\left(\begin{bmatrix} -1 \\ (-1 + \frac{1}{2}\frac{v}{M}) \end{bmatrix} < \vec{x} < \begin{bmatrix} 1 \\ (0 + \frac{1}{2}\frac{v}{M}) \end{bmatrix}\right)\right\}.$$

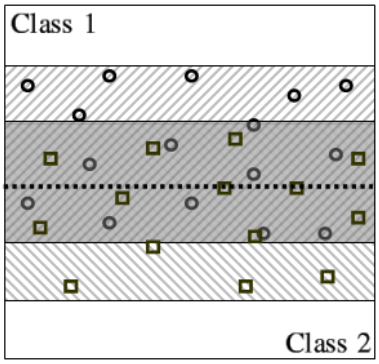
We set v to every decile of M, i.e. v= 0, $0.1M$, $0.2M$, $\cdots$, $0.9M$, M. Fig. 13 shows the distributions of $v = 0.3M$, $v = 0.5M$, and $v = 0.7M$. The larger values of v correspond to more overlap patterns. We set out to see how the computation time of NPPS works with the changing value of v– in particular whether the computation time is linearly proportional to v as aforementioned. Fig. 14.a shows the number of selected patterns through evaluation when v grows from 0 up to $M = 10,000$. In Fig. 14.b, one can clearly see that the computation time is proportional to v.

B Experiments on the number of neighbors

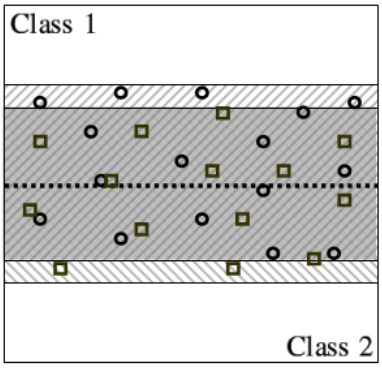
Based on the procedure presented in Fig. 7, we briefly present experiments on Pima Indian Diabetes and Wisconsin Breast Cancer. According to the procedure, the lower bounds of v were estimated as 393 and 108 from the training error rates, $P_{error} = 32.0\%$ and $P_{error} = 9.9\%$, respectively. These led to $k^* = 4$ in Pima Indian Diabetes and $k^* = 6$ in Wisconsin Breast Cancer. (The values of $k^{\min}$ were also 4 and 6, respectively.) Fig.15 shows that the test error rate stabilizes at about 30.0% for Pima Indian Diabetes when $k \geq 4$, and at about 6.7% for Wisconsin Breast Cancer when $k \geq 6$.



