SVM Tutorial: Classification, Regression, and Ranking

Hwanjo Yu and Sungchul Kim

1 Introduction

Support Vector Machines(SVMs) have been extensively researched in the data mining and machine learning communities for the last decade and actively applied to applications in various domains. SVMs are typically used for learning classification, regression, or ranking functions, for which they are called classifying SVM, support vector regression (SVR), or ranking SVM (or RankSVM) respectively. Two special properties of SVMs are that SVMs achieve (1) *high generalization* by maximizing the *margin* and (2) support an efficient learning of nonlinear functions by *kernel trick*. This chapter introduces these general concepts and techniques of SVMs for learning classification, regression, and ranking functions. In particular, we first present the SVMs for binary classification in Section 2, SVR in Section 3, ranking SVM in Section 4, and another recently developed method for learning ranking SVM called *Ranking Vector Machine (RVM)* in Section 5.

2 SVM Classification

SVMs were initially developed for classification [5] and have been extended for regression [23] and preference (or rank) learning [14, 27]. The initial form of SVMs is a binary classifier where the output of learned function is either positive or negative. A multiclass classification can be implemented by combining multiple binary classifiers using pairwise coupling method [13, 15]. This section explains the moti-

Hwanjo Yu
POSTECH, Pohang, South Korea, e-mail: hwanjoyu@postech.ac.kr

Sungchul Kim
POSTECH, Pohang, South Korea, e-mail: subright@postech.ac.kr

vation and formalization of SVM as a binary classifier, and the two key properties – margin maximization and kernel trick.

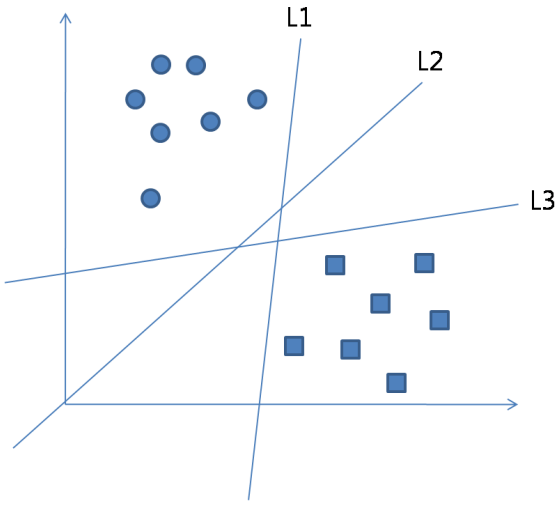


Fig. 1 Linear classifiers (hyperplane) in two-dimensional spaces

Binary SVMs are classifiers which discriminate data points of two categories. Each data object (or data point) is represented by a *n*-dimensional vector. Each of these data points belongs to only one of two classes. A *linear* classifier separates them with an hyperplane. For example, Fig. 1 shows two groups of data and separating hyperplanes that are lines in a two-dimensional space. There are many linear classifiers that correctly classify (or divide) the two groups of data such as L1, L2 and L3 in Fig. 1. In order to achieve maximum separation between the two classes, SVM picks the hyperplane which has the largest margin. The margin is the summation of the shortest distance from the separating hyperplane to the nearest data point of both categories. Such a hyperplane is likely to generalize better, meaning that the hyperplane correctly classify "unseen" or testing data points.

SVMs does the mapping from input space to feature space to support nonlinear classification problems. The kernel trick is helpful for doing this by allowing the absence of the exact formulation of mapping function which could cause the issue of curse of dimensionality. This makes a linear classification in the new space (or the feature space) equivalent to nonlinear classification in the original space (or the input space). SVMs do these by mapping input vectors to a higher dimensional space (or feature space) where a maximal separating hyperplane is constructed.

2.1 Hard-margin SVM Classification

To understand how SVMs compute the hyperplane of maximal margin and support nonlinear classification, we first explain the hard-margin SVM where the training data is free of noise and can be correctly classified by a linear function.

