Journal of the Chinese Statistical Association
Vol. 52, (2014) 85–114

A REVIEW OF REDUCED KERNEL TRICK IN MACHINE LEARNING

Yi-Ren Yeh[1] , Su-Yun Huang[2], Hsing-Kuo Pao[3] and Yuh-Jye Lee[3]

[1]Department of Applied Mathematics, Chinese Culture University

[2]Institute of Statistical Science, Academia Sinica

[3]Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology

ABSTRACT

We give a comprehensive introduction to the reduced support vector machine, its extensions and applications. We describe original RSVM algorithm and the statistical theory behind it. Three schemes for selecting the representative reduced set are introduced. These schemes lead to a smaller reduced set than the random sampling scheme without sacrificing prediction accuracy. Although smaller reduced set will have faster support vector machine training, one has to pay extra CPU time in learning the reduced set selection. In addition to classification, applications of reduced kernel trick to regression and dimension reduction are also included in this survey paper. We finally embed the RSVMs in the MapReduce framework for extremely large scale datasets. Some preliminary numerical studies show that RSVMs in MapReduce framework has a good potential for solving large scale nonlinear support vector machines. We believe that the reduced kernel trick will be an important technique in the Big Data era.

Key words and phrases: Big Data, Dimension Reduction, Kernel Methods, Reduced Kernel Trick, Supervised Learning, Support Vector Machine.

JEL classification: C1.

1. Introduction

In the last decade, support vector machines (SVMs) have become one of the most promising algorithms for classification and regression problems (Burges, 1998; Smola and Schölkopf, 2004) as well as other learning tasks such as dimension reduction (Alpaydin, 2004; Cook, 1998), semi-supervised learning (Pedrycz and Rai, 2008; Zhou and Li, 2005; Blum and Mitchell, 1998), etc. The success of SVMs is due to two main important reasons. First, the SVM algorithm is rooted in the statistical learning theory, namely, the structural risk minimization (Vapnik, 2000). By minimizing the training errors and model complexity at the same time, SVM classifier is able to cope with the high-dimensionality in nonparametric modeling so that it has good generalization ability. Secondly, the *kernel trick* has been introduced to bring the training data from the input space to a high dimensional even infinite dimensional feature space via an implicitly defined nonlinear mapping. Then, the SVM classifier is built in this feature space. Via the use of the kernel trick, variants of SVMs have successfully incorporated flexible and effective nonlinear models. However, there are some major challenges for nonlinear SVM with massive data due to the fully dense kernel matrix. This kernel matrix has size $\ell \times \ell$, where ℓ is the size of the training dataset. To overcome these computational difficulties, some researchers have proposed low-rank approximation to the full kernel matrix (Zhang et al., 2008; Williams and Seeger, 2001). The reduced support vector machine (RSVM, Lee and Huang (2007)) is an alternative. The key ideas of RSVM are as follows. Prior to training, it randomly selects a small portion of dataset to generate a *thin rectangular* kernel matrix. It is a small submatrix of the full *squared* kernel matrix. Then it uses this much smaller rectangular kernel matrix to replace the full kernel matrix in the nonlinear SVM formulation. Memory usage and the computational time will be cut down dramatically both in kernel matrix generating as well as in the SVM training. As a result, RSVM also simplifies the characterization of the nonlinear separating surface. Thus, it will be more efficient in the testing phase. According to intensive numerical comparisons in (Lee and Huang, 2007; Lee and Mangasarian, 2001), though RSVM usually has higher training error than the nonlinear SVM with full kernel, it has comparable, or even slightly smaller testing error. In other words,

RSVM has comparable, or slightly better generalization ability. This phenomenon can be interpreted by the Occam's razor (Blumer et al., 1987). Some theoretical investigation on reduced kernel can be found in (Lee and Huang, 2007; Chang et al., 2013). In this paper, we aim to give a comprehensive introduction to RSVM and its applications. We begin with an introduction to reduced kernel trick, functional approximation justification behind it and related statistical properties in Section 2 for support vector classification and regression. Random subset is the computationally cheapest way for obtaining reduced sets. In Section 3 we introduce three different sampling schemes other than the random subset, namely, incremental selection, systematical selection and selection by clustering. In Section 4, we include an extension of RSVM to dimension reduction. In Section 5, we describe RSVM under the MapReduce framework so that it has the ability to solve large scale nonlinear support vector machines. The reduced kernel trick has the potential as an important technique in the Big Data era. We conclude the paper in Section 6.

A word about our notation and background material is given below. All vectors will be column vectors unless transposed to a row vector by a superscript $\top$. For a vector $\mathbf{x}$ in the d-dimensional real space $\mathbb{R}^d$, the plus function $\mathbf{x}_+$ is defined as $(\mathbf{x}_+)_i = \max\{0, x_i\}$, while the step function $\mathbf{x}_*$ is defined as $(\mathbf{x}_*)_i = 1$ if $x_i > 0$ else $(\mathbf{x}_*)_i = 0$, $i = 1, \ldots, d$. The scalar (inner) product of two vectors $\mathbf{x}$ and $\mathbf{z}$ in the d-dimensional real space $\mathbb{R}^d$ will be denoted by $\mathbf{x}^\top\mathbf{z}$ and the p-norm of $\mathbf{x}$ will be denoted by $\|\mathbf{x}\|_p$. For a matrix $A \in \mathbb{R}^{\ell\times d}$, A_i is the ith row of A which is a row vector in $\mathbb{R}^d$. A column vector of ones of arbitrary dimension will be denoted by $\mathbf{1}$. For $A \in \mathbb{R}^{\ell\times d}$ and $B \in \mathbb{R}^{d\times m}$, the kernel $K(A, B)$ maps $\mathbb{R}^{\ell\times d} \times \mathbb{R}^{d\times m}$ into $\mathbb{R}^{\ell\times m}$. In particular, if $\mathbf{x}$ and $\mathbf{z}$ are column vectors in $\mathbb{R}^d$ then, $K(\mathbf{x}^\top, \mathbf{z})$ is a real number, $K(\mathbf{x}^\top, A^\top)$ is a *row* vector in $\mathbb{R}^\ell$ and $K(A, A^\top)$ is an $\ell \times \ell$ matrix. The difference of two sets A and B is defined as $A \setminus B = \{\mathbf{x} | \mathbf{x} \in A \text{ and } \mathbf{x} \notin B\}$.

2. Reduced Support Vector Machine: Classification and Regression

We consider the *supervised learning* problems, classification and regression, here. We are given a training dataset,

$$S = \{(\mathbf{x}_i, y_i) | \, i = 1, 2, \ldots, \ell\} \subset \mathbb{R}^d \times \mathcal{Y}.$$

Each instance $\mathbf{x}_i$ is a point in the d-dimensional real space $\mathbb{R}^d$ and comes with a *class label* $y_i \in \mathcal{Y}$. We would like to construct a decision function $h : \mathbb{R}^d \to \mathcal{Y}$ in an *inductive* way based on the given dataset S. The *image* of $\mathbf{x}_i$, $h(\mathbf{x}_i)$ should be equal to or very close to $y_i \in \mathcal{Y}$. Thus, we can *predict* the class label of a new instance $\mathbf{x}$ that does not appear in the given training dataset S with $h(\mathbf{x})$, the image of $\mathbf{x}$. For the classification problems, the *label* set $\mathcal{Y}$ is a *finite* set, for example $\mathcal{Y} = \{1, 2, \ldots, c\}$. In particular, $\mathcal{Y} = \{-1, +1\}$ will be a *binary* classification problem. If $\mathcal{Y} = \mathbb{R}$, then it will represent a regression problem.

2.1 Smooth Support Vector Machine

In classification problems, support vector machine (SVM) determines an optimal separating hyperplane that classifies data points into different categories. Here, "optimality" refers to the sense that the separating hyperplane has the best generalization ability for unseen data points, based on statistical learning theory. The conventional SVM (Vapnik, 2000) can be formulated as follows:

$$\begin{aligned} \min_{(\mathbf{w},b,\xi)\in\mathbb{R}^{d+1+\ell}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i \geq 1\,, \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \ldots, \ell\,, \end{aligned} \tag{1}$$

where $C > 0$ is a positive parameter that balances the weight of the penalty term $\sum_{i=1}^{\ell}\xi_i$ and the regularization term $\frac{1}{2}\|\mathbf{w}\|_2^2$. Minimizing $\frac{1}{2}\|\mathbf{w}\|_2^2$ is equivalent to *maximizing* the margin between two parallel bounding planes as well as *minimizing* the VC-error bound (Vapnik, 2000). By solving the optimization problem, we have the decision function $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ for the unseen instance $\mathbf{x}$.

