Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition Letters 26 (2005) 707–718

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Invariance of neighborhood relation under input space to feature space mapping

Hyunjung Shin *, Sungzoon Cho

Department of Industrial Engineering, Seoul National University, San 56-1, Shillim-Dong, Kwanak-Gu, Seoul 151-744, Republic of Korea

Received 23 February 2004; received in revised form 10 June 2004
Available online 11 November 2004

Abstract

If the training pattern set is large, it takes a large memory and a long time to train support vector machine (SVM). Recently, we proposed neighborhood property based pattern selection algorithm (NPPS) which selects only the patterns that are likely to be near the decision boundary ahead of SVM training [Proc. of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Lecture Notes in Artificial Intelligence (LNAI 2637), Seoul, Korea, pp. 376–387]. NPPS tries to identify those patterns that are likely to become support vectors in feature space. Preliminary reports show its effectiveness: SVM training time was reduced by two orders of magnitude with almost no loss in accuracy for various datasets. It has to be noted, however, that decision boundary of SVM and support vectors are all defined in feature space while NPPS described above operates in input space. If neighborhood relation in input space is not preserved in feature space, NPPS may not always be effective. In this paper, we show that the neighborhood relation is invariant under input to feature space mapping. The result assures that the patterns selected by NPPS in input space are likely to be located near decision boundary in feature space.

Keywords: Support vector machine (SVM); Pattern selection; Neighborhood relation; Input to feature space mapping

1. Introduction

In support vector machine (SVM) quadratic programming (QP) formulation, the dimension of kernel matrix ($M \times M$) is equal to the number of training patterns (M). A standard QP solver has time complexity of order $O(M^3)$: MINOS, CPLEX, LOQO, and MATLAB QP routines. In order to attack the large scale SVM QP problem,

* Corresponding author. Present address: Department of Empirical Inference (Department of Dr. Schölkopf), Max-Planck-Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany. Tel.: +82 2 8834913; fax: +82 2 8898560.

E-mail addresses: shin@tuebingen.mpg.de, hjshin72@snu.ac.kr (H. Shin), zoon@snu.ac.kr (S. Cho).

0167-8655/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2004.09.023

the decomposition methods or iterative methods have been suggested which break down the large QP problem into a series of smaller QP problems: Chunking, SMO, SVM$^{\text{light}}$, and SOR (Hearst et al., 1997; Platt, 1999). The general time complexity of those methods is approximately $T \cdot O(Mq + q^3)$ where T is the number of iterations and q is the size of the working set. Needless to say, T increases as M increases.

One way to circumvent this computational burden is to select some of training patterns in advance which contain most information given to learning. One of the merits of SVM theory distinguishable from other learning algorithms is that it is clear that which patterns are of importance to training. Those are called support vectors (SVs), distributed near the decision boundary, and fully and succinctly define the classification task at hand (Cauwenberghs and Poggio, 2001; Pontil and Verri, 1998; Vapnik, 1999). Furthermore, on the same training set, the SVMs trained with different kernel functions, i.e., RBF, polynomial, and sigmoid, have selected almost identical subset as support vectors (Schölkopf et al., 1995). Therefore, it is worth finding such *would-be* support vectors prior to SVM training (see Fig. 1).

To date, there have been several approaches on pattern selection for SVM. Lyhyaoui et al. (1999) implemented RBF classifiers which somewhat resemble SVMs, to make clear the difference between both methods. RBF classifiers were built based on the patterns near the decision boundary. To find them, they proposed to search 1-nearest neighbor in opposite class after class-wise clustering. But this method presumes that the training set should be clean. Almeida et al. (2000) conducted k-means clustering first on the entire training set regardless of patterns' class-membership. Those clusters which contain patterns from one class are called *homogeneous*, while those which do not are called *heterogeneous*. 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 of this research is that it is not clear how to determine the number of clusters. Koggalage and Halgamuge (2004) also employed clustering to select the patterns from the training set. It is quite similar to Almeida et al. (2000) approach in that they conducted clustering on the entire training set first and chose the patterns which belong to the clusters having heterogeneous members. For a homogeneous cluster, on the contrary, the patterns along the rim of cluster were selected not the centroid. It is relatively a safer approach since even for homogeneous clusters there can exist the patterns near the decision boundary 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, it is still vague how to set the radius and how to define the width of the rim from it. Zheng et al. (2003) proposed to substitute clus-

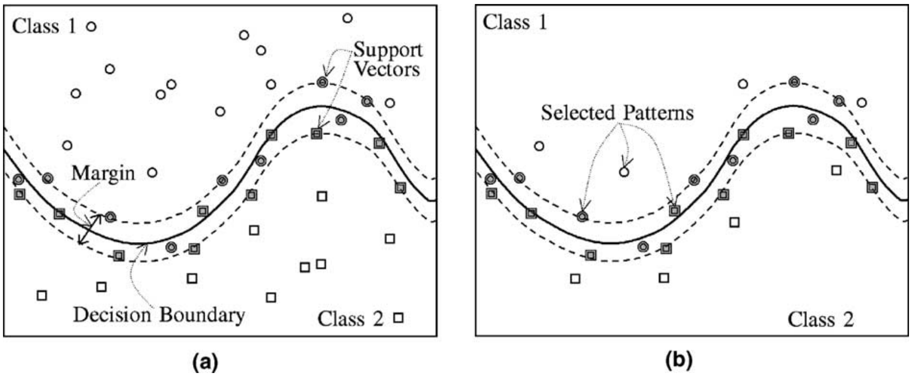

Fig. 1. Pattern selection: a large training (original training) set shown in (a) is condensed to a small training (selected training) set (b) which is composed of only potential support vectors.