The data points D in Fig. 1 (or training set) can be expressed mathematically as follows.

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\} \tag{1}$$

where $\mathbf{x}_i$ is a n-dimensional real vector, y_i is either 1 or -1 denoting the class to which the point $\mathbf{x}_i$ belongs. The SVM classification function $F(\mathbf{x})$ takes the form

$$F(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - b. \tag{2}$$

$\mathbf{w}$ is the weight vector and b is the bias, which will be computed by SVM in the training process.

First, to correctly classify the training set, $F(\cdot)$ (or $\mathbf{w}$ and b) must return positive numbers for positive data points and negative numbers otherwise, that is, for every point $\mathbf{x}_i$ in D,

$$\mathbf{w} \cdot \mathbf{x}_i - b > 0 \text{ if } y_i = 1, \text{and}$$
$$\mathbf{w} \cdot \mathbf{x}_i - b < 0 \text{ if } y_i = -1$$

These conditions can be revised into:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) > 0, \forall(\mathbf{x}_i, y_i) \in D \tag{3}$$

If there exists such a linear function F that correctly classifies every point in D or satisfies Eq.(3), D is called *linearly separable*.

Second, F (or the hyperplane) needs to maximize the *margin*. Margin is the distance from the hyperplane to the closest data points. An example of such hyperplane is illustrated in Fig. 2. To achieve this, Eq.(3) is revised into the following Eq.(4).

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \forall(\mathbf{x}_i, y_i) \in D \tag{4}$$

Note that Eq.(4) includes equality sign, and the right side becomes 1 instead of 0. If D is linearly separable, or every point in D satisfies Eq.(3), then there exists such a F that satisfies Eq.(4). It is because, if there exist such $\mathbf{w}$ and b that satisfy Eq.(3), they can be always rescaled to satisfy Eq.(4)

The distance from the hyperplane to a vector $\mathbf{x}_i$ is formulated as $\frac{|F(\mathbf{x}_i)|}{||\mathbf{w}||}$. Thus, the margin becomes

$$margin = \frac{1}{||\mathbf{w}||} \tag{5}$$

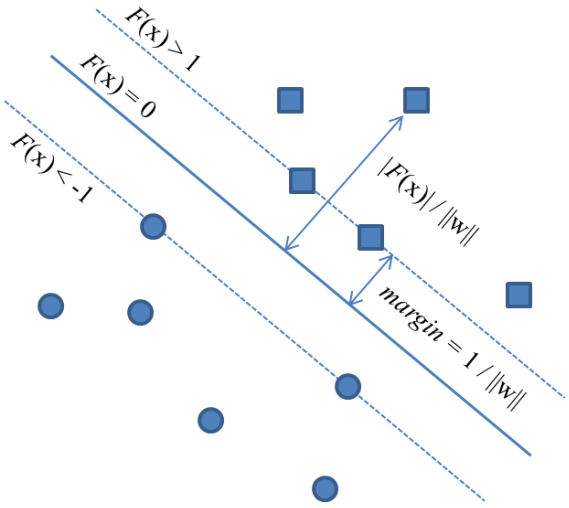


Fig. 2 SVM classification function: the hyperplane maximizing the margin in a two-dimensional space

because when $\mathbf{x}_i$ are the closest vectors, $F(\mathbf{x})$ will return 1 according to Eq.(4). The closest vectors, that satisfy Eq.(4) with equality sign, are called *support vectors*.

Maximizing the margin becomes minimizing $||\mathbf{w}||$. Thus, the training problem in SVM becomes a constrained optimization problem as follows.

$$\text{minimize:} \quad Q(\mathbf{w}) = \tfrac{1}{2}||\mathbf{w}||^2 \tag{6}$$

$$\text{subject to: } y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \geq 1,\ \forall(\mathbf{x}_i, y_i) \in D \tag{7}$$

The factor of $\frac{1}{2}$ is used for mathematical convenience.