The conventional support vector machine formulation (1) is a standard convex quadratic program (Bertsekas, 1999; Mangasarian, 1994; Nocedal andWright, 2006). The Wolfe dual problem of (1) is expressed as follows:

$$\begin{aligned} \min_{\alpha\in\mathbb{R}^\ell} \quad & \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^{\ell} y_i \alpha_i = 0\,, \\ & 0 \le \alpha_i \le C \quad \text{for } i = 1, 2, \ldots, \ell\,, \end{aligned} \tag{2}$$

where $\mathbf{x}_i^\top \mathbf{x}_j$ is the inner product of $\mathbf{x}_i$ and $\mathbf{x}_j$. The primal variable $\mathbf{w}$ is given by:

$$\mathbf{w} = \sum_{\alpha_i > 0} y_i \alpha_i \mathbf{x}_i\,. \tag{3}$$

The decision function is expressed by

$$h(\mathbf{x}) = \sum_{\alpha_i > 0} y_i \alpha_i \mathbf{x}_i^\top \mathbf{x} + b.$$

In contrast to the conventional SVM of (1), smooth support vector machine (SSVM, Lee and Mangasarian (2001)) minimizes the square of the 2-norm of the slack vector ξ. In addition, SSVM prefers a solution with a small value of b (also in 2-norm). This leads to the following minimization problem: (1) is expressed as follows:

$$\begin{aligned} \min_{(\mathbf{w},b,\xi)\in\mathbb{R}^{d+1+\ell}} \quad & \frac{1}{2}(\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{\ell}\xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i \ge 1 \\ & \xi_i \ge 0, \quad \text{for } i = 1, 2, \ldots, \ell\,. \end{aligned} \tag{4}$$

As a solution of (4), $\boldsymbol{\xi}$ is given by $\boldsymbol{\xi}_i = \{1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}_+$ for all i, where the plus function x_+ is defined as $x_+ = \max\{0, x\}$. Thus, we can replace $\boldsymbol{\xi}_i$ in (4) by $\{1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}_+$. It converts the problem (4) into an unconstrained minimization problem as follows:

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{d+1}} \quad \frac{1}{2}(\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{\ell}\{1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}_+^2\,. \tag{5}$$

Compared to (4), this formulation reduces the number of variables from $d+1+\ell$ to $d+1$. However, the objective function to be minimized is no longer twice differentiable. In SSVM, we prefer a twice differentiable form so that a fast Newton method can be applied. We approximate the plus function x_+ by a smooth p-function:

$$p(x,\beta) = x + \frac{1}{\beta}\log(1+e^{-\beta x})\,, \tag{6}$$

where $\beta > 0$ is the smooth parameter which controls the "steepness" of the curve or the closeness to the original plus function x_+. By replacing the plus function x_+ with a very accurate approximation p-function, it gives the SSVM formulation:

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{d+1}} \ \frac{1}{2}(\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{\ell} p(\{1-y_i(\mathbf{w}^\top\mathbf{x}_i+b)\},\beta)^2\,. \tag{7}$$

The objective function in problem (7) is strongly convex and infinitely differentiable. Hence, it has a unique solution and can be solved by using a fast *Newton-Armijo* algorithm which is globally and quadratically convergent (Lee and Mangasarian, 2001).

2.2 Reduced Smooth Support Vector Machine

In many cases, a dataset, as collected in a vector form full of attributes, cannot be well separated by a linear separating hyperplane. However, it is likely that the dataset becomes linearly separable after mapped into a higher dimensional space by a nonlinear map. A nice property of SVM methodology is that we do not even need to know the nonlinear map explicitly; still, we can apply a linear algorithm to the classification problem in the high dimensional space. The property comes from the dual form of SVM which can express the formulation in terms of inner product of data points. By taking the advantage of dual form, the "kernel trick" is used for the nonlinear extension of SVM. From the dual SVM formulation (2), all we need to know is simply the inner product between training data vectors. Let us map the training data points from the input space $\mathbb{R}^d$ to a higher-dimensional feature space $\mathcal{F}$ by a nonlinear map Φ. The training data $\mathbf{x}$ in $\mathcal{F}$ becomes $\Phi(\mathbf{x})$. Based on the above observation, if we know the inner product $\Phi(\mathbf{x}_i)^\top\Phi(\mathbf{x}_j)$ for all $i,j = 1,2,\ldots,\ell$, then we can perform the linear SVM algorithm in the feature space $\mathcal{F}$. The separating hyperplane will be linear in the feature space $\mathcal{F}$ but is a nonlinear surface in the input space $\mathbb{R}^d$ (see Figure 1).

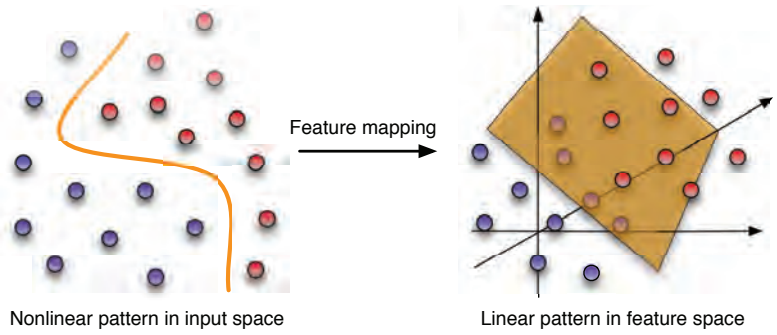

Figure 1: The illustration of nonlinear SVM

Note that we do not need to know the nonlinear map Φ explicitly. It can be achieved by employing a kernel function. Let $K(\mathbf{x}^\top, \mathbf{z}) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be an inner product kernel function satisfying *Mercer's condition* (Burges, 1998; Cherkassky and Mulier, 1998; Courant and Hilbert, 1953; Cristianini and Shawe-Taylor, 1999; Vapnik, 2000). We can construct a nonlinear map Φ such that $K(\mathbf{x}_i^\top, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$, $i, j = 1, 2, \ldots, \ell$. Hence, the linear SVM formulation can be used on $\Phi(\mathbf{x})$ in the feature space $\mathcal{F}$ by replacing the $\mathbf{x}_i^\top \mathbf{x}_j$ in the objective function of (2) with a nonlinear kernel function $K(\mathbf{x}_i^\top, \mathbf{x}_j)$. For the nonlinear case, the formulation of SSVM (7) can be extended to the nonlinear version by utilizing the kernel trick as follows:

$$\min_{(\mathbf{u},b)\in\mathbb{R}^{\ell+1}} \frac{1}{2}(\|\mathbf{u}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{\ell} p([1 - y_i\{\sum_{j=1}^{\ell} u_j K(\mathbf{x}_i^\top, \mathbf{x}_j) + b\}], \beta)^2, \tag{8}$$

where $K(\mathbf{x}_i^\top, \mathbf{x}_j)$ is a kernel function. The nonlinear SSVM decision function $h(\mathbf{x})$ can be expressed as follows:

$$h(\mathbf{x}) = \sum_{u_j \neq 0} u_j K(\mathbf{x}_j^\top, \mathbf{x}) + b\,. \tag{9}$$

Nowadays very often we have classification or regression problems with massive data, such as data from network traffic, gene expressions, web documents, etc. In a large scale SVM, the full kernel matrix will be very large and dense, so it may not be appropriate to use the full kernel matrix when dealing with (8). To avoid facing such a large and dense full kernel matrix, we brought in the *reduced* kernel technique (Lee and Huang, 2007; Chang et al., 2013). The key idea of the reduced kernel technique

is to randomly select a small portion of data and to generate a thin rectangular kernel matrix, then to use this much smaller rectangular kernel matrix to replace the full kernel matrix. In the process of replacing the full kernel matrix by a reduced kernel, we use the Nyström approximation (Smola and Schölkopf, 2000; Williams and Seeger, 2001) for the full kernel matrix:

$$K(A, A^\top) \approx K(A, \widetilde{A}^\top) K(\widetilde{A}, \widetilde{A}^\top)^{-1} K(\widetilde{A}, A^\top)\,, \tag{10}$$

where $\widetilde{A}_{\widetilde{\ell}\times d}$ is a subset of $A = \{\mathbf{x}_i\}_{i=1}^{\ell}$ and $K(A, \widetilde{A}^\top) = \widetilde{K}_{\ell\times\widetilde{\ell}}$ is a reduced kernel. Thus, we have

$$K(A, A^\top)\mathbf{u} \approx K(A, \widetilde{A}^\top) K(\widetilde{A}, \widetilde{A}^\top)^{-1} K(\widetilde{A}, A^\top)\mathbf{u} \;= K(A, \widetilde{A}^\top)\widetilde{\mathbf{u}}\,, \tag{11}$$

where $\widetilde{\mathbf{u}} \in \mathbb{R}^{\widetilde{\ell}}$. The reduced kernel technique was invented for dealing with large dataset. Thus, $\widetilde{\ell} \ll \ell$. By using the approximation, reduced SVM randomly selects a small subset $\widetilde{A}$ to generate the dictionary function set $\widetilde{\mathcal{B}}$:

$$\widetilde{\mathcal{B}} = \{1\} \cup \{K(\cdot, \mathbf{x}_i)\}_{i=1}^{\widetilde{\ell}}$$

The formulation of reduced SSVM, hence, is expressed as follows:

$$\min_{\widetilde{\mathbf{u}}, b} \qquad \frac{1}{2}(\|\widetilde{\mathbf{u}}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{\widetilde{\ell}} p([1 - y_i\{\sum_{j=1}^{\widetilde{\ell}} \widetilde{u}_j K(\mathbf{x}_i^\top, \mathbf{x}_j) + b\}], \beta)^2 \tag{12}$$

and its decision function is in the form

$$h(\mathbf{x}) = \sum_{j=1}^{\widetilde{\ell}} \widetilde{u}_j K(\mathbf{x}^\top, \mathbf{x}_j) + b\,. \tag{13}$$

The reduced kernel method constructs a compressed model and cuts down the computational cost for SSVM from $\mathcal{O}(\ell^3)$ to $\mathcal{O}(\widetilde{\ell}^3)$. It has been shown that the solution of reduced kernel matrix approximates the solution of full kernel matrix well and with theoretical error bounds (Lee and Huang, 2007; Chang et al., 2013).