ters' centroids for random samples of Lee and Mangasarian (2001)'s reduced SVM (RSVM). RSVM is to choose random samples from the training set and regard them as support vectors. But all the training patterns are still used as constraints of SVM QP. In RSVM, it is not clear that how many random samples are required not to degrade the original accuracy of SVM. Zheng et al. (2003)'s idea on RSVM is based on that the centroids are more representative than random samples. In summary, clustering-based algorithms have a common weakness: the selected patterns are fully dependent on the clustering performance which could be unstable. A related performance comparison was dealt with in the research of Liu and Nakagawa (2001). A bit different approach was done by Sohn and Dagli (2001). In order to reduce the SVM training set as well as to eliminate noisy patterns (outliers), they utilized fuzzy class membership through k nearest neighbors. According to the value of fuzzy class membership, they could check a possibility of the pattern belonging to the class and eliminate the ones having a weak possibility. However, they seem to overlook the importance of the patterns near the decision boundary by treating them equal to the noisy patterns (outliers far from the decision boundary).

In this paper, we propose *neighborhood property based pattern selection algorithm* (NPPS). The practical time complexity of NPPS is $O(vM)$ where v is the number of patterns in the overlap region around 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 "an overlap or a noisy pattern tends to belong to a different class from its neighbors." And the *third* neighborhood property is that "the neighbors of a pattern located near the decision boundary tend to be located near the decision boundary as well." The first one is used for identifying those patterns located near the decision boundary. The second one is used for removing the patterns located on the wrong side of the decision boundary. And the third one is used for skipping calculation of unnecessary distances between patterns, thus accelerating the pattern selection procedure. In short, NPPS uses only local neighbor information to identify those patterns likely to be located near decision boundary. NPPS algorithm and its performance will be covered in the next section. And further details are also available in (Shin and Cho, 2003a,b).

It has to be noted, however, that decision boundary of SVM and support vectors are all defined in *feature space* while NPPS described above operates in *input space*. Since the mapping from input space to feature space is highly nonlinear and dimension expanding, distortion of neighborhood relation could occur. In other words, neighborhood relation in input space may not be preserved in feature space. If that is the case, local information in input space may not be correct in feature space, thus impairing the effectiveness of NPPS. There are two approaches to solve this problem. The *first* involves running NPPS in feature space, and the *second* involves proving that the neighborhood relation is invariant under the input to feature space mapping. Let us consider the first approach. In order to compute the distance between two patterns, one has to have the optimal kernel function and hyper-parameters, which are usually found by trial-and-error accompanying with multiple trials of SVM training with all patterns. Obviously, that is not acceptable since the purpose of pattern selection is to avoid training SVM with all patterns. On the other hand, in the second approach, NPPS can be executed only once in input space since it does not involve searching for optimal kernel and hyper-parameters. Thus, we take the second approach in this paper by showing that the neighborhood relation is invariant under the input to feature space mapping. Through the theoretical justification for the second approach, we can avoid the computational burden which could be incurred by taking the first approach.

The remaining of this paper is organized as follows. In Section 2, we briefly introduce NPPS algorithm and its performance reported earlier. In Section 3, we provide proofs on the invariance of the neighborhood relation under the input to feature space mapping through two typical kernel functions: RBF and polynomial. In Section 4, we conclude this paper with remarks on limitations of our proofs.

2. Neighborhood property based pattern selection

The proposed idea is to select only those patterns located around decision boundary since they are the ones that contain most information. Obviously, the decision boundary is not known until a classifier is built. Thus, the algorithm utilizes neighborhood properties to infer the proximity of a pattern to the decision boundary. The first neighborhood property is that "a pattern located near the decision boundary tends to have more heterogeneous neighbors in their class-membership." Thus, the proximity of pattern $\vec{x}$'s to the decision boundary is estimated by "Neighbors_ Entropy $(\vec{x}, k)$", which is defined as the entropy of the pattern $\vec{x}$'s k-nearest neighbors' class labels,

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

where P_j is defined as k_j/k where k_j is the number of neighbors belonging to class j among the k nearest neighbors of $\vec{x}$ in J class classification problem. A pattern with a positive Neighbors_Entropy $(\vec{x}, k)$ value is assumed to be close to the decision boundary, thus selected for training. Those patterns are likely to be SVs. Among the patterns having a positive value of Neighbors_Entropy $(\vec{x}, k)$, however, overlap or noisy patterns are also present. Here, let us define *overlap patterns* as the patterns that are located in the *other* side of the decision boundary since the class distributions' overlap. Genuine *noisy patterns* are also defined as the patterns that are located in the other side of the decision boundary, but far away from the decision boundary. Those are not adjacent to overlap patterns since they occur due to reasons other than class distribution overlap. Overlap region is a region in feature space occupied by the overlap patterns from either side of the decision boundary. Note that the overlap region contains not only the overlap patterns, but also the *close non-overlap* patterns which are located close to the decision boundary, yet in the *right* side of the decision boundary. Among the patterns in the overlap region, overlap or noisy patterns have to be identified and removed as much as possible since they are more likely to be the error SVs which would be misclassified. The second neighborhood property thus dictates that "an overlap or a noisy pattern 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 labeled. The measure "*Neighbors_Match* $(\vec{x}, k)$" is defined as the ratio of $\vec{x}$'s neighbors whose label matches that of $\vec{x}$,

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

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 is likely *to* be the ones incorrectly labeled. Only the patterns satisfying [Selecting Condition],

$$\boxed{\begin{aligned}&\text{Neighbors\_Entropy }(\vec{x}, k) > 0\\ &\quad\text{and}\quad \text{Neighbors\_Match }(\vec{x}, k) \geqslant \beta \cdot \frac{1}{J}.\end{aligned}}$$

are selected where $0 < \beta \leqslant 1$. The larger value of β leads to the smaller number of selected patterns. Setting β with a value of 1 means that we select the patterns correctly classified by kNN, preliminary to the original classifier, SVM. Thus, some of the critical patterns near the decision boundary might be discarded. To conserve the original SVM accuracy, we lessen the prior influence of kNN by weighing down $\beta = 0.5$. By doing so, the overlap (genuine noise) patterns far away from the decision boundary can still be eliminated while the patterns near the decision boundary can be more preserved.