a. $v = 0.3M$

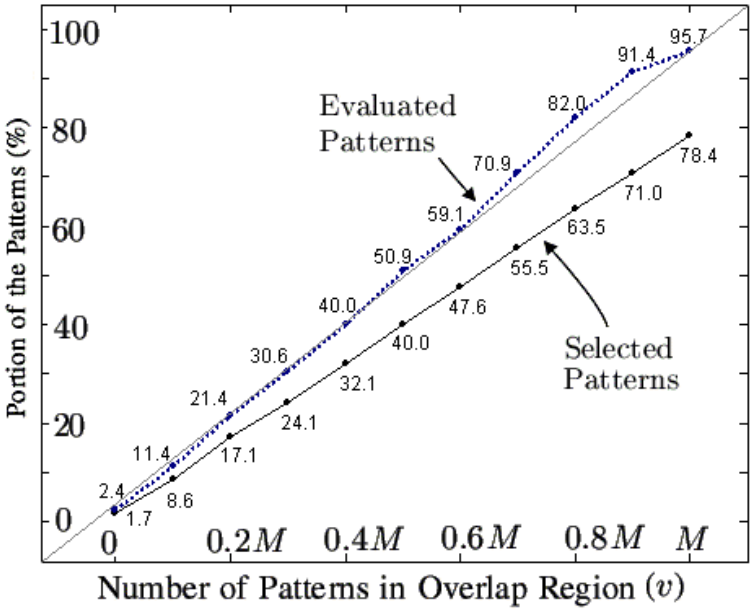


b. $v = 0.5M$

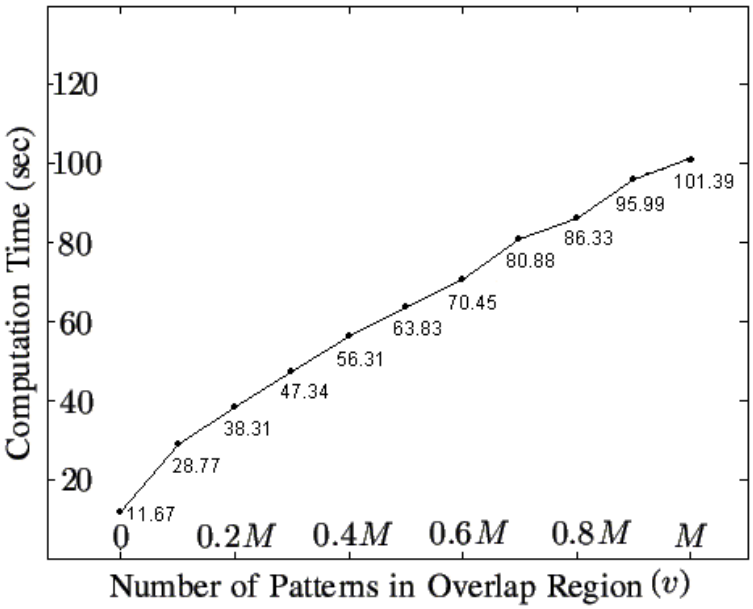


c. $v = 0.7M$

Figure 13: Two uniform distributions' overlap: the dark gray area is the overlap region which contains v patterns. The number of overlap patterns, v, is set to every decile of training set size, M. (a), (b), and (c) depict when $v = 0.3M$, $v = 0.5M$, and $v = 0.7M$, respectively.

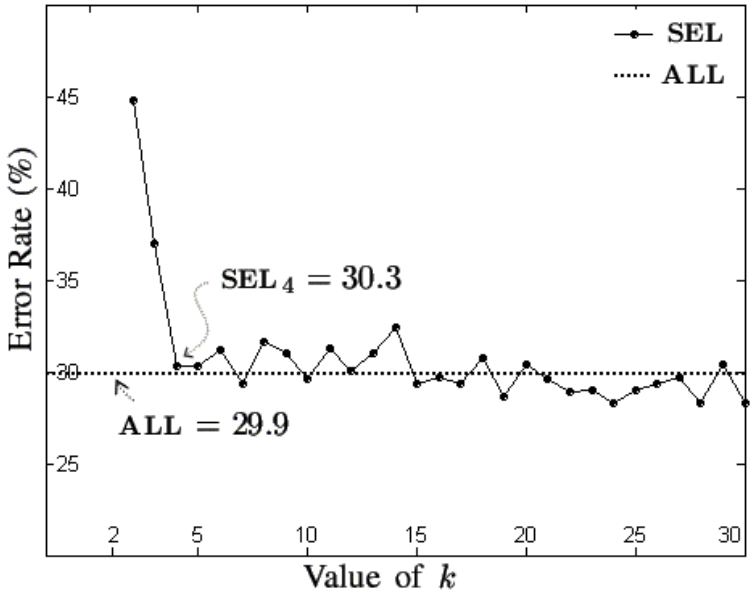


a. The number of selected patterns (%)