2.3 ε-Support Vector Regression

In regression problems, the class label set $\mathcal{Y}$ belongs to real numbers. We would like to find a linear or nonlinear regression function, $h(\mathbf{x})$, that tolerates a small error in fitting the given dataset. It can be achieved by utilizing the ε-insensitive loss function that sets an ε-insensitive "tube" around fitted model. When the errors between the observed data values and fitted values are within the ε-tube, they are not counted into the loss function. Only errors go beyond the ε-tube are counted.

We start with the linear case, that is the regression function $h(\mathbf{x})$ defined as $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$. The SVR minimization can be formulated as an unconstrained problem given by:

$$\min_{(\mathbf{w},b,\xi)\in\mathbb{R}^{d+1+\ell}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{\ell} |\xi_i|_\varepsilon \,, \tag{14}$$

where $|\xi_i|_\varepsilon = \max\{0, |\mathbf{w}^\top \mathbf{x}_i + b - y_i| - \varepsilon\}$ represents the fitting error and the positive control parameter C here weights the tradeoff between the fitting errors and the flatness of the linear regression function $f(\mathbf{x})$. Similar to the idea in SVM, the regularization term $\|\mathbf{w}\|_2^2$ in (14) is also applied for improving the generalization ability. To deal with the ε-insensitive loss, conventionally it is reformulated as a constrained minimization problem defined as follows:

$$\begin{aligned} \min_{(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*)\in R^{d+1+2\ell}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{x}_i + b - y_i \le \varepsilon + \xi_i \,, \\ & -\mathbf{w}^\top \mathbf{x}_i - b + y_i \le \varepsilon + \xi_i^* \,, \\ & \xi_i,\, \xi_i^* \ge 0 \text{ for } i = 1, 2, \ldots, \ell \,. \end{aligned} \tag{15}$$

This formulation (15) is equivalent to the formulation (14) and its corresponding dual

form is

$$\begin{aligned} \max_{\boldsymbol{\alpha},\widehat{\boldsymbol{\alpha}}\in\mathbb{R}^\ell} \quad & \sum_{i=1}^{\ell}(\widehat{\alpha}_i-\alpha_i)y_i-\varepsilon\sum_{i=1}^{\ell}(\widehat{\alpha}_i+\alpha_i) \\ & -\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\widehat{\alpha}_i-\alpha_i)(\widehat{\alpha}_j-\alpha_j)\mathbf{x}_i^\top\mathbf{x}_j\,, \\ \text{s.t.} \quad & \sum_{i=1}^{\ell}(\widehat{\alpha}_i-\alpha_i)=0\,, \\ & 0\le\alpha_i,\widehat{\alpha}_i\le C\,, \quad \text{for } i=1\,,\ldots,\ell\,. \end{aligned} \tag{16}$$

From (16), one can apply the kernel trick on this dual form of ε-SVR for nonlinear extension. That is, $\mathbf{x}_i^\top\mathbf{x}_j$ is directly replaced by a kernel function $K(\mathbf{x}_i^\top,\mathbf{x}_j)$ as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha},\widehat{\boldsymbol{\alpha}}\in\mathbb{R}^\ell} \quad & \sum_{i=1}^{m}(\widehat{\alpha}_i-\alpha_i)y_i-\varepsilon\sum_{i=1}^{\ell}(\widehat{\alpha}_i+\alpha_i) \\ & -\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\widehat{\alpha}_i-\alpha_i)(\widehat{\alpha}_j-\alpha_j)K(\mathbf{x}_i^\top,\mathbf{x}_j)\,, \\ \text{s.t.} \quad & \sum_{i=1}^{\ell}(\widehat{\alpha}_i-\alpha_i)=0\,, \\ & 0\le\alpha_i,\widehat{\alpha}_i\le C\,, \quad \text{for } i=1\,,\ldots,\ell\,, \end{aligned} \tag{17}$$

with the decision function $h(\mathbf{x})=\sum_{i=1}^{\ell}(\widehat{\alpha}_i-\alpha_i)K(\mathbf{x}_i^\top,\mathbf{x})+b\,.$

Similar to the smooth approach in SSVM, the formulation (14) can be modified slightly as a smooth unconstrained minimization problem. Before we derive the smooth approximation function, we show some interesting observations:

$$|x|_\varepsilon=(x-\varepsilon)_+ +(-x-\varepsilon)_+ \tag{18}$$

and

$$(x-\varepsilon)_+\cdot(-x-\varepsilon)_+=0 \;\text{ for all } x\in\mathbb{R} \text{ and } \varepsilon>0\,. \tag{19}$$

Thus we have

$$|x|_\varepsilon^2 \;=\; (x-\varepsilon)_+^2+(-x-\varepsilon)_+^2\,. \tag{20}$$

It is straightforward to replace $|x|_\varepsilon^2$ by a very accurate smooth approximation given by:

$$p_\varepsilon^2(x,\beta) = (p(x-\varepsilon,\beta))^2 + (p(-x-\varepsilon,\beta))^2\,. \tag{21}$$

We use this approximation p_ε^2-function with smoothing parameter β to obtain the smooth support vector regression (ε-SSVR, Lee et al. 2005)):

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{d+1}} \frac{1}{2}(\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{\ell} p_\varepsilon^2(\mathbf{w}^\top\mathbf{x}_i + b - y_i, \beta)\,, \tag{22}$$

where $p_\varepsilon^2(\mathbf{w}^\top\mathbf{x}_i + b - y_i, \beta)$ is defined as (21). For the nonlinear case, this formulation can be extended to the nonlinear ε-SSVR by using the kernel trick as follows:

$$\min_{(\mathbf{u},b)\in\mathbb{R}^{\ell+1}} \frac{1}{2}(\|\mathbf{u}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{\ell} p_\varepsilon^2(\sum_{j=1}^{\ell} u_j K(\mathbf{x}_j^\top, \mathbf{x}_i) + b - y_i, \beta)\,, \tag{23}$$

where $K(\mathbf{x}_i^\top, \mathbf{x}_j)$ is a kernel function. The nonlinear ε-SSVR decision function $h(\mathbf{x})$ can be expressed as follows:

$$h(\mathbf{x}) = \sum_{i=1}^{\ell} u_i K(\mathbf{x}_j^\top, \mathbf{x}) + b\,. \tag{24}$$

Note that the reduced kernel technique can be applied to ε-SSVR when encountering a large scale regression problem.

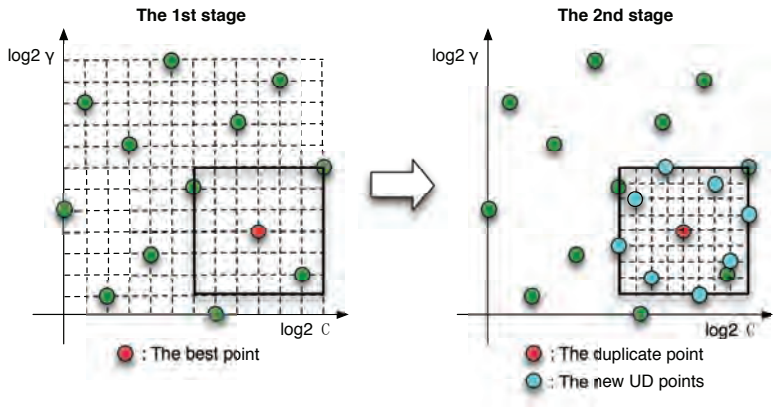

Figure 2: The nested uniform design (UD) model selection with a 13-points UD at the first stage and a 9-points UD at the second stage

2.4 Model Selection of SVMs

Choosing a good parameter setting for a better generalization performance of SVMs is the so called model selection problem. Model selection is usually done by minimizing an estimate of generalization error. This problem can be treated as finding the maximum (or minimum) of a function which is only vaguely specified and has many local maxima (or minima).