However, the *naive* NPPS evaluating kNNs for M patterns has time complexity of $O(M^2)$, so the pattern selection process itself can be time-consuming. To accelerate the pattern selection procedure, let us 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 compute only the neighbors' label entropy for the patterns near the decision boundary instead of all the training patterns. A pattern is expanded or a pattern's neighbors are evaluated when its Neighbors_Entropy is positive. With an initial set of randomly selected patterns, we evaluate only the neighbors of a pattern satisfying [Expanding Condition]

$$\boxed{\text{Neighbors\_Entropy }(\vec{x},k) > 0}$$

in the next step. This successive *neighbors only* evaluation of the *current* pattern set is repeated until all the patterns near the decision boundary are chosen and evaluated.

For better understanding, Fig. 2 presents a toy example. Let us assume $J = 2$, $k = 3$, and $\beta = 1$. At the initial stage shown in Fig. 2(a), three patterns $\vec{x}^1$, $\vec{x}^2$, and $\vec{x}^3$ are randomly selected. They are marked as outlined circles or squares. Neighbor searching is represented by dotted arrows. After its neighbors' label composition is evaluated, $\vec{x}^1$ is not expanded since it does not meet the Expanding Condition. On the other hand, $\vec{x}^2$ and $\vec{x}^3$ are expanded. $\vec{x}^2$ satisfies the Selecting Condition as well as the Expanding Condition. Thus, it is added to the selected pattern set. $\vec{x}^3$, however, does satisfy the Expanding Condition only. The patterns like $\vec{x}^1$ are depicted as "gray" circles or squares, $\vec{x}^2$ as "black solid" and $\vec{x}^3$ as "double outlined" in the next stage. Now in Fig. 2(b), $\vec{x}^4$, $\vec{x}^5$, $\vec{x}^6$ and $\vec{x}^7$, $\vec{x}^8$, $\vec{x}^9$ which are the neighbors of $\vec{x}^2$ and $\vec{x}^3$, respectively, are evaluated. Among them, $\vec{x}^4$ is excluded from expanding, while the rest are all selected and expanded in Fig. 2(c). Neighborhood relationship is more than somewhat mutual, hence expanding is conducted only once per pattern to

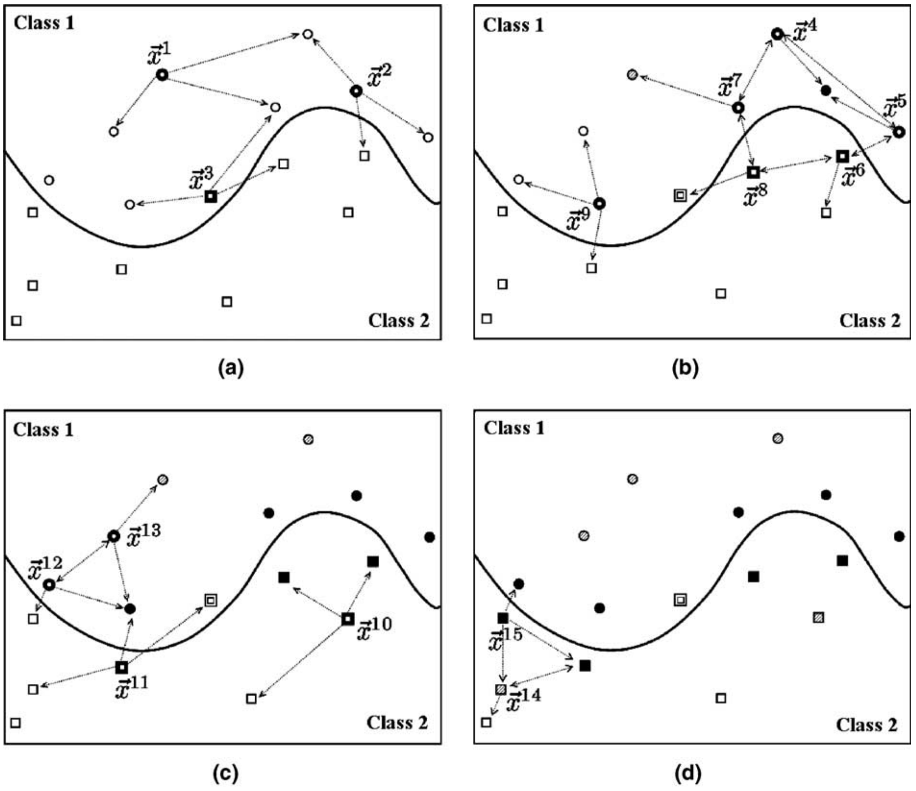

Fig. 2. A toy example for fast NPPS: the procedure starts with randomly sampled patterns from the initial stage (a), and gets at the final stage (d). "Outlined" circles or squares are the patterns to be expanded (to find its neighbors or to evaluate its neighbors). Among them the selected patterns are marked as "black solid" while expanded but not selected ones as "double outlined". Neighbor searching is represented as dotted arrows. (a) The first stage, (b) the second stage, (c) the third stage and (d) the fourth stage.

avoid redundant evaluation. Therefore, only those patterns that were not evaluated before such as $\vec{x}^{10}$, $\vec{x}^{11}$, $\vec{x}^{12}$, and $\vec{x}^{13}$ are evaluated as shown in Fig. 2(c). Similarly, $\vec{x}^{10}$ and $\vec{x}^{13}$ are not expanded, and $\vec{x}^{11}$ and $\vec{x}^{12}$ are added to the selected pattern set, and their neighbors $\vec{x}^{14}$ and $\vec{x}^{15}$ are evaluated in the last stage, Fig. 2(d).