2.1.1 Solving the Constrained Optimization Problem

The constrained optimization problem (6) and (7) is called *primal problem*. It is characterized as follows:

- The objective function (6) is a *convex* function of $\mathbf{w}$.
- The constraints are *linear* in $\mathbf{w}$.

Accordingly, we may solve the constrained optimization problem using the method of Largrange multipliers [3]. First, we construct the Largrange function:

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \sum_{i=1}^{m}\alpha_i\{y_i(\mathbf{w}\cdot\mathbf{x}_i - b) - 1\} \tag{8}$$

where the auxiliary nonnegative variables α are called Largrange multipliers. The solution to the constrained optimization problem is determined by the saddle point of the Lagrange function $J(\mathbf{w}, b, \alpha)$, which has to be minimized with respect to $\mathbf{w}$ and b; it also has to be maximized with respect to α. Thus, differentiating $J(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b and setting the results equal to zero, we get the following two conditions of optimality:

$$Condition1 : \frac{\partial J(\mathbf{w},b,\alpha)}{\partial \mathbf{w}} = 0 \tag{9}$$

$$Condition2 : \frac{\partial J(\mathbf{w},b,\alpha)}{\partial b} = 0 \tag{10}$$

After rearrangement of terms, the Condition 1 yields

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i, \mathbf{x}_i \tag{11}$$

and the Condition 2 yields

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{12}$$

The solution vector $\mathbf{w}$ is defined in terms of an expansion that involves the m training examples.

As noted earlier, the primal problem deals with a convex cost function and linear constraints. Given such a constrained optimization problem, it is possible to construct another problem called *dual problem*. The dual problem has the same optimal value as the primal problem, but with the Largrange multipliers providing the optimal solution.

To postulate the dual problem for our primal problem, we first expand Eq.(8), term by term, as follows:

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{w} \cdot \mathbf{x}_i - b \sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i \tag{13}$$

The third term on the right-hand side of Eq.(13) is zero by virtue of the optimality condition of Eq.(12). Furthermore, from Eq.(11) we have

$$\mathbf{w} \cdot \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{14}$$

Accordingly, setting the objective function $J(\mathbf{w}, b, \alpha) = Q(\alpha)$, we can reformulate Eq.(13) as

$$Q(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{15}$$

where the α_i are nonnegative.

We now state the dual problem:

$$\text{maximize: } Q(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{16}$$

$$\text{subject to:} \quad \sum_i \alpha_i y_i = 0 \tag{17}$$

$$\alpha \geq 0 \tag{18}$$

Note that the dual problem is cast entirely in terms of the training data. Moreover, the function $Q(\alpha)$ to be maximized depends only on the input patterns in the form of a set of dot product $\{\mathbf{x}_i \cdot \mathbf{x}_j\}_{(i,j)=1}^{m}$.

Having determined the optimum Lagrange multipliers, denoted by α_i^*, we may compute the optimum weight vector $\mathbf{w}^*$ using Eq.(11) and so write

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x}_i \tag{19}$$

Note that according to the property of Kuhn-Tucker conditions of optimization theory, The solution of the dual problem α_i^* must satisfy the following condition.

$$\alpha_i^* \{y_i(\mathbf{w}^* \cdot \mathbf{x}_i - b) - 1\} = 0 \text{ for } i = 1, 2, ..., m \tag{20}$$

and either α_i^* or its corresponding constraint $\{y_i(\mathbf{w}^* \cdot \mathbf{x}_i - b) - 1\}$ must be nonzero. This condition implies that only when $\mathbf{x}_i$ is a support vector or $y_i(\mathbf{w}^* \cdot \mathbf{x}_i - b) = 1$, its corresponding coefficient α_i will be nonzero (or nonnegative from Eq.(18)). In other words, the $\mathbf{x}_i$ whose corresponding coefficients α_i are zero will not affect the optimum weight vector $\mathbf{w}^*$ due to Eq.(19). Thus, the optimum weight vector $\mathbf{w}^*$ will only depend on the support vectors whose coefficients are nonnegative.