Suppose the Gaussian kernel

$$K(\mathbf{x}^\top, \mathbf{z}) = e^{-\gamma||\mathbf{x}-\mathbf{z}||_2^2},$$

is used, where γ is the width parameter. The nonlinear SVM requires two parameters C and γ. The most common approach for SVM model selection is probably based on cross-validation over a certain parameter grids, and an exhaustive grid search is adopted. The exhaustive grid search forms a two dimension uniform grid (say $p \times p$) of points in a pre-specified search range to find a good combination for C and γ. It is obvious that the exhaustive grid search is computationally expensive. Many improved methods have been proposed to reduce the number of trials in parameter combinations (Keerthi and Lin, 2003; Chapelle et al., 2002; Larsen et al., 1998; Bengio, 2000; Staelin, 2003; Huang et al., 2007). Here we focus on the 2-stage uniform design (UD) model selection in Huang et al. (2007) due to its good efficiency. The 2-stage UD procedure first sets out a crude search for a highly likely candidate region of global optimum and then confines a finer second-stage search therein. We use Figure 2 for a pictorial illustration of the 2-stage UD procedure. At the first stage, we use a 13-runs UD sampling pattern in an appropriate search range in log-scale. At the second stage, we halve the search range for each parameter coordinate and let the best point from the first stage be the center point of the new search box. Then we use a 9-runs UD sampling pattern in this new search box. Moreover, to deal with large sized datasets, we combine a 9-runs and a 5-runs sampling pattern at these two stages. The performance in (Huang et al., 2007) shows merits of the nested UD model selection method. Besides, the method of nested UDs is not limited to 2 stages and can be applied in a sequential manner and one may consider a finer net of UDs to start with.

3. Selection Schemes for Reduced Set

As we explained in the previous section, the decision function $h(\mathbf{x})$ is represented as a linear combination of the entire dictionary function set $\mathcal{B} = \{1\} \cup \{K(\cdot, \mathbf{x}_i)\}_{i=1}^{\ell}$ based on the given training set for the conventional nonlinear SVMs. While RSVM randomly selects a small dictionary subset $\overline{\mathcal{B}}$ to represent the decision function $h(\mathbf{x})$. Although the original random selection scheme is very simple to implement and has a good theoretical foundation (Lee and Huang, 2007), it will need a sufficient large reduced set to guarantee the performance. In this section, we describe three schemes for selecting the most *informative* instances using different criteria.

3.1 Incremental Reduced Support Vector Machines

In this section, we introduce Incremental Reduced Support Vector Machine (IRSVM) algorithm that automatically and incrementally selects informative data points from the training set to generate the thin rectangular kernel matrix used in RSVM (Lee et al., 2003). The decision function $h(\mathbf{x})$ is represented as a linear combination of kernel functions in the reduced dictionary subset. Intuitively, if the kernel functions in the dictionary subset are very "*similar*", the hypothesis space spanned by this dictionary subset will be very limited. Based on this intuition, Lee et al. (2003) propose a process that sequentially adding a kernel function into the dictionary subset, only when the function is "*dissimilar*" to the current dictionary subset and carrying sufficient extra information over the current subset. Here the *dissimilarity* is measured by the distance between a kernel vector and the column space of the rectangular kernel matrix generated by current dictionary subset. Suppose that we start with a very small reduced set $\widetilde{A}$, typically a size of 2, then we add a new data point $\mathbf{x}_i$ into the reduced set only when the extra information carried in the vector $K(A, \mathbf{x}_i) \in \mathbb{R}^{\ell \times 1}$ with respect to the column space of $K(A, \widetilde{A}^\top)$ is greater than a certain positive threshold. This can be achieved by solving a least squares problem. Let $\widetilde{K} = K(A, \widetilde{A}^\top) \in \mathbb{R}^{\ell \times \widetilde{\ell}}$. The least squares problem we need to solve is

$$\min_{\beta \in \mathbb{R}^{\widetilde{\ell}}} \left\| \widetilde{K}\beta - K(A, \mathbf{x}_i) \right\|_2^2, \tag{25}$$

where $\beta \in \mathbb{R}^{\widetilde{\ell}}$ is a free vector variable and $\widetilde{K}\beta \in \mathbb{R}^{\ell}$ is a linear combination of the functions $K(A, \mathbf{x}_i), i = 1, \ldots, \widetilde{\ell}$ that represents a vector in the column space of $K(A, \widetilde{A}^\top)$. According to the first order optimality condition (Bertsekas, 1999; Mangasarian, 1994), finding out the optimal solution β^* of above problem (25) is equivalent to solving a system of normal equations:

$$\widetilde{K}^\top \widetilde{K}\beta = \widetilde{K}^\top K(A, \mathbf{x}_i). \tag{26}$$

If the columns of the rectangular kernel matrix generated by the initial reduced set are linearly independent, our IRSVM algorithm will keep the independence property throughout the whole process, so that the least squares problem (25) has a unique solution β^*,

$$\beta^* = (\widetilde{K}^\top \widetilde{K})^{-1} \widetilde{K}^\top K(A, \mathbf{x}_i). \tag{27}$$

The distance r from $K(A, \mathbf{x}_i)$ to the column space of $\widetilde{K}$ is the squared root of the optimal value of (25) and is computed by

$$r = \left\| \widetilde{K}\beta^* - K(A, \mathbf{x}_i) \right\|_2. \tag{28}$$

The square distance can be written in the form $r^2 = (I - P)K(A, \mathbf{x}_i)$, where $P = \widetilde{K}(\widetilde{K}^\top \widetilde{K})^{-1}\widetilde{K}^\top$ is the projection matrix of $\mathbb{R}^{\widetilde{\ell}}$ onto the column space of $\widetilde{K}$. In other words, r^2 represents the amount of the extra information carried in $K(A, \mathbf{x}_i)$ over $K(A, \widetilde{A}^\top)$. Note that the size of the reduced set is very small, hence it will not lead to any computational difficulty in solving the least squares problem, though we have to solve it many times in the whole process. In fact, the time complexity of this step is $O(\widetilde{\ell}^3)$, where $\widetilde{\ell}$ is the current size of the reduced set. That is the main cost of solving the normal equations depends on $\widetilde{K}^\top \widetilde{K}$, but not on $K(A, \mathbf{x}_j)$. We describe the IRSVM algorithm in Algorithm 1.

A bigger δ will make a new instance be more difficult to be added into the reduced set. If the IRSVM algorithm ends up with a too small reduced set then the performance of the RSVM may not be satisfactory. In this case, we should lower down the threshold value. A reduced set generated by a bigger threshold value can be used as the initial reduced set for a new threshold.

Algorithm 1 Incremental Reduced Support Vector Machine (IRSVM)

Require: Given a certain threshold δ and a small initial reduced set $\widetilde{A}_0$.

Ensure: Return a reduced set, $\widetilde{A}$ and the corresponding rectangular kernel matrix, $K(A, \widetilde{A}^\top)$ for RSVM.

0) Let $\widetilde{A}_{new} = \widetilde{A}_0$ and compute $K(A, \widetilde{A}_{new}^\top)$.

1) Randomly choose an instance $\mathbf{x}_j \in A \setminus \widetilde{A}_{new}$.

2) Compute the distance r between the kernel vector $K(A, \mathbf{x}_j)$ and the column space of $K(A, \widetilde{A}_{new}^\top)$ by using the equation (28).

3) IF $r \geq \delta$ THEN $\widetilde{A}_{new} = \widetilde{A}_{new} \cup \mathbf{x}_j$

4) Repeat Step 1) to Step 3) until several successive points are fail in Step 3).

5) Output $\widetilde{A} = \widetilde{A}_{new}$ and $K(A, \widetilde{A}^\top) = K(A, \widetilde{A}_{new}^\top)$.

Algorithm 2 Systematic Sampling Algorithm for Reduced Set

Require: Given an extremely small initial reduced set $\widetilde{A}_0$, the RSVM decision function $h_0(\mathbf{x})$ based on $\widetilde{A}_0$ and a *validation dataset* and a threshold.

Ensure: The final RSVM decision function $h(\mathbf{x})$.

0) Let I_+ be the *index* set of misclassified instances of positive examples from the training set, i.e., $I_+ = \{i | h(\mathbf{x}^i) < 0,\ \mathbf{x}^i \text{ is a positive instance}\}$. Similarly, let $I_- = \{i | h(\mathbf{x}^i) > 0,\ \mathbf{x}^i \text{ is a negative instance}\}$.

1) Sort the sets I_+ and I_- by the values of $h(\mathbf{x}^j)$ separately. Let $\overline{S}_+$ and $\overline{S}_-$ be the resulting sorted sets.

2) Partition $\overline{S}_+$ and $\overline{S}_-$ into k (roughly) equal-sized subsets.

3) Randomly select an instance from each subset to form a new reduced set and generate the RSVM decision function $h(\mathbf{x})$ by using this new reduced set.

4) Evaluate the performance of $h(\mathbf{x})$ on the given validation set.

5) Repeat Step 0) to Step 4) until the accuracy of $h(\mathbf{x})$ on the validation excesses the threshold.