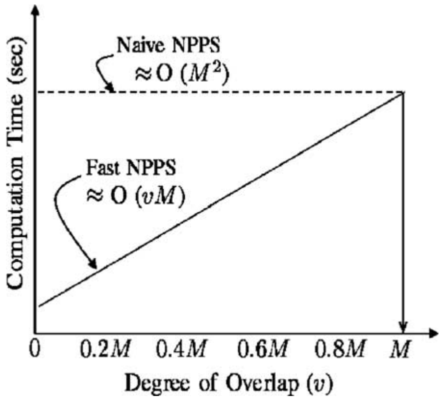

Fig. 3. Theoretical relationship between the computation time and v: the computation time of fast NPPS is linearly proportional to v when M is fixed.

This lazy evaluation of *fast* NPPS reduces the practical time complexity from $O(M^2)$ to $O(vM)$, where v is the number of patterns in the overlap region. In most practical problems, $v < M$ holds. Fig. 3 depicts the theoretical relationship between the computation time and v of the two algorithms. The computation time of naive NPPS, $O(M^2)$, does not change as long as M is fixed. Meanwhile, that of fast NPPS is linearly proportional to v. Therefore, fast NPPS is always faster than naive NPPS except in the case of $v = M$. Preference of one over the other algorithm is of concern to a tradeoff between ease of implementation and time complexity. We provided the time complexity proofs for NPPS in (Shin and Cho, 2003b). The algorithm and related notations are shown in Fig. 4 and Table 1.

Fig. 5–7 visualize the previous experimental results on artificial problems (Shin and Cho,

```
NPPS ($\mathbf{D}$, $k$) {
  [0] Initialize $\mathbf{D}_e^0$ with randomly chosen patterns from $\mathbf{D}$.
      Constants $k$ and $J$ are given. Initialize $i$ and various sets as follows:
      $i \leftarrow 0,\quad \mathbf{S}_o^0 \leftarrow \emptyset,\quad \mathbf{S}_x^0 \leftarrow \emptyset,\quad \mathbf{S}^0 \leftarrow \emptyset.$

  while $\mathbf{D}_e^i \neq \emptyset$ do
    [1] Choose $\vec{x}$ satisfying [Expanding Condition].
        $\mathbf{D}_o^i \leftarrow \{\vec{x} \mid Neighbors\_Entropy(\vec{x},k) > 0,\ \vec{x} \in \mathbf{D}_e^i\}.$
        $\mathbf{D}_x^i \leftarrow \mathbf{D}_e^i - \mathbf{D}_o^i.$

    [2] Select $\vec{x}$ satisfying [Selecting Condition].
        $\mathbf{D}_s^i \leftarrow \{\vec{x} \mid Neighbors\_Match(\vec{x},k) \geq \beta/J,\ \vec{x} \in \mathbf{D}_o^i\}.$

    [3] Update the pattern sets.
        $\mathbf{S}_o^{i+1} \leftarrow \mathbf{S}_o^i \cup \mathbf{D}_o^i$ : the expanded,
        $\mathbf{S}_x^{i+1} \leftarrow \mathbf{S}_x^i \cup \mathbf{D}_x^i$ : the non-expanded,
        $\mathbf{S}^{i+1} \leftarrow \mathbf{S}^i \cup \mathbf{D}_s^i$ : the selected.

    [4] Compute the next evaluation set $\mathbf{D}_e^{i+1}$.
        $\mathbf{D}_e^{i+1} \leftarrow \bigcup_{\vec{x} \in \mathbf{D}_o^i} k\mathbf{NN}(\vec{x}) - (\mathbf{S}_o^{i+1} \cup \mathbf{S}_x^{i+1}).$
    [5] $i \leftarrow i + 1.$
  end
  return $\mathbf{S}^i$
}
```

Fig. 4. NPPS.

Table 1
Notation

Symbol	Meaning
$\boldsymbol{D}$	The original training set whose cardinality is M
$\boldsymbol{D}_e^i$	The evaluation set at ith step
$\boldsymbol{D}_o^i$	A subset of $\boldsymbol{D}_e^i$, the set of patterns to be "expanded" from $\boldsymbol{D}_e^i$ each element of which will compute its k nearest neighbors to constitute the next evaluation set, $\boldsymbol{D}_e^{+1+1}$
$\boldsymbol{D}_x^i$	A subset of $\boldsymbol{D}_e^i$, the set of patterns "not to be expanded" from $\boldsymbol{D}_e^i$, or $D_x^i = \boldsymbol{D}_e^i - \boldsymbol{D}_o^i$
$\boldsymbol{D}_s^i$	The set of "selected" patterns from D_o^i at ith step
$\boldsymbol{S}_o^i$	The accumulated set of expanded patterns, $\bigcup_{j=0}^{i-1} \boldsymbol{D}_o^j$
$\boldsymbol{S}_x^i$	The accumulated set of non-expanded patterns, $\bigcup_{j=0}^{i-1} \boldsymbol{D}_x^j$
$\boldsymbol{S}^i$	The accumulated set of selected patterns, $\bigcup_{j=0}^{i-1} \boldsymbol{D}_s^j$ the last of which $\boldsymbol{S}^N$ is the reduced training pattern set
$k\boldsymbol{NN}(\vec{x})$	The set of k nearest neighbors of $\vec{x}$