Once we compute the nonnegative α_i^* and their corresponding suppor vectors, we can compute the bias b using a positive support vector $\mathbf{x}_i$ from the following equation.

$$b^* = 1 - \mathbf{w}^* \cdot \mathbf{x}_i \tag{21}$$

The classification of Eq.(2) now becomes as follows.

$$F(\mathbf{x}) = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} - b \tag{22}$$

2.2 Soft-margin SVM Classification

The discussion so far has focused on linearly separable cases. However, the optimization problem (6) and (7) will not have a solution if D is not linearly separable. To deal with such cases, *soft margin* SVM allows mislabeled data points while still maximizing the margin. The method introduces slack variables, ξ_i, which measure

the degree of misclassification. The following is the optimization problem for soft margin SVM.

$$\text{minimize: } Q_1(w,b,\xi_i) = \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \tag{23}$$

$$\text{subject to:} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \geq 1-\xi_i, \qquad \forall(\mathbf{x}_i,y_i)\in D \tag{24}$$

$$\xi_i \geq 0 \tag{25}$$

Due to the ξ_i in Eq.(24), data points are allowed to be misclassified, and the amount of misclassification will be minimized while maximizing the margin according to the objective function (23). C is a parameter that determines the tradeoff between the margin size and the amount of error in training.

Similarily to the case of hard-margin SVM, this primal form can be transformed to the following dual form using the Lagrange multipliers.

$$\text{maximize: } Q_2(\alpha) = \sum_i \alpha_i - \sum_i\sum_j \alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j \tag{26}$$

$$\text{subject to:} \quad \sum_i \alpha_i y_i = 0 \tag{27}$$

$$C \geq \alpha \geq 0 \tag{28}$$

Note that neither the slack variables ξ_i nor their Lagrange multipliers appear in the dual problem. The dual problem for the case of nonseparable patterns is thus similar to that for the simple case of linearly separable patterns except for a minor but important difference. The objective function $Q(\alpha)$ to be maximized is the same in both cases. The nonseparable case differs from the separable case in that the constraint $\alpha_i \geq 0$ is replaced with the more stringent constraint $C \geq \alpha_i \geq 0$. Except for this modification, the constrained optimization for the nonseparable case and computations of the optimum values of the weight vector $\mathbf{w}$ and bias b proceed in the same way as in the linearly separable case.

Just as the hard-margin SVM, α constitute a dual representation for the weight vector such that

$$\mathbf{w}^* = \sum_{i=1}^{m_s} \alpha_i^* y_i \mathbf{x}_i \tag{29}$$

where m_s is the number of support vectors whose corresponding coefficient $\alpha_i > 0$. The determination of the optimum values of the bias also follows a procedure similar to that described before. Once α and b are computed, the function Eq.(22) is used to classify new object.

We can further disclose relationships among α, ξ, and C by the Kuhn-Tucker conditions which are defined by

$$\alpha_i\{y_i(\mathbf{w}\cdot\mathbf{x}_i - b) - 1 + \xi_i\} = 0, \;\; i = 1,2,...,m \tag{30}$$

and

$$\mu_i \xi_i = 0, \quad i = 1, 2, ..., m \tag{31}$$

Eq.(30) is a rewrite of Eq.(20) except for the replacement of the unity term $(1 - \xi_i)$. As for Eq.(31), the μ_i are Lagrange multipliers that have been introduced to enforce the nonnegativity of the slack variables ξ_i for all i. At the saddle point the derivative of the Lagrange function for the primal problem with respect to the slack variable ξ_i is zero, the evaluation of which yields

$$\alpha_i + \mu_i = C \tag{32}$$

By combining Eqs.(31) and (32), we see that

$$\xi_i = 0 \text{ if } \alpha_i < C, \text{ and} \tag{33}$$
$$\xi_i \geq 0 \text{ if } \alpha_i = C \tag{34}$$

We can graphically display the relationships among α_i, ξ_i, and C in Fig. 3.