6) Return the decision function $h(\mathbf{x})$.

3.2 Generating the Reduced Set by Systematic Sampling

The second scheme for selecting the reduced set is based on the systematic sampling strategy (Chien et al., 2010). We start with an extremely small reduced set $\widetilde{A}_0$. Imagine that the decision function $h_0(\mathbf{x})$ will then be poor due to the insufficient size of the reduced set. There will be quite some $misclassified$ points $\mathbf{x}_i$ such that $y_i \cdot h_0(\mathbf{x}_i) < 0$.

How should we improve upon the current decision function? We suggest to treat these misclassified points as candidate instances for a new reduced set. A *uniform sampling* scheme is adopted to choose a small portion of data from this candidate set and use them to expand the reduced set. This procedure can be repeated until the decision function $h(\mathbf{x})$ achieves a satisfactory level. We describe this procedure in the Algorithm 2.

3.3 Clustering Model Selection for Reduced Support Vector Machines

The third scheme is based on the k-means clustering algorithm (Chien et al., 2010). Within each class, k-means algorithm is used to find representative points. In this scheme, we also explore the distribution of each cluster. That will suggest a *radius* estimation for each cluster. We will use this information to *tune* the *width* parameter in the Gaussian kernel. The Gaussian kernel is also named *radial basis function* (RBF) kernel. Different from the previous three schemes, including the random selection, the dictionary function set in this approach is more flexible. In fact, the reduced set generated by this mechanism is not a subset of original training dataset. Before we state the algorithm, we look at the form of Gaussian or RBF kernel,

$$K(\mathbf{x}, \mathbf{c}) = e^{-\frac{\|\mathbf{x}-\mathbf{c}\|_2^2}{2\sigma^2}},$$

where σ is the width parameter and $\mathbf{c}$ is the centroid. In Oyang and Hwang (2002), the authors propose an estimation for the width parameter σ as follows,

$$\sigma = \frac{\overline{R}(\mathbf{c}) \cdot \delta \cdot \sqrt{\pi}}{\sqrt[d]{(n_r+1)\Gamma(\frac{d}{2}+1)}}, \text{ where } \delta \cdot \sqrt{\pi} = 1.6210 \tag{29}$$

and $\overline{R}(\mathbf{c})$ is defined as

$$\overline{R}(\mathbf{c}) = \frac{d+1}{d}\left(\frac{1}{n_r}\sum_{q=1}^{n_r}\|\mathbf{x}_q - \mathbf{c}\|_2\right), \tag{30}$$

where d is the dimensionality, $\Gamma(\cdot)$ is the Gamma function, and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_r}$ are the n_r *nearest* instances to the cluster centroid $\mathbf{c}$. In the case, where the cluster size is smaller than n_r, we use all instances in that cluster to compute $\overline{R}(\mathbf{c})$. This estimation of the width parameter reflects the distribution of the clusters in d-dimensional space. The detailed procedure is presented in Algorithm 3.

Algorithm 3 Clustering Reduced Support Vector Machine

Require: Let k be the number of clusters within each class and n_r be a positive integer for n_r-nearest neighbors.

Ensure: Return the RSVM decision function $h(\mathbf{x})$.

0) For both classes, run the k-means algorithm to find the cluster centroids $\{\mathbf{c}_j^+, \mathbf{c}_j^-\}_{j=1}^k$.

1) Use these cluster centroids from both classes to form the reduced set: $\widetilde{A} = [\mathbf{c}_1^+, \mathbf{c}_2^+, \ldots, \mathbf{c}_k^+, \mathbf{c}_1^-, \mathbf{c}_2^-, \ldots, \mathbf{c}_k^-]^\top \in \mathbb{R}^{2k\times d}$.

2) For each centroid, compute the width parameters σ_j^+ and σ_j^- by (29) for RBF kernel functions : $\sigma_j^+ = \frac{\overline{R}(\mathbf{c}_j^+)\cdot\delta\cdot\sqrt{\pi}}{\sqrt[d]{(n_r+1)\Gamma(\frac{d}{2}+1)}}$ or $\sigma_j^- = \frac{\overline{R}(\mathbf{c}_j^-)\cdot\delta\cdot\sqrt{\pi}}{\sqrt[d]{(n_r+1)\Gamma(\frac{d}{2}+1)}}$ for $1 \leqslant j \leqslant k$.

3) Generate the rectangular kernel matrix $K(A, \widetilde{A}^\top) \in \mathbb{R}^{\ell\times 2k}$ and its columns are $K(A, \mathbf{c}_j^+) = [e^{-\frac{\|\mathbf{x}_1-\mathbf{c}_j^+\|_2^2}{2\sigma_j^{+2}}}, e^{-\frac{\|\mathbf{x}_2-\mathbf{c}_j^+\|_2^2}{2\sigma_j^{+2}}}, \ldots, e^{-\frac{\|\mathbf{x}_\ell-\mathbf{c}_j^+\|_2^2}{2\sigma_j^{+2}}}]^\top$ or $K(A, \mathbf{c}_j^-) = [e^{-\frac{\|\mathbf{x}_1-\mathbf{c}_j^-\|_2^2}{2\sigma_j^{-2}}}, e^{-\frac{\|\mathbf{x}_2-\mathbf{c}_j^-\|_2^2}{2\sigma_j^{-2}}}, \ldots, e^{-\frac{\|\mathbf{x}_\ell-\mathbf{c}_j^-\|_2^2}{2\sigma_j^{-2}}}]^\top$ for $1 \leqslant j \leqslant k$.

4) Generate the RSVM decision function $h(\mathbf{x})$ with $K(A, \widetilde{A}^\top)$.

4. Applications to Dimension Reduction

Dimension reduction is an important topic in machine learning and data mining. The main demand comes from complex data analysis, data visualization, and parsimonious modeling. Modern data are usually complex, high dimensional, and with nonlinear structures. Dimension reduction techniques can help us to characterize the key data structure using only a few main features ranked by their importance. It thus provides a way for data visualization to gain better intuitive insights of the underlying data. In this section, we will apply our reduced kernel technique to three classical dimension reduction methods: principal component analysis, sliced inverse regression, and canonical correlation analysis.

4.1 Kernel Principal Component Analysis

PCA is a well known unsupervised dimension reduction method, which determines the principal directions of the data distribution. To obtain these principal directions, one needs to construct the data covariance matrix and calculate its dominant eigenvectors. These eigenvectors will be the *most informative* among the vectors

in the original data space, and are thus considered as the principal directions. Let $A = [\mathbf{x}_1^\top; \mathbf{x}_2^\top; \ldots; \mathbf{x}_\ell^\top] \in \mathbb{R}^{\ell \times d}$, where each row $\mathbf{x}_i^\top$ represents a data instance in a d-dimensional space, and ℓ is the number of the instances. Typically, PCA is formulated as the following optimization problem

$$\max_{U \in \mathbb{R}^{d \times k}, U^\top U = I_k} \sum_{i=1}^{\ell} U^\top (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^\top U, \tag{31}$$

where U is a matrix consisting of k dominant eigenvectors and $\overline{\mathbf{x}}$ is the global mean. From this formulation, one can see that the standard PCA can be viewed as a task of determining a subspace where the projected data has the largest variation.

Generally, the problem in (31) can be solved by deriving an eigenvalue decomposition of the covariance matrix, i.e.,

$$\Sigma_A U = U\Lambda, \tag{32}$$

where the covariance matrix is given by

$$\Sigma_A = \frac{1}{\ell} \sum_{i=i}^{\ell} (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^\top. \tag{33}$$

Each column of U represents an eigenvector of Σ_A, and the corresponding diagonal entry in Λ is the associated eigenvalue. For the purpose of dimension reduction, the last few eigenvectors will be discarded due to their negligible contribution to the data variation.