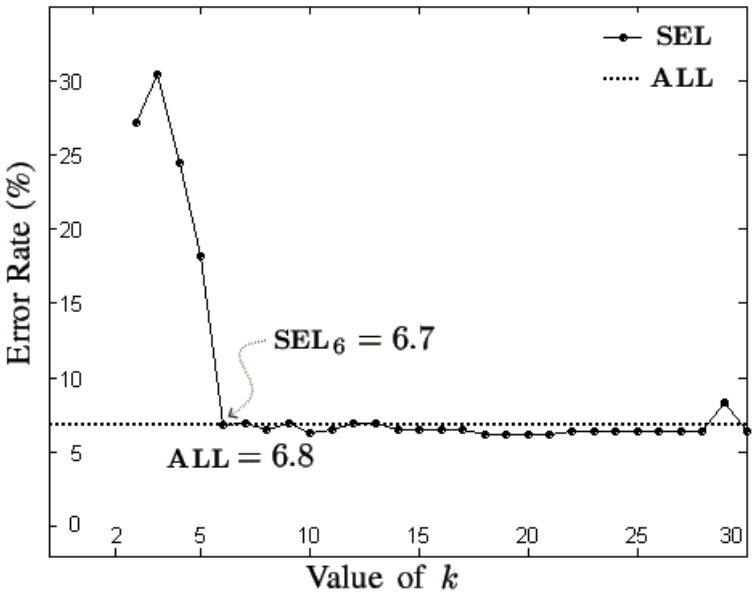


b. Computation time

Figure 14: Empirical complexity analysis: NPPS with increasing number of overlap patterns v.



a. Pima Indians Diabetes ($k^* = 4$)



b. Wisconsin Breast Cancer ($k^* = 6$)

Figure 15: SVM test error rates: the error rate is stabilized at about 30.0% for Pima Indian Diabetes when $k \geq 4$, and at about 6.7% for Wisconsin Breast Cancer when $k \geq 6$.

C Empirical comparison on SVM solvers

We provide a short comparison between OSU-SVM (OSU SVM Classifier, Kernel Machines Organization) and RSVM (Lee & Mangasarian, 2001), which are known as the fastest SVM learning algorithms. All the experimental setting was identical as in Section 5.2. The parameter of RSVM, the random sampling ratio, was set to 5–10% of training patterns according to Lee & Mangasarian (2001). In Table 7, the computational time of RSVM is shown as a ratio to the SVM training time of OSU-SVM. The results show that both of the algorithms are almost similar in accuracy. Also in training time, it is hard to tell which is superior to the other. However, RSVM was not available for the MNIST problems: it failed to load the the kernel matrix on the memory (note that RSVM is not an iterative solver). Therefore, we chose to use OSU-SVM as a base learner in our experiment because it is relatively more scalable and also parameter-free.

Table 7: Empirical comparison on different SVM solvers

		Num. of Training Patterns	Time Ratio: Training	Test Error Rate (%)	McNemar's p-value
Continuous XOR: 1000 test patterns					
OSU-SVM	$C = 10, \sigma = 0.5$	600	1.00	9.67	–
RSVM	$C = 10, \sigma = 0.5$	600	1.20	9.97	0.79
Sine Function: 1000 test patterns					
OSU-SVM	$C = 10, \sigma = 0.50$	500	1.00	13.33	–
RSVM	$C = 100, \sigma = 0.25$	500	0.31	13.46	0.85
4x4 Checkerboard: 10000 test patterns					
OSU-SVM	$C = 20, \sigma = 0.25$	1000	1.00	4.03	–
RSVM	$C = 100, \sigma = 0.25$	1000	1.87	3.88	0.00
Pima Indian Diabetes: 153 test patterns (5-cv out of 768 patterns)					
OSU-SVM	$C = 100, p = 2$	615	1.00	29.90	–
RSVM	$C = 10, p = 1$	615	0.10	29.64	0.72
Wisconsin Breast Cancer: 136 test patterns (5-cv out of 683 patterns)					
OSU-SVM	$C = 5, p = 3$	546	1.00	6.80	–
RSVM	$C = 10, p = 3$	546	1.65	8.02	0.45
MNIST: 3–8: 1984 test patterns					
OSU-SVM	$C = 10, p = 5$	11982	1.00	0.50	–
RSVM		11982	N/A	N/A	N/A
MNIST: 6–8: 1932 test patterns					
OSU-SVM	$C = 10, p = 5$	11769	1.00	0.25	–
RSVM		11769	N/A	N/A	N/A
MNIST: 9–8: 1983 test patterns					
OSU-SVM	$C = 10, p = 5$	11800	1.00	0.41	–
RSVM		11800	N/A	N/A	N/A

References

Aha, D., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1), 37–66.

Almeida, M. B., Braga, A., & Braga, J. P. (2000). SVM-KM: Speeding svms learning with a priori cluster selection and k-means. *Proc. of the 6th Brazilian Symposium on Neural Networks*, 162–167.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., & Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6), 891–923.

Bartlett, P., Ben-David, S., & Kulkarni, S. (2000). Learning changing concepts by exploiting the structure of change. *Machine Learning*, 41, 153–174.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of ACM*, 18, 509–517.