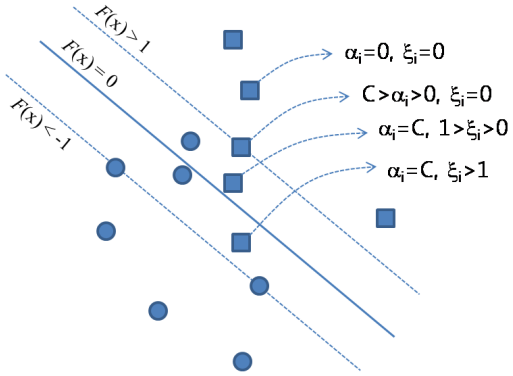


Fig. 3 Graphical relationships among α_i, ξ_i, and C

Data points outside the margin will have $\alpha = 0$ and $\xi = 0$ and those on the margin line will have $C > \alpha > 0$ and still $\xi = 0$. Data points within the margin will have $\alpha = C$. Among them, those correctly classified will have $1 > \xi > 0$ and misclassified points will have $\xi > 1$.

2.3 Kernel Trick for Nonlinear Classification

If the training data is not linearly separable, there is no straight hyperplane that can separate the classes. In order to learn a nonlinear function in that case, linear SVMs must be extended to nonlinear SVMs for the classification of nonlinearly separable

data. The process of finding classification functions using nonlinear SVMs consists of two steps. First, the input vectors are transformed into high-dimensional feature vectors where the training data can be linearly separated. Then, SVMs are used to find the hyperplane of maximal margin in the new feature space. The separating hyperplane becomes a linear function in the transformed feature space but a nonlinear function in the original input space.

Let $\mathbf{x}$ be a vector in the n-dimensional input space and $\varphi(\cdot)$ be a nonlinear mapping function from the input space to the high-dimensional feature space. The hyperplane representing the decision boundary in the feature space is defined as follows.

$$\mathbf{w} \cdot \varphi(\mathbf{x}) - b = 0 \tag{35}$$

where $\mathbf{w}$ denotes a weight vector that can map the training data in the high dimensional feature space to the output space, and b is the bias. Using the $\varphi(\cdot)$ function, the weight becomes

$$\mathbf{w} = \sum \alpha_i y_i \varphi(\mathbf{x}_i) \tag{36}$$

The decision function of Eq.(22) becomes

$$F(\mathbf{x}) = \sum_{i}^{m} \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) - b \tag{37}$$

Furthermore, the dual problem of soft-margin SVM (Eq.(26)) can be rewritten using the mapping function on the data vectors as follows.

$$Q(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) \tag{38}$$

holding the same constraints.

Note that the feature mapping functions in the optimization problem and also in the classifying function always appear as dot products, e.g., $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$. $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$ is the inner product between pairs of vectors in the transformed feature space. Computing the inner product in the transformed feature space seems to be quite complex and suffer from the curse of dimensionality problem. To avoid this problem, the kernel trick is used. The kernel trick replaces the inner product in the feature space with a kernel function K in the original input space as follows.

$$K(\mathbf{u}, \mathbf{v}) = \varphi(\mathbf{u}) \cdot \varphi(\mathbf{v}) \tag{39}$$

The Mercer's theorem proves that a kernel function K is valid, if and only if, the following conditions are satisfied, for any function $\psi(\mathbf{x})$. (Refer to [9] for the proof in detail.)

$$\int K(\mathbf{u}, \mathbf{v}) \psi(\mathbf{u}) \psi(\mathbf{v}) dx dy \leq 0 \tag{40}$$

$$\text{where} \int \psi(x)^2 d\mathbf{x} \leq 0$$

The Mercer's theorem ensures that the kernel function can be always expressed as the inner product between pairs of input vectors in some high-dimensional space, thus the inner product can be calculated using the kernel function only with input vectors in the original space without transforming the input vectors into the high-dimensional feature vectors.