Similar to SVM, the kernel trick can also be applied to PCA and it results in kernel PCA (KPCA, Schölkopf et al. (1998)). Let Φ be a nonlinear mapping from $\mathcal{X}$ to a feature space $\mathcal{F}$, and the kernel function $K(\mathbf{x}^\top, \mathbf{z}) = \Phi(\mathbf{x})^\top \Phi(\mathbf{z})$. Using the kernel trick, KPCA can be formulated as follows:

$$\Sigma_K \alpha = \lambda \alpha, \tag{34}$$

where

$$\Sigma_K = \frac{1}{\ell} \sum_{i=1}^{\ell} (K(A, \mathbf{x}_i) - \overline{\mathbf{k}})(K(A, \mathbf{x}_i) - \overline{\mathbf{k}})^\top \in \mathbb{R}^{\ell \times \ell}, \tag{35}$$

where $\overline{\mathbf{k}} = \frac{1}{\ell}\sum_{i=1}^{\ell} K(A, \mathbf{x}_i)$. When the sample size ℓ is large, the reduced kernel technique introduced in Section 2 can be used to replace the full kernel to speed up the computation. Let $\widetilde{A} \in \mathbb{R}^{\widetilde{\ell} \times d}$ be the reduced set (often $\widetilde{\ell} \ll \ell$), and the reduced KPCA (Huang et al., 2009b) can be formulated as follows:

$$\Sigma_{\widetilde{K}}\widetilde{\alpha} = \lambda\,\widetilde{\alpha}, \tag{36}$$

where $\widetilde{\alpha} \in \mathbb{R}^{\widetilde{\ell}}$ and $\Sigma_{\widetilde{K}}$ is the covariance matrix of reduced kernel data $K(A, \widetilde{A}^{\top})$ given by

$$\Sigma_{\widetilde{K}} = \frac{1}{\ell}\sum_{i=1}^{\ell}(K(\widetilde{A}, \mathbf{x}_i) - \widetilde{\mathbf{k}})(K(\widetilde{A}, \mathbf{x}_i) - \widetilde{\mathbf{k}})^{\top} \in \mathbb{R}^{\widetilde{\ell} \times \widetilde{\ell}}, \tag{37}$$

where $\widetilde{\mathbf{k}} = \frac{1}{\ell}\sum_{i=1}^{\ell} K(\widetilde{A}, \mathbf{x}_i)$. The reduced kernel method cuts the computational cost from $O(\ell^3)$ to $O(\widetilde{\ell}^3)$.

4.2 Kernel Sliced Inverse Regression

Sliced inverse regression (SIR) is first introduced by (Li, 1991), which shows that the pattern *e.d.r.* subspace can be estimated by the leading directions from the generalized eigenvalue decomposition of $Cov\{E(\mathbf{x}|\mathbf{y})\}$, denoted by $\Sigma_{E(\mathbf{x}|\mathbf{y})}$, with respect to $Cov(\mathbf{x})$, denoted by $\Sigma_{\mathbf{x}}$. With $\mathbf{x}$ being standardized, SIR finds the leading directions, in which the inverse regression function $\mathbf{g}(\mathbf{y}) := E(\mathbf{x}|\mathbf{y})$ has the largest variation. These are the most informative directions in the input pattern space for describing $\mathbf{y}$. In practical supervised learning tasks, the joint distribution of the input vector $\mathbf{x}$ and the output variable $\mathbf{y}$ is unknown. The empirical data version of sliced inverse regression finds the dimension reduction directions by solving the following generalized eigenvalue problem:

$$\Sigma_{E(A|Y_J)}\beta = \lambda\Sigma_A\beta, \tag{38}$$

where Σ_A is the sample covariance matrix of A, Y_J denotes the membership in slices and there are J many slices, and $\Sigma_{E(A|Y_J)}$ denotes the between-slice sample covariance matrix based on slice means given by

$$\Sigma_{E(A|Y_J)} = \frac{1}{\ell}\sum_{j=1}^{J}\ell_j(\overline{\mathbf{x}}^j - \overline{\mathbf{x}})(\overline{\mathbf{x}}^j - \overline{\mathbf{x}})^{\top}.$$

Here $\overline{\mathbf{x}}$ is the sample grand mean, and $\overline{\mathbf{x}}^j = \frac{1}{\ell_j}\sum_{i\in S_j}\mathbf{x}_i^j$ is the sample mean the jth slice where S_j is the index set and ℓ_j is the sample size for the jth slice, respectively. Note that the slices are extracted from A according to the sorted responses Y. For classification, $\overline{\mathbf{x}}^j$ is simply the sample mean of input attributes for the jth class. The solution, denoted by β_1, gives the first *e.d.r.* direction such that slice means projected along β_1 are most spreading out, where β_1 is normalized with respect to the sample covariance matrix Σ_A. Repeatedly solving this optimization problem with the orthogonality constraints $\beta_k^\top \Sigma_A \beta_l = \delta_{k,l}$, where $\delta_{k,l}$ is the Kronecker delta, the sequence of solutions $\beta_1, \ldots, \beta_d$ forms the basis for *e.d.r.* subspace. Some insightful discussion to enhance the SIR methodology and applications can be found in Chen and Li (1998).

The classical SIR is designed to find a *linear* transformation from the input space to a low dimensional *e.d.r.* subspace that keeps as much information as possible for the output variable $\mathbf{y}$. However, it does not work for nonlinear feature extraction. To improve the performance for nonlinear feature extraction, the kernel SIR (KSIR) can easily describe key features by a few nonlinear components (Yeh et al., 2009; Wu, 2008). Similar to KPCA, the vectors β can be represented by the linear combination of feature data by kernel map. It then leads to the KSIR formulation:

$$\Sigma_{E(K|Y_J)}\alpha = \lambda \Sigma_K \alpha, \tag{39}$$

where Σ_K is the sample kernel covariance matrix of $K(A, A^\top)$ and $\Sigma_{E(K|Y_J)}$ denotes the between-slice sample kernel covariance matrix based on slice means given by

$$\Sigma_{E(K|Y_J)} = \frac{1}{\ell}\sum_{j=1}^{J}\ell_j(\overline{\mathbf{k}}^j - \overline{\mathbf{k}})(\overline{\mathbf{k}}^j - \overline{\mathbf{k}})^\top,$$

where $\overline{\mathbf{k}}^j = \frac{1}{\ell_j}\sum_{i=1}^{\ell_j}K(A, \mathbf{x}_i^j)$ and $\overline{\mathbf{k}} = \frac{1}{\ell}\sum_{i=1}^{\ell}K(A, \mathbf{x}_i)$.

Similar to KPCA, using full kernel encounters a heavy computing load (even not doable), and the effective rank of the covariance matrix of kernel data is very low. The reduced kernel technique can also be applied to KSIR. Let $\widetilde{A} \in \mathbb{R}^{\widetilde{\ell}\times d}$ be the reduced set, and The reduced KSIR can be formulated as follows:

$$\Sigma_{E(\widetilde{K}|Y_J)}\widetilde{\alpha} = \lambda \Sigma_{\widetilde{K}}\widetilde{\alpha}, \tag{40}$$

where $\widetilde{\alpha} \in \mathbb{R}^{\widetilde{\ell}}$ and $\widetilde{K} = K(A, \widetilde{A}^\top)$. This approximation will enhance the numerical stability without much information loss.

4.3 Kerenl Canonical Correlation Analysis

Given two sets of ℓ unlabeled observations $A^s = [\mathbf{x}_1^s, \ldots, \mathbf{x}_\ell^s]^\top \in \mathbb{R}^{\ell \times d_s}$ and $A^t = [\mathbf{x}_1^t, \ldots, \mathbf{x}_\ell^t]^\top \in \mathbb{R}^{\ell \times d_t}$, CCA learns the projection vectors $\mathbf{u}^s \in \mathbb{R}^{d_s}$ and $\mathbf{u}^t \in \mathbb{R}^{d_t}$, which maximize the correlation coefficient ρ:

$$\max_{\mathbf{u}^s, \mathbf{u}^t} \rho = \frac{\mathbf{u}^{s\top} \Sigma_{st} \mathbf{u}^t}{\sqrt{\mathbf{u}^{s\top} \Sigma_{ss} \mathbf{u}^s} \sqrt{\mathbf{u}^{t\top} \Sigma_{tt} \mathbf{u}^t}}, \tag{41}$$

where $\Sigma_{st} = \frac{1}{\ell} \sum_{i=1}^{\ell} (\mathbf{x}_i^s - \overline{\mathbf{x}}^s)(\mathbf{x}_i^t - \overline{\mathbf{x}}^t)^\top$, $\Sigma_{ss} = \frac{1}{\ell} \sum_{i=1}^{\ell} (\mathbf{x}_i^s - \overline{\mathbf{x}}^s)(\mathbf{x}_i^s - \overline{\mathbf{x}}^s)^\top$, $\Sigma_{tt} = \frac{1}{\ell} \sum_{i=1}^{\ell} (\mathbf{x}_i^t - \overline{\mathbf{x}}^t)(\mathbf{x}_i^t - \overline{\mathbf{x}}^t)^\top$, and $\rho \in [0, 1]$. As proved in Hardoon et al. (2004), $\mathbf{u}^s$ in (41) can be solved by a generalized eigenvalue decomposition problem:

$$\Sigma_{st} (\Sigma_{tt})^{-1} \Sigma_{st}^\top \mathbf{u}^s = \eta \Sigma_{ss} \mathbf{u}^s. \tag{42}$$

Once $\mathbf{u}^s$ is obtained, $\mathbf{u}^t$ can be calculated by $\Sigma_{tt}^{-1} \Sigma_{st} \mathbf{u}^s / \eta$. Generally, one can derive more than one pair of projection vectors $\{\mathbf{u}_j^s\}_{j=1}^d$ and $\{\mathbf{u}_j^t\}_{j=1}^d$ with corresponding ρ_j in descending order (i.e., $\rho_j \geq \rho_{j+1}$) by the generalized eigenvalue decomposition problem (42). Note that d is the dimensionality of the resulting CCA subspace.