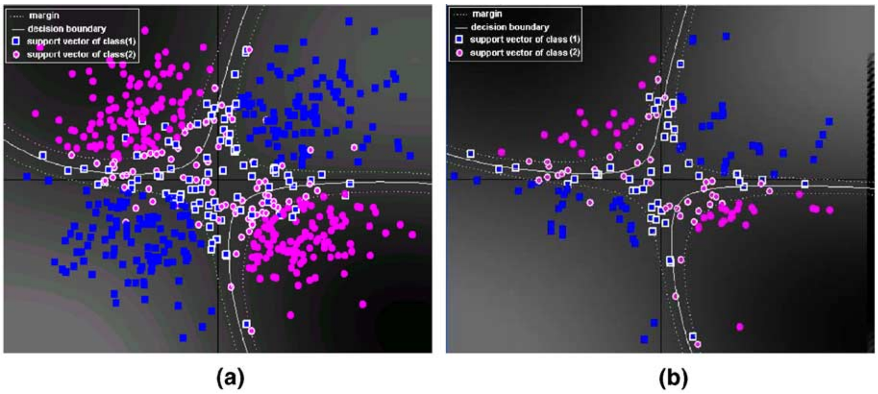

Fig. 5. Continuous XOR Problem: SVM result. (a) ALL patterns; (b) SELECTED patterns.

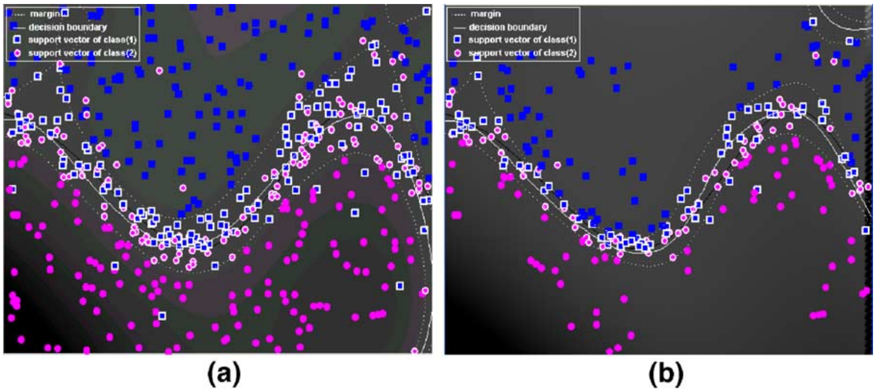

Fig. 6. Sine Function Problem: SVM result. (a) ALL patterns; (b) SELECTED patterns.

2003a): Continuous XOR (Fig. 5), Sine Function (Fig. 6), and 4 × 4 Checkerboard (Fig. 7). The decision boundary is depicted as a solid line and the *margins* are defined by the dotted lines in both sides of it. Support vectors are outlined. In each figure, sub-figure (a) indicates a typical SVM result

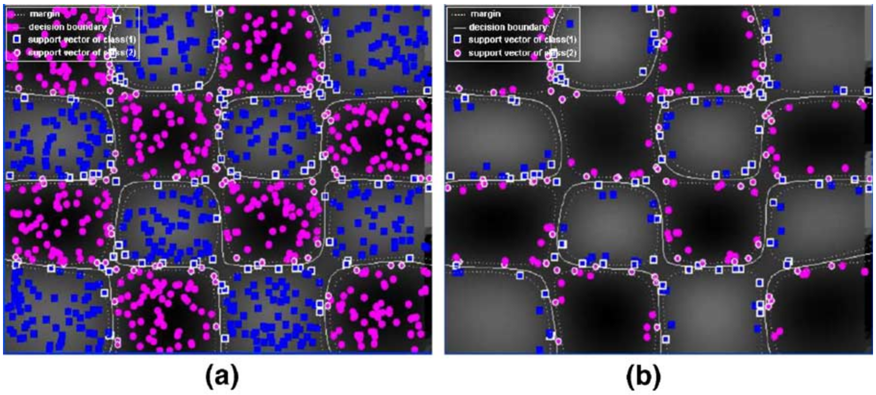

Fig. 7. 4 × 4 Checkerboard Problem: SVM result. (a) ALL patterns; (b) SELECTED patterns.

Table 2
Empirical result comparison

	No. of training patterns		No. of support vectors		Execution time (s)		Test error (%)	
	ALL	SELECTED	ALL	SELECTED	ALL	SELECTED (=SVM + NPPS)	ALL	SELECTED
Continuous XOR*	600	179	167	84	454.83	4.06 (=3.85 + 0.21)	9.67	9.67
Sine function*	500	264	250	136	267.76	8.96 (=8.79 + 0.17)	13.33	13.33
4 × 4 Checkerboard	1000	275	172	148	3.81	0.41 (=0.09 + 0.32)	4.03	4.66
Pima Indian diabetes	615	311	330	216	203.91	28.00 (=27.86 + 0.14)	29.90	30.30
Wisconsin breast cancer	546	96	87	41	2.14	0.13 (=0.03 + 0.10)	6.80	6.80
MNIST: 3–8	11 982	4089	1253	1024	477.25	147.73 (=49.84 + 97.89)	0.50	0.45
MNIST: 6–8	11 769	1135	594	421	222.84	58.96 (=14.69 + 44.27)	0.31	0.31
MNIST: 9–8	11 800	1997	823	631	308.73	86.23 (=26.61 + 59.62)	0.41	0.43
DMEF4	81 226	8871	35 529	6624	4820.06	129.29 (=68.29 + 61.00)	34.83	35.13

'*' stands for that the SVM training of corresponding problems was conducted with a standard QP solver, i.e., Gunn's SVM MATLAB Toolbox. On the contrary, because of heavy memory burden and lengthy training time caused by large training set, others were trained with an iterative SVM solver known as one of the fastest solvers, i.e., OSU SVM Classifier Toolbox (Kernel-Machines Organization).

of *all* patterns while sub-figure (b) stands for that of *selected* patterns by NPPS. The decision boundaries in both sub-figures look quite similar, thus, generalization performance is similar. Table 2 summarizes the empirical results of NPPS reported in (Shin and Cho, 2003a, in press). The table includes the results obtained from tests with three artificial datasets mentioned above, and 5 real-world bench-marking datasets (UCI Machine Learning Repository; MNIST database) as well as a marketing dataset (DMEF dataset Library). The results show that NPPS reduced SVM training time up to almost two orders of magnitude with virtually no loss of accuracy.