The dual problem is now defined using the kernel function as follows.

$$\text{maximize: } Q_2(\alpha) = \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{41}$$

$$\text{subject to: } \sum_i \alpha_i y_i = 0 \tag{42}$$

$$C \geq \alpha \geq 0 \tag{43}$$

The classification function becomes:

$$F(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) - b \tag{44}$$

Since $K(\cdot)$ is computed in the input space, no feature transformation will be actually done or no $\varphi(\cdot)$ will be computed, and thus the weight vector $\mathbf{w} = \sum \alpha_i y_i \varphi(\mathbf{x})$ will not be computed either in nonlinear SVMs.

The followings are popularly used kernel functions.

- Polynomial: $K(a,b) = (a \cdot b + 1)^d$
- Radial Basis Function (RBF): $K(a,b) = exp(-\gamma||a-b||^2)$
- Sigmoid: $K(a,b) = tanh(\kappa a \cdot b + c)$

Note that, the kernel function is a kind of similarity function between two vectors where the function output is maximized when the two vectors become equivalent. Because of this, SVM can learn a function from any shapes of data beyond vectors (such as trees or graphs) as long as we can compute a similarity function between any pairs of data objects. Further discussions on the properties of these kernel functions are out of the scope. We will instead give an example of using polynomial kernel for learning an XOR function in the following section.

2.3.1 Example: XOR problem

To illustrate the procedure of training a nonlinear SVM function, assume we are given a training set of Table 1.

Figure 4 plots the training points in the 2-D input space. There is no linear function that can separate the training points.

To proceed, let

$$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x} \cdot \mathbf{x}_i)^2 \tag{45}$$

If we denote $\mathbf{x} = (x_1, x_2)$ and $\mathbf{x}_i = (x_{i1}, x_{i2})$, the kernel function is expressed in terms of monomials of various orders as follows.

Input vector **x**	Desired output y
(-1, -1)	-1
(-1, +1)	+1
(+1, -1)	+1
(+1, +1)	-1

Table 1 XOR Problem

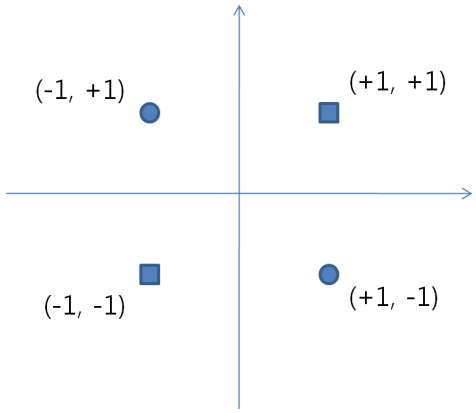


Fig. 4 XOR Problem

$$K(\mathbf{x},\mathbf{x}_i) = 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2} \tag{46}$$

The image of the input vector **x** induced in the feature space is therefore deduced to be

$$\varphi(\mathbf{x}) = (1, x_1^2, \sqrt{2}x_1 x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2) \tag{47}$$

Based on this mapping function, the objective function for the dual form can be derived from Eq. (41) as follows.

$$\begin{aligned} Q(\alpha) = {} & \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\ & -\frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha 1\alpha_4 \\ & +9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3 - 2\alpha 3\alpha_4 + \alpha_4^2) \end{aligned} \tag{48}$$

Optimizing $Q(\alpha)$ with respect to the Lagrange multipliers yields the following set of simultaneous equations:

$$
\begin{aligned}
9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 &= 1 \\
-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 &= 1 \\
-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 &= 1 \\
\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 &= 1
\end{aligned}
$$