Similar to SVM, the use of linear models for CCA might not be sufficient in describing data when deriving the feature subspace. To overcome this problem, nonlinear mapping via kernel trick can be applied to CCA and it results in kernel CCA (KCCA, Hardoon et al. (2004); Huang et al. (2009a)) as follows:

$$\max_{\alpha^s, \alpha^t} \rho = \frac{\alpha^{s\top} \Sigma_{K_{st}} \alpha^t}{\sqrt{\alpha^{s\top} \Sigma_{K_s} \alpha^s} \sqrt{\alpha^{t\top} \Sigma_{K_t} \alpha^t}}, \tag{43}$$

where $\alpha^s \in \mathbb{R}^\ell$ and $\alpha^t \in \mathbb{R}^\ell$ are the projection vectors in the feature space,

$$\Sigma_{K_{st}} = \frac{1}{\ell} \sum_{i=1}^{\ell} (K(A^s, \mathbf{x}_i^s) - \overline{\mathbf{k}}^s)(K(A^t, \mathbf{x}_i^t) - \overline{\mathbf{k}}^t)^\top, \tag{44}$$

$$\Sigma_{K_{ss}} = \frac{1}{\ell} \sum_{i=1}^{\ell} (K(A^s, \mathbf{x}_i^s) - \overline{\mathbf{k}}^s)(K(A^s, \mathbf{x}_i^s) - \overline{\mathbf{k}}^s)^\top, \tag{45}$$

and

$$\Sigma_{K_{tt}} = \frac{1}{\ell}\sum_{i=1}^{\ell}(K(A^t, \mathbf{x}_i^t) - \overline{\mathbf{k}}^t)(K(A^t, \mathbf{x}_i^t) - \overline{\mathbf{k}}^t)^\top. \tag{46}$$

Note that $\overline{\mathbf{k}}^s = \frac{1}{\ell}\sum_{i=1}^{\ell} K(A^s, \mathbf{x}_i^s)$ and $\overline{\mathbf{k}}^t = \frac{1}{\ell}\sum_{i=1}^{\ell} K(A^t, \mathbf{x}_i^t)$. As the covariance matrices Σ_{K_s} and Σ_{K_t} in denominator are singular and their effective ranks are low, it can cause some numerical difficulty in computing the solution pair $\{\alpha^s, \alpha^t\}$. To overcome this problem, techniques like adding a additional regularization term or advancing reduced kernels have been proposed.

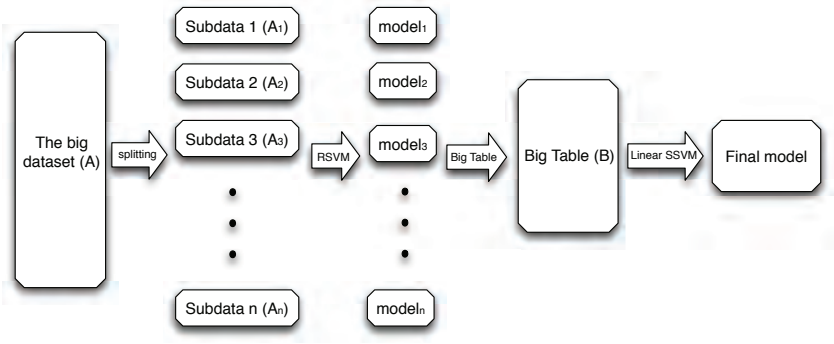

Figure 3: The MapReduce RSVM framework

For applying the reduced kernel technique to KCCA (Huang et al., 2009a), we let $\widetilde{A}^s$ and $\widetilde{A}^t$ be subsets of A^s and A^t, and then we can replace the full kernel matrices $K(A^s, A^{s\top}) \in \mathbb{R}^{\ell\times\ell}$ and $K(A^t, A^{t\top}) \in \mathbb{R}^{\ell\times\ell}$ by the reduced kernel matrices $K(A^s, \widetilde{A}^{s\top}) \in \mathbb{R}^{\ell\times\widetilde{\ell}}$ and $K(A^t, \widetilde{A}^{t\top}) \in \mathbb{R}^{\ell\times\widetilde{\ell}}$, respectively. The reduced kernel version for the KCCA now solves the following problem:

$$\max_{\widetilde{\alpha}^s, \widetilde{\alpha}^t} \rho = \frac{\widetilde{\alpha}^{s\top}\Sigma_{\widetilde{K}_{st}}\widetilde{\alpha}^t}{\sqrt{\widetilde{\alpha}^{s\top}\Sigma_{\widetilde{K}_s}\widetilde{\alpha}^s}\sqrt{\widetilde{\alpha}^{t\top}\Sigma_{\widetilde{K}_t}\widetilde{\alpha}^t}}, \tag{47}$$

where $\Sigma_{\widetilde{K}_{st}}$, $\Sigma_{\widetilde{K}_s}$, and $\Sigma_{\widetilde{K}_t}$ are calculated by the same definition as (44), (45), (46) with the reduced kernel matrices, respectively. Similar to (42), we can solve (47) via

$$\Sigma_{\widetilde{K}_{st}}(\Sigma_{\widetilde{K}_{tt}})^{-1}\Sigma_{\widetilde{K}_{st}}^\top\widetilde{\alpha}^s = \eta\Sigma_{\widetilde{K}_{ss}}\widetilde{\alpha}^s. \tag{48}$$

The reduced kernel method reduces the computational cost from $O(\ell^3)$ to $O(\widetilde{\ell}^3)$ (typically $\widetilde{\ell} \ll \ell$).

5. Big Data Applications

Nowadays more and more data can be collected easily from different applications, e.g., sensor networks, social networks, Internet documents, etc. How to deal with such a large amount of data is a critical task for data analysts. To efficiently solve for a nonlinear support vector machine with an extremely large dataset, we apply the RSVM under the MapReduce framework for nonlinear model (as shown in Figure 3). In our MapReduce RSVM (MRRSVM), the training procedure consists of steps of Map and Reduce. In the Map-step, the original data are split into n stratified disjoint subsets. Each data subset is used to learn an RSVM model associated with a reduced subset distributively. Note that the associated reduced subset is much smaller than the corresponding data subset. After processing these n RSVM models independently, these models will be combined and concluded in the Reduce-step. More specifically, in the Reduce-step we first collect all the instances from these n reduced subsets, and next we encode each of these instances as an n-dimensional vector by its predicted values made by these n RSVM models. That is, we can have a much more compact representation for the reduced subsets (called Big Table in Figure 3), where the data size is equal to the size of all the reduced subsets and the n-dimensional features are described by those n RSVM models. Once we have extracted the compressed data as described above, we apply the linear SVM to compressed data for concluding the final MRRSVM model. The detailed training procedure is presented in Algorithm 4. For the MRRSVM prediction phase, it also consists of Map and Reduce steps. In Map-step, each testing data instance is first encoded as an n-dimensional vector using those n RSVM model prediction values distributedly, which are learned from MRRSVM training phase. The testing data prediction is then made by the MRRSVM model (i.e., the Reduce-step). The detailed procedure for testing phase is also presented in Algorithm 5.

To evaluate the performance of our proposed MRRSVM, we test it on three datasets, `adult`, `connect-4` and `shuttle` from UCI machine learning repository (Frank and Asuncion, 2010). Since we focus on binary classification, we transform the latter two datasets from a multi-class problem to a binary classification problem by deleting partial data. For the dataset `connect-4`, it has three classes: “win”, “draw”, and “lose”. We

Algorithm 4 MapReduce RSVM training

Require: The big dataset A.

Ensure: RSVM models $\{model_i\}_{i=1}^n$, the reduced set of each subset $\{\widetilde{A}_i\}$, the linear SSVM model (*final_model*) with new features generated by $\{model_i\}_{i=1}^n$.

0) Split A into n subsets and their associated reduced sets: $\{A_i\}_{i=1}^n$ and $\{\widetilde{A}_i\}_{i=1}^n$.

1) Learn the RSVM $model_i$ for each subset A_i with its reduced set $\widetilde{A}_i$

2) Generate a new representation $B \in \mathbb{R}^{(\sum \widetilde{\ell}_i)\times n}$ (i.e., the Big Table) for the reduced subsets and the jth row of B is represented as follows:
$\mathbf{x}_j^{new} = [model_1(\mathbf{x}_j), model_2(\mathbf{x}_j), \ldots, model_n(\mathbf{x}_j)]^\top \in \mathbb{R}^n$ for $\mathbf{x}_j \in \{\widetilde{A}_i\}_{i=1}^n$

3) Learn the linear SSVM model with B:
final_model $\leftarrow$ *linear_SVM_train(B)*

4) Return $\{model_i\}_{i=1}^n$, $\{\widetilde{A}_i\}_{i=1}^n$, and *final_model*.

delete the class "draw". The dataset size is reduced from 67557 to 61108. We use 45227 instances as the training data and the remainder as testing data. For dataset `shuttle`, it has 7 classes. About 80% of data belong to label "1", and 15% of data belong to label "4". We use these two classes for this evaluation study. The size of training data is reduced from 43500 to 40856, and the size of testing data is reduced from 14500 to 13633. The results are shown in Table 1. From this table, it shows that our proposed MRRSVM can achieve a comparable accuracy performance with nonlinear SVM models, while the computational time can be dramatically reduced. It indicates that MRRSVM has a good potential for large scale problems.