Berchtold, S., Böhm, C., Keim, D. A., & Kriegel, H.-P. (1997). A cost model for nearest neighbor search in high-dimensional data space. *Proc. of Annual ACM Symposium on Principles Database Systems*, 78–86.

Borodin, A., Ostrovsky, R., & Rabani, Y. (1999). Lower bound for high dimensional nearest neighbor search and related problems. *Proc. of the 31st ACM Symposium on Theory of Computing*, 312–321.

Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. *Proc. of the 20th International Conference on Machine Learning (ICML)*, 59–66.

Byun, H. & Lee, S. (2002). Applications of support vector machines for pattern recognition: A survey. *Lecture Notes in Computer Science (LNCS 2388)*, 213–236.

Campbell, C., Cristianini, N., & Smola, A. J. (2000). Query learning with large margin classifiers. *Proc. of the 17th International Conference on Machine Learning (ICML)*, 111–118.

Cauwenberghs, G. & Poggio, T. (2001). Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems (NIPS)*, 13, 409–415.

Choi, S. H. & Rockett, P. (2002). The training of neural classifiers with condensed dataset. *IEEE Transactions on Systems, Man, and Cybernetics-PART B: Cybernetics*, 32(2), 202–207.

Cristianini, N. & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and other kernel-based Learning Methods*. Cambridge University Press.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10, 1895–1923.

DMEF: The Direct Marketing Association. `http://www.the-dma.org/dmef/dmefdset.shtml`.

Dumais, S. (1998). Using SVMs for text categorization. *IEEE Intelligent Systems*, 21–23.

Ferri, F. J., Albert, J. V., & Vidal, E. (1999). Considerations about sample-size sensitivity of a family of edited nearest-neighbor rules. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 29(4), 667–672.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3), 209–226.

Gates, G. W. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, IT:18(3), 431–433.

Grother, P. J., Candela, G. T., & Blue, J. L. (1997). Fast implementations of nearest neighbor classifiers. *Pattern Recognition*, 30(3), 459–465.

Guttman. A. (1984). R-trees: A dynamic index structure for spatial searching. *Proc. of the ACM SIGMOD International Conference on Management of Data*, 47–57.

Ha, K., Cho, S., & MacLachlan, D. (2005). Response models based on bagging neural networks. *Journal of Interactive Marketing*, 17–30, 2005.

Hara, K. & Nakayama, K. (2000). A training method with small computation for classification. *Proc. of the IEEE-INNS-ENNS International Joint Conference*, 3, 543–548.

Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, IT:14(3), 515–516.

Hearst, M. A., Schölkopf, B., Dumais, S., Osuna, E., & Platt, J. (1997). Trends and controversies - support vector machines. *IEEE Intelligent Systems*, 13, 18–28.

Heisele, B., Poggio, T., & Pontil, M. (2000). Face detection in still gray images. Technical Report 1687, MIT AI Lab.

Helmbold, D. & Long, P. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1), 27–46.

Hush, D., Kelly, P., Scovel, C., & Steinwart, I. (2006). QP algorithms with guaranteed accuracy and run time for support vector machines. *Journal of Machine Learning Research*, 7, 733–769.

Indyk, P. (1998). On approximate nearest neighbors in non-euclidean spaces. *Proc. of IEEE Symposium on Foundations of Computer Science*, 148–155.

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms. Series: The International Series in Engineering and Computer Science, 668*. Kluwer Academic Publisher.

Johnson, M. H., Ladner, R. E., & Riskin, E. A. (2000). Fast nearest neighbor search of entropy-constrained vector quantization. *IEEE Transactions on Image Processing*, 9(9), 1435–1437.

Kleingberg, J. M. (1997). Two algorithms for nearest-neighbor search in high dimensions. *Proc. of 29th ACM symposium on Theory of Computing*, 599–608.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3), 281–300.

Kernel Machines Organization. `http://www.kernel-machines.org/`.

Koggalage, R. & Halgamuge, S. (2004). Reducing the number of training samples for fast support vector machine classification. *Neural Information Processing–Letters and Reviews*, 2(3), 57–65.

Laskov, P. (2002). Feasible direction decomposition algorithms for training support vector machines. *Machine Learning*, 46, 315–349.