Hence, the optimal values of the Lagrange multipliers are

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \tfrac{1}{8}$$

This result denotes that all four input vectors are support vectors and the optimum value of $Q(\alpha)$ is

$$Q(\alpha) = \tfrac{1}{4}$$

and

$$\frac{1}{2}||w||^2 = \frac{1}{4}, \quad \text{or} \quad ||w|| = \frac{1}{\sqrt{2}}$$

From Eq.(36), we find that the optimum weight vector is

$$
\begin{aligned}
w &= \frac{1}{8}[-\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) + \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)] \\
&= \frac{1}{8}\left[-\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} \right] = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \end{bmatrix}
\end{aligned} \tag{49}
$$

The bias b is 0 because the first element of w is 0. The optimal hyperplane becomes

$$w \cdot \varphi(\mathbf{x}) = [0 \quad 0 \quad \frac{-1}{\sqrt{2}} \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}_2 \end{bmatrix} = 0 \tag{50}$$

which reduces to

$$-x_1x_2 = 0 \tag{51}$$

$-x_1x_2 = 0$ is the optimal hyperplane, the solution of the XOR problem. It makes the output $y = 1$ for both input points $x_1 = x_2 = 1$ and $x_1 = x_2 = -1$, and $y = -1$ for both input points $x_1 = 1, x_2 = -1$ or $x_1 = -1, x_2 = 1$. Figure. 5 represents the four points in the transformed feature space.

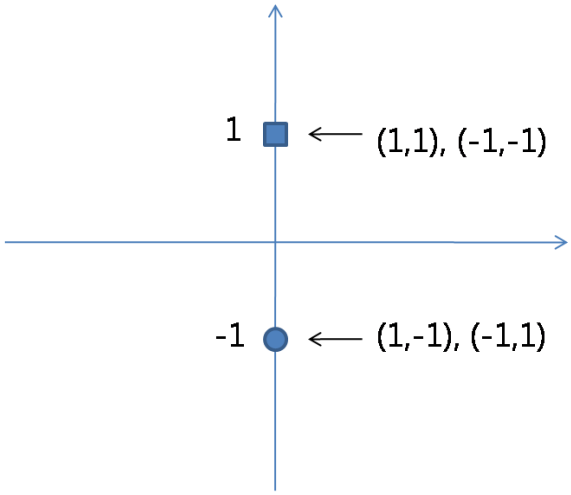


Fig. 5 The 4 data points of XOR problem in the transformed feature space

3 SVM Regression

SVM Regression (SVR) is a method to estimate a function that maps from an input object to a real number based on training data. Similarily to the classifying SVM, SVR has the same properties of the margin maximization and kernel trick for non-linear mapping.

A training set for regression is represented as follows.

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\} \tag{52}$$

where $\mathbf{x}_i$ is a n-dimensional vector, y is the real number for each $\mathbf{x}_i$. The SVR function $F(\mathbf{x}_i)$ makes a mapping from an input vector $\mathbf{x}_i$ to the target y_i and takes the form.

$$F(\mathbf{x}) \Longrightarrow \mathbf{w} \cdot \mathbf{x} - b \tag{53}$$

where $\mathbf{w}$ is the weight vector and b is the bias. The goal is to estimate the parameters ($\mathbf{w}$ and b) of the function that give the best fit of the data. An SVR function $F(\mathbf{x})$

approximates all pairs ($\mathbf{x}_i$, y_i) while maintaining the differences between estimated values and real values under ε precision. That is, for every input vector $\mathbf{x}$ in D,

$$y_i - \mathbf{w} \cdot \mathbf{x_i} - b \leq \varepsilon \tag{54}$$
$$\mathbf{w} \cdot \mathbf{x_i} + b - y_i \leq \varepsilon \tag{55}$$

The margin is

$$margin = \frac{1}{||\mathbf{w}||} \tag{56}$$

By minimizing $||\mathbf{w}||^2$ to maximize the margin, the training in SVR becomes a constrained optimization problem as follows.