3. Proofs on validity of pattern selection in input space

As described in Section 2, NPPS operates in *input space* (I) using local information there. However, decision boundary of SVM and support vectors are all defined in *feature space* (Φ). Since the mapping $I \mapsto \Phi$ is highly nonlinear as well as dimension expanding, we have to ensure that neighborhood relation in input space be preserved in feature space. We now provide proofs on that the k nearest neighbors of a pattern in the input space I are also the k nearest neighbors of the pattern in the feature space Φ.

Definition 1 (*kNN Invariance*). Let $kNN_I(\vec{x})$ be the set of k nearest neighbors of a pattern $\vec{x}$ in the input space I, and $kNN_\Phi(\vec{x})$ be that of the pattern $\Phi(\vec{x})$ in the feature space Φ. If both sets are identical

$$kNN_I(\vec{x}) = kNN_\Phi(\vec{x}), \quad \forall k > 0,\ \forall \vec{x}$$

the invariance of the k nearest neighbors (NN) holds.

Finding the nearest neighbors necessarily implies distance calculation. In terms of the squared Euclidean distance which is the most commonly used distance measure, the distance among patterns in the input space I is

$$\|\vec{x} - \vec{y}\|^2 = \vec{x} \cdot \vec{x} + \vec{y} \cdot \vec{y} - 2\vec{x} \cdot \vec{y}. \tag{1}$$

The distance in the feature space Φ is similarly drawn as

$$\begin{aligned}\|\Phi(\vec{x}) - \Phi(\vec{y})\|^2 &= \Phi(\vec{x}) \cdot \Phi(\vec{x}) + \Phi(\vec{y}) \cdot \Phi(\vec{y}) \\ &\quad - 2\Phi(\vec{x}) \cdot \Phi(\vec{y})\end{aligned} \tag{2}$$

where $\Phi(\cdot)$ is a mapping function from the input space to the feature space, $\Phi(\cdot)$: $I \mapsto \Phi$. One might obtain $\Phi(\vec{x})$ directly but the formula is extremely complicated. Thanks to the fact that the mapping $\Phi(\cdot)$ always appears within a form of inner product during SVM QP calculation, one thus uses *kernel trick* which substitutes the inner product to a kernel function, $\Phi(\vec{x}) \cdot \Phi(\vec{y}) = K(\vec{x},\vec{y})$. If this kernel trick is applied to Eq. (2), then the distance in the feature space becomes

$$\|\Phi(\vec{x}) - \Phi(\vec{y})\|^2 = K(\vec{x},\vec{x}) + K(\vec{y},\vec{y}) - 2K(\vec{x},\vec{y}). \tag{3}$$

We will consider the following typical kernel functions:

$$\begin{aligned}&\text{RBF}:\ K(\vec{x},\vec{y}) = \exp\left(-\frac{\|\vec{x}-\vec{y}\|^2}{2\sigma^2}\right), \\ &\text{polynomial}:\ K(\vec{x},\vec{y}) = (\vec{x} \cdot \vec{y} + 1)^p.\end{aligned} \tag{4}$$

As long as the relative distance magnitude of the input space is preserved in the feature space Φ for all patterns, the composition of the k nearest neighbors of a pattern will be invariant. We now define *proximity invariance*.

Definition 2 (*Proximity Invariance*). For the patterns $\vec{x}$, $\vec{y}_1$ and $\vec{y}_2$ ($\vec{x} \neq \vec{y}_1$, $\vec{x} \neq \vec{y}_2$, and $\vec{y}_1 \neq \vec{y}_2$) in the input space I satisfying

$$\|\vec{x} - \vec{y}_1\|^2 < \|\vec{x} - \vec{y}_2\|^2,$$

the invariance of proximity holds if they preserve their relative distances in the feature space Φ

$$\|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 < \|\Phi(\vec{x}) - \Phi(\vec{y}_2)\|^2.$$

It is obvious that *kNN invariance* holds if *proximity invariance* holds. The following two theorems provide proofs on kNN invariance for the two kernels, RBF and polynomial, by inducing proximity invariance for each of them.

Theorem 1 (kNN Invariance for RBF Kernel). *kNN invariance holds when the mapping function $\Phi(\vec{x})$ is defined such that*

$$\Phi(\vec{x}) \cdot \Phi(\vec{y}) = K(\vec{x},\vec{y}) = \exp\left(-\frac{\|\vec{x}-\vec{y}\|^2}{2\sigma^2}\right).$$

Proof. Let $\vec{y}_1$ and $\vec{y}_2$ be two distinct neighbors of $\vec{x}$ with $\|\vec{x} - \vec{y}_1\|^2 < \|\vec{x} - \vec{y}_2\|^2$, i.e., $\vec{y}_1$ is closer to $\vec{x}$ than $\vec{y}_2$. Suppose the invariance of proximity does not hold for mapping function $\Phi(\vec{x})$, i.e., $\|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 \geqslant \|\Phi(\vec{x}) - \Phi(\vec{y}_2)\|^2$. Using Eq. (3), one can rewrite the inequality as

$$\begin{aligned}&K(\vec{x},\vec{x}) + K(\vec{y}_1,\vec{y}_1) - 2K(\vec{x},\vec{y}_1) \\ &\quad \geqslant K(\vec{x},\vec{x}) + K(\vec{y}_2,\vec{y}_2) - 2K(\vec{x},\vec{y}_2).\end{aligned}$$