Algorithm 5 MapReduce RSVM prediction

Require: A testing instance $\mathbf{x}_t$, $\{model_i\}_{i=1}^n$, $\{\widetilde{A}_i\}_{i=1}^n$, and *final_model*.

Ensure: Predicted label

0) Generate the new representation for the testing instance $\mathbf{x}_t$:
$\mathbf{x}_t^{new} = [model_1(\mathbf{x}_t), model_2(\mathbf{x}_t), \ldots, model_n(\mathbf{x}_t)]^\top \in \mathbb{R}^n$.

1) Predicted label $\leftarrow$ sign(*final_model*$(\mathbf{x}_t^{new})$).

Table 1: The comparison for linear SSVM, nonlinear SSVM, and MRRSVM for large datasets

Dataset	Method	Training time (CPU sec.)	Accuracy (%)
adult	linear SSVM	0.0704	82.41
	nonlinear SSVM	4.6630	85.14
	MRRSVM	0.6071	84.90
connect-4	linear SSVM	0.2513	65.91
	nonlinear SSVM	8.8672	67.86
	MRRSVM	0.8899	68.14
shuttle	linear SSVM	0.0425	97.25
	nonlinear SSVM	5.3293	99.97
	MRRSVM	0.5374	99.97

6. Conclusion and Discussion

This survey paper gives a comprehensive introduction to the reduced support vector machine, its extensions and applications. We start with the original RSVM algorithm and some statistical theory behind it. While the conventional nonlinear support vector machine has used a full model, RSVM can be viewed as a compressed model for representing the decision function for a given learning task. Generating a compressed model will tremendously cut down the usage of memory and computational time complexity. Thus, we can overcome the computational difficulties in dealing with large scale problems. Moreover, RSVM trims down the model complexity thus it has a gain in better generalization ability. The reduced kernel trick can be interpreted as Nyström low rank approximation of the full kernel matrix combining the step of changing variables. This gives an intuitive justification for RSVM. We also describe three schemes for selecting reduced set. These schemes will generate a smaller reduced set than the random sampling scheme without sacrificing the prediction accuracy. Smaller reduced set will have faster training time. However, there is no free lunch, we have to pay the CPU time in training the reduced set selection. For simplicity, we still suggest the random sampling selection scheme. On the other hand, if a more refined and compact reduced set is preferable, then we should pay extra CPU time and resort to, e.g., the

sampling schemes introduced here. The applications of reduced kernel trick in different learning tasks and algorithms are also included in this survey paper. We finally embed the RSVMs in the MapReduce framework and demonstrate some preliminary numerical tests to show the potential for solving the nonlinear support vector machine with extremely large scale datasets. We believe that the reduced kernel trick embedded in the MapReduce framework can be an important technique in the Big Data era.

References

Alpaydin, E. (2004). *Introduction to Machine Learning*. The MIT Press.

Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena Scientific.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory*, pages 92–100.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1987). Occam's razor. *Information processing letters*, 24(6):377–380.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discover*, 2(2):121–167.

Chang, L.-B., Bai, Z., Huang, S.-Y., and Hwang, C.-R. (2013). Asymptotic error bounds for kernel-based nyström low-rank approximation matrices. *Journal of Multivariate Analysis*, 120:102–119.

Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159.

Chen, C. and Li, K. C. (1998). Can SIR be as popular as multiple linear regression? *Statistica Sinica*, 8:289–316.

Cherkassky, V. and Mulier, F. (1998). *Learning from data: concepts, theory, and methods*. John Wiley and Sons.

Chien, L.-J., Chang, C.-C., and Lee, Y.-J. (2010). Variant methods of reduced set selection for reduced support vector machines. *Journal of Information Science and Engineering*, 26(1):183–196.

Cook, R. D. (1998). *Regression Graphics: Ideas for Studying Regressions Through Graphics*. John Wiley and Sons.

Courant, R. and Hilbert, D. (1953). *Methods of Mathematical Physics*. Interscience Publishers.

Cristianini, N. and Shawe-Taylor, J. (1999). *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository.

Hardoon, D., Szedmak, S., and Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*,16(12): 2639–2664.

Huang, C.-M., Lee, Y.-J., Lin, D. K. J., and Huang, S.-Y. (2007). Model selection for support vector machines via uniform design. *Computational Statistics and Data Analysis*, 52:335–346.

Huang, S.-Y., Lee, M.-H., and Hsiao, C. K. (2009a). Nonlinear measures of association with kernel canonical correlation analysis and applications. *Journal of Statistical Planning and Inference*, 139(7):2162–2174.

Huang, S.-Y., Yeh, Y.-R., and Eguchi, S. (2009b). Robust kernel principal component analysis. *Neural Computation*, 21(11):3179–3213.

Keerthi, S. S. and Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689.

Larsen, J., Svarer, C., Andersen, L. N., and Hansen, L. K. (1998). Adaptive regularization in neural network modeling. *Lecture Notes in Computer Science*, pages 113–132.

Lee, Y.-J., Hsieh, W.-F., and Huang, C.-M. (2005). SSVR: A smooth support vector machine for ϵ-insensitive regression. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):678–685.

Lee, Y.-J. and Huang, S.-Y. (2007). Reduced support vector machines: A statistical theory. *IEEE Transactions on Neural Networks*, 18(1):1–13.

Lee, Y.-J., Lo, H.-Y., and Huang, S.-Y. (2003). Incremental reduced support vector machines. In *Proceedings of International Conference on Informatics, Cybernetics and Systems*.

Lee, Y.-J. and Mangasarian, O. L. (2001). SSVM: A smooth support vector machine. *Computational Optimization and Applications*, 20:5–22.

Li, K. C. (1991). Sliced inverse regression for dimension reduction (with discussion). *Journal of the American Statistical Association*, 86:316–342.

Mangasarian, O. L. (1994). *Nonlinear Programming*. Society for Industrial and Applied Mathematics.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer.

Oyang, Y.-J. and Hwang, S.-C. (2002). An efficient learning algorithm for function approximation with radial basis function networks. In *Proceedings of International Conference on Neural Information Processing*.

Pedrycz, W. and Rai, P. (2008). A multifaceted perspective at data analysis: A study in collaborative intelligent agents. *IEEE Transactions on Systems, Man, and Cybernetics, B: Cybernetics*, 38(4):1062–1072.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

Smola, A. J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings of International Conference on Machine Learning*, pages 911–918.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

Staelin, C. (2003). Parameter selection for support vector machines. *Hewlett-Packard Company, Technical Report HPL-2002-354R1.*

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory.* Springer.

Williams, C. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS).*

Wu, H.-M. (2008). Kernel sliced inverse regression with applications to classification. *Journal of Computational and Graphical Statistics*, 17(3):590–610.

Yeh, Y.-R., Huang, S.-Y., and Lee, Y.-J. (2009). Nonlinear dimension reduction with kernel sliced inverse regression. *IEEE Transactions on Knowledge and Data Engineering*, 21(11):1590–1603.

Zhang, K., Tsang, I. W., and Kwok, J. T. (2008). Improved nyström low-rank approximation and error analysis. In *Proceedings of International Conference on Machine Learning (ICML).*

Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

[Received December 2013; accepted February 2014.]

Journal of the Chinese Statistical Association
Vol. 52, (2014) 85–114

縮減核函數在機器學習的應用與回顧

葉倚任[1] 陳素雲[2] 鮑興國[3] 李育杰[3]

[1]中國文化大學應用數學系
[2]中央研究院統計科學研究所
[3]國立台灣科技大學資訊工程系

摘 要

在此篇文章中我們將介紹縮減支撐向量機 (reduced support vector machine) 的數學模型, 其中包含在統計上的理論基礎、延伸版本以及在機器學習演算法上的應用。主要的內容首先將包含三種不同縮減資料集 (reduced set) 的篩選方法, 在實驗上我們也驗證透過這三種方法所得到較精簡的縮減集亦可以達到非常良好之效果。除了縮減資料集的篩選方法之外, 我們也將介紹應用縮減支撐向量機之概念於機器學習演算法的例子, 其中包含迴歸問題 (regression) 與資料維度縮減問題 (dimension reduction)。最後, 由於所遭遇的問題資料量日趨龐大, 如何利用非線性模型處理如此龐大之資料也是目前許多研究所關心的議題。因此, 我們也將介紹縮減支撐向量機於巨量資料的應用, 其中包含利用縮減支撐向量機之概念於 MapReduce 的架構之中, 進而可以處理巨量資料之問題。

關鍵詞: 巨量資料、維度縮減、核函數、縮減核技巧、監督式學習、支撐向量法。
JEL classification: C1.