$$\text{minimize:} \quad L(\mathbf{w}) = \tfrac{1}{2}||\mathbf{w}||^2 \tag{57}$$
$$\text{subject to: } y_i - \mathbf{w} \cdot \mathbf{x_i} - b \leq \varepsilon \tag{58}$$
$$\mathbf{w} \cdot \mathbf{x_i} + b - y_i \leq \varepsilon \tag{59}$$

The solution of this problem does not allow any errors. To allow some errors to deal with noise in the training data, The soft margin SVR uses slack variables ξ and $\hat{\xi}$. Then, the optimization problem can be revised as follows.

$$\text{minimize: } L(\mathbf{w}, \xi) = \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_i (\xi_i^2, \hat{\xi}_i^2),\ C > 0 \tag{60}$$
$$\text{subject to:} \quad y_i - \mathbf{w} \cdot \mathbf{x_i} - b \leq \varepsilon + \xi_i, \qquad \forall(\mathbf{x}_i, y_i) \in D \tag{61}$$
$$\mathbf{w} \cdot \mathbf{x_i} + b - y_i \leq \varepsilon + \hat{\xi}_i, \qquad \forall(\mathbf{x}_i, y_i) \in D \tag{62}$$
$$\xi, \hat{\xi}_i \geq 0 \tag{63}$$

The constant $C > 0$ is the trade-off parameter between the margin size and the amount of errors. The slack variables ξ and $\hat{\xi}$ deal with infeasible constraints of the optimization problem by imposing the penalty to the excess deviations which are larger than ε.

To solve the optimization problem Eq.(60), we can construct a Lagrange function from the objective function with Lagrange multipliers as follows:

$$\text{minimize: } L = \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_i(\xi_i + \hat{\xi}_i) - \sum_i(\eta_i\xi_i + \hat{\eta}_i\hat{\xi}_i) \tag{64}$$
$$-\sum_i \alpha_i(\varepsilon + \eta_i - y_i + \mathbf{w}\cdot\mathbf{x}_i + b)$$
$$-\sum_i \hat{\alpha}_i(\varepsilon + \hat{\eta}_i + y_i - \mathbf{w}\cdot\mathbf{x}_i - b)$$
$$\text{subject to:} \quad \eta, \hat{\eta}_i \geq 0 \tag{65}$$
$$\alpha, \hat{\alpha}_i \geq 0 \tag{66}$$

where $\eta_i, \hat{\eta}_i, \alpha, \hat{\alpha}_i$ are the Lagrange multipliers which satisfy positive constraints. The following is the process to find the saddle point by using the partial derivatives of L with respect to each lagrangian multipliers for minimizing the function L.

$$\frac{\partial L}{\partial b} = \sum_i(\alpha_i - \hat{\alpha}_i) = 0 \tag{67}$$
$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \Sigma(\alpha_i - \hat{\alpha}_i)\mathbf{x}_i = 0, \mathbf{w} = \sum_i(\alpha_i - \hat{\alpha}_i)\mathbf{x}_i \tag{68}$$
$$\frac{\partial L}{\partial \hat{\xi}_i} = C - \hat{\alpha}_i - \hat{\eta}_i = 0, \hat{\eta}_i = C - \hat{\alpha}_i \tag{69}$$

The optimization problem with inequality constraints can be changed to following dual optimization problem by substituting Eq. (67), (68) and (69) into (64).

$$\text{maximize: } L(\alpha) = \sum_i y_i(\alpha_i - \hat{\alpha}_i) - \varepsilon\sum_i(\alpha_i + \hat{\alpha}_i) \tag{70}$$
$$-\tfrac{1}{2}\sum_i\sum_j(\alpha_i - \hat{\alpha}_i)(\alpha_i - \hat{\alpha}_i)\mathbf{x}_i\mathbf{x}_j \tag{71}$$
$$\text{subject to:} \quad \sum_i(\alpha_i - \hat{\alpha}_i) = 0 \tag{72}$$
$$0 \leq \alpha, \hat{\alpha} \leq C \