Since $K(\vec{a},\vec{a}) = 1$ and $K(\vec{a},\vec{b}) > 0 \forall\ \vec{a},\ \vec{b}$, the inequality is simplified as

$$K(\vec{x},\vec{y}_1) \leqslant K(\vec{x},\vec{y}_2).$$

Plugging the definition of RBF kernel, we obtain

$$\exp\left(-\frac{\|\vec{x}-\vec{y}_1\|^2}{2\sigma^2}\right) \leqslant \exp\left(-\frac{\|\vec{x}-\vec{y}_2\|^2}{2\sigma^2}\right),$$

which in turn can be simplified into

$$\|\vec{x} - \vec{y}_1\|^2 \geqslant \|\vec{x} - \vec{y}_2\|^2.$$

This is contradictory to our initial assumption that $\vec{y}_1$ is closer to $\vec{x}$ than $\vec{y}_2$. Thus the assumption that *the invariance of proximity does not hold* is not true. Therefore, kNN invariance holds for RBF kernel. □

Theorem 2 (kNN Invariance for Polynomial Kernel). *kNN invariance holds when the mapping function $\Phi(\vec{x})$ is defined such that*

$$\Phi(\vec{x}) \cdot \Phi(\vec{y}) = K(\vec{x},\vec{y}) = (\vec{x} \cdot \vec{y} + 1)^p,$$

and training patterns are all norm vectors ($\| \cdot \| = 1$).

Proof. Let $\vec{y}_1$ and $\vec{y}_2$ be two distinct neighbors of $\vec{x}$ with $\|\vec{x} - \vec{y}_1\|^2 < \|\vec{x} - \vec{y}_2\|^2$, i.e., $\vec{y}_1$ is closer to $\vec{x}$ than $\vec{y}_2$. Suppose the invariance of proximity does not hold for mapping function $\Phi(\vec{x})$, i.e., $\|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 \geqslant \|\Phi(\vec{x}) - \Phi(\vec{y}_2)\|^2$. Using Eq. (3), one can rewrite the inequality as

$$\begin{aligned} &K(\vec{x},\vec{x}) + K(\vec{y}_1,\vec{y}_1) - 2K(\vec{x},\vec{y}_1) \\ &\quad \geqslant K(\vec{x},\vec{x}) + K(\vec{y}_2,\vec{y}_2) - 2K(\vec{x},\vec{y}_2). \end{aligned}$$

Since $K(\vec{a},\vec{a}) = 2^p$ from $\vec{a} \cdot \vec{a} = 1$, the inequality becomes

$$K(\vec{x},\vec{y}_1) \leqslant K(\vec{x},\vec{y}_2).$$

Plugging the definition of polynomial kernel, one obtains

$$(\vec{x} \cdot \vec{y}_1 + 1)^p \leqslant (\vec{x} \cdot \vec{y}_2 + 1)^p.$$

The polynomial degree p can be eliminated from both sides since norm vectors always satisfy $-1 < \vec{a} \cdot \vec{b} < 1$, $\forall \vec{a} \neq \vec{b}$, and $(\vec{a} \cdot \vec{b} + 1) > 0$. Therefore, we get

$$\vec{x} \cdot \vec{y}_1 \leqslant \vec{x} \cdot \vec{y}_2.$$

The inner product between the patterns can be represented

$$\|\vec{x}\| \|\vec{y}_1\| \cos\theta_1 \leqslant \|\vec{x}\| \|\vec{y}_2\| \cos\theta_2$$

where θ_1 and θ_2 denote the angles between $\vec{x}$ and $\vec{y}_1$, and between $\vec{x}$ and $\vec{y}_2$, respectively. Since $\|\vec{x}\| = \|\vec{y}_1\| = \|\vec{y}_2\| = 1$, finally one has

$$\cos\theta_1 \leqslant \cos\theta_2,$$

and hence $\theta_1 \geqslant \theta_2$. In other words,

$$\|\vec{x} - \vec{y}_1\|^2 \geqslant \|\vec{x} - \vec{y}_2\|^2,$$

which is contradictory to an initial assumption that $\vec{y}_1$ is closer to $\vec{x}$ than $\vec{y}_2$. Thus the assumption that *the invariance of proximity does not hold* is not true, which yields that kNN invariance holds for polynomial kernel. □

Note that Theorem 2 does not hold without the norm vector assumption ($\| \cdot \| = 1$). Consider three patterns, $\vec{x} = 1$, $\vec{y}_1 = 2$, and $\vec{y}_2 = -1$ in the input space (1-dim). They satisfy

$$\|\vec{x} - \vec{y}_1\|^2 < \|\vec{x} - \vec{y}_2\|^2,$$

i.e., $\vec{y}_1$ is closer to $\vec{x}$ than $\vec{y}_2$. Let us consider $p = 2$ where $\Phi(a) \cdot \Phi(b) = K(\vec{a},\vec{b}) = (\vec{a} \cdot \vec{b} + 1)^p$. The distance between $\vec{x}$ and $\vec{y}_1$ in the feature space can be computed as

$$\begin{aligned} \|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 &= \|\Phi(1) - \Phi(2)\|^2 \\ &= K(1,1) + K(2,2) - 2K(1,2) \\ &= (1 \cdot 1 + 1)^2 + (2 \cdot 2 + 1)^2 \\ &\quad - 2(1 \cdot 2 + 1)^2 \\ &= 4 + 25 - 18 = 11. \end{aligned}$$

The distance between $\vec{x}$ and $\vec{y}_2$ in the feature space can be similarly computed, resulting in 8. Thus, we have $\|\Phi(\vec{x}) - \Phi(\vec{y}_1)\|^2 > \|\Phi(\vec{x}) - \Phi(\vec{y}_2)\|^2$, which violates the proximity invariance.