Lee, Y. & Mangasarian, O. L. (2001). RSVM: Reduced support vector machines. *Proc. of the SIAM International Conference on Data Mining* (in CD), `http://www.cs.wisc.edu/~olvi/olvi.html`.

Liu, C. L. & Nakagawa, M. (2001). Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. *Pattern Recognition*, 34, 601–615.

Lyhyaoui, A., Martinez, M., Mora, I., Vazquez, M., Sancho, J., & Figueiras-Vaidal, A. R. (1999). Sample selection via clustering to construct support vector-like classifiers. *IEEE Transactions on Neural Networks*, 10(6), 1474–1481.

Malthouse, E. C. (2001). Assessing the performance of direct marketing models. *Journal of Interactive Marketing*, 15(1), 49–62.

Malthouse, E. C. (2002). Performance–based variable selection for scoring models. *Journal of Interactive Marketing*, 16(4), 10–23.

Masuyama, N., Kudo, M., Toyama, J., & Shimbo, M. (1999). Termination conditions for a fast k-nearest neighbor method. *Proc. of 3rd International Conference on Knoweldge-Based Intelligent Information Engineering Systems*, 443–446.

MNIST database. `http://yann.lecun.com/exdb/mnist/`.

Moghaddam, B. & Yang, M. H. (2000). Gender classification with support vector machines. *Proc. of International Conference on Pattern Recognition*, 306–311.

Ong, C. S., Smola, A. J., & Williamson, R. C. (2005). Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6, 1045–1071.

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: an application to face detection. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 130–136.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C. J. C., & Smola, A. J., editors, *Advances in Kernel Methods: Support Vector Machines*, 185–208. MIT Press, Cambridge, MA.

Pontil, M. & Verri, A. (1998). Properties of support vector machines. *Neural Computation*, 10, 955–974.

Pratt, K. B. & Tschapek. G. (2003). Visualizing concept drift. *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 735–740.

Schohn, G. & Cohn, D. (2000). Less is more: Active learning with support vector machines. *Proc. of the 17th International Conference on Machine Learning (ICML)*, 839–846.

Schölkopf, B., Burges, C. J. C., & Vapnik, V. (1995). Extracting support data for a given task. *Proc. of 1st International Conference on Knowledge Discovery and Data Mining*, 252–257.

Schölkopf, B., Burges, C. J. C., & Smola, A. J. (1999). *Advances in Kernel Methods: Support Vector Machines*. MIT press, Cambridge, MA.

Schölkopf, B. & Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, Cambridge, MA.

Shin, H. J. & Cho. S. (2001). Pattern selection using the bias and variance of ensemble. *Journal of the Korean Institute of Industrial Engineers*, 28(1), 112–127.

Shin, H. J. & Cho, S. (2003a). Fast pattern selection algorithm for support vector classifiers: *Time Complexity Analysis*. *Lecture Notes in Computer Science (LNCS 2690)*, 1008–1015. `http://www.kyb.mpg.de/publication.html?user=shin`.

Shin, H. J. & Cho, S. (2003b). How many neighbors to consider in pattern pre-selection for support vector classifiers? *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, 565–570.

Short, R. D. & Fukunaga, K. (1981). The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information and Theory*, IT:27(5), 622–627.

Sohn, S. & Dagli, C. H. (2001). Advantages of using fuzzy class memberships in self-organizing map and support vector machines. *Proc. of International Joint Conference on Neural Networks (IJCNN)*, (in CD).

Sonnenburg, S., Rätsch, G., Schäfer, S., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7, 1531–1565.

Stanley, K. O. (2003). Learning concept drift with a committee of decision trees. Technical Report UTAI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA.

Tsaparas, P. (1999). Nearest neighbor search in multidimensional spaces. Technical Report Technical Report 319-02, Department of Computer Science, University of Toronto.

UCI repository. `http://www.ics.uci.edu/~mlearn/`.

Vapnik, V. (1999). *The Nature of Statistical Learning Theory*. Springer Verlag, 2 edition.

Widmer, G. & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1), 69–101.

Zheng, S.F., Lu, X.F., Zheng, N.N., & Xu, W.P. (2003). Unsupervised clustering based reduced support vector machines. *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2, 821–824.