4. Conclusion

This paper provided a theoretical demonstration on our previous assumption that neighborhood relation is invariant under input to feature space mapping. In other words, the composition of k nearest neighbors surrounding a pattern in input space does not change after mapping to feature space. The result leads us to conclude that the patterns selected in input space are identical to the patterns selected in feature space. Thus, it is justified to select patterns in input space that are likely to be support vectors in feature space.

We now address some limitations. *First*, *fast* NPPS is geared up for efficient searching with initial random samples (see Fig. 4). A small proportion of random sampling, however, may cause a problem when the class distribution is multimodal. A small lump of patterns rather isolated could be excluded from the searching unless an initial random sample hits one of them. A current remedy for the risk is to check the class distribution first. If the class distribution falls on a multi-modal case, then we recommend to use *naive*

NPPS instead. Actually, we implemented *fast* NPPS to be able to easily switch to *naive* NPPS just by setting the sampling ratio to 1. *Second*, in case of polynomial kernel in Section 3, we assumed that all patterns of training set are *norm vectors* ($\|\cdot\| = 1$), which could be somewhat restrictive. However, it should be noted that in some machine learning applications such as text mining, it is conventional to preprocess a training set by scaling $\vec{x}' = \vec{x}/\|\vec{x}\|$ so as to calculate a cosine distance $\mathrm{dist}(\vec{x}, \vec{y}) = 1 - \vec{x}' \cdot \vec{y}'$.

Even though the proof should be similar in nature, we did not deal with sigmoid kernel here since it is a kernel with innate restrictions. Sigmoid kernel is a conditionally positive definite (CPD) kernel which satisfies Mercer's condition only for some values of the hyper-parameters ρ and δ, and only for norm vector (Schölkopf and Smola, 2002; Vapnik, 1999). Furthermore, sigmoid kernel has rarely been used since it has two hyper-parameters. It is a kernel devised for showing that SVM includes traditional MLP (multi-layer perceptron) neural network. A recent study of Lin and Lin (2003) on sigmoid kernel is worth reading. Even though sigmoid kernel seems to work well in a certain condition, it is not better than RBF. As RBF has properties of being PD and having fewer parameters, there is no reason to use the sigmoid.

Acknowledgments

This work was supported by the Korean Science and Engineering Foundation (KOSEF). And the authors are grateful to Dr. Se June Hong (IBM Research Division) and Dr. Jahwan Kim (KAIST) for their invaluable advice.

References

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

Cauwenberghs, G., Poggio, T., 2001. Incremental and decremental support vector machine learningAdv. Neural Inform. Process. Syst., vol. 13. MIT press, Cambridge, MA, pp. 409–415.

DMEF dataset Library. Available from <http://www.the-dma.org/dmef/dmefdset.shtml>.

Hearst, M.A., Schölkopf, B., Dumais, S., Osuna, E., Platt, J., 1997. Trends and controversies-support vector machines. IEEE Intell. Syst. 13, 18–28.

Kernel-Machines Organization. Available from <http://www.kernel-machines.org/>.

Koggalage, R., Halgamuge, S., 2004. Reducing the number of training samples for fast support vector machine classification. Neural Inform. Process. Lett. Rev. 2 (3), 57–65.

Lee, Y., Mangasarian O.L., 2001. RSVM: Reduced Support Vector Machines. In: CD Proc. of the SIAM International Conference on Data Mining, Chicago, April, SIAM, Philadelphia. Available from <http://www.cs.wisc.edu/olvi/olvi.html>.

Lin, H.-T., Lin, C.-J., 2003. A study on sigmoid kernels for SVM and the training of non-PSD Kernels by SMO-type methods. Available from <http://www.csie.ntu.edu.tw/cjlin/papers.html>.

Liu, C.L., Nakagawa, M., 2001. Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. Pattern Recognition Lett. 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 Trans. Neural Networks 10 (6), 1474–1481.

MNIST database. Available from <http://yann.lecun.com/exdb/mnist/>.

Platt, J.C., 1999. Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In: Adv. Kernel Meth.: Support Vector Machines. MIT Press, Cambridge, MA, pp. 185–208.

Pontil, M., Verri, A., 1998. Properties of Support Vector Machines. Neural Comput. 10, 955–974.

Schölkopf, B., Burges, D., Vapnik, V., 1995. Extracting support data for a given task. In: Proc. of 1st International Conference on Knowledge Discovery and Data Mining, AAAI, pp. 252–257.

Schölkopf, B., Smola, A.J., 2002. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press.

Shin, H.J., Cho, S., 2003a. Fast pattern selection for support vector classifiers. In: Proc. of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Lecture Notes in Artificial Intelligence (LNAI 2637), Seoul, Korea, pp. 376–387.

Shin, H.J., Cho, S., 2003b. Fast pattern selection algorithm for support vector classifiers: time complexity analysis. In: Proc. of the 3rd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Lecture Notes in Computer Science (LNCS 2690), Hong Kong, China, pp. 1008–1015.

Shin, H.J., Cho, S., in press. Response Modeling with Support Vector Machines.

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

UCI Machine Learning Repository. Available from <http://www.ics.uci.edu/~mlearn/>.

Vapnik, V., 1999. The Nature of Statistical Learning Theory, second ed. Springer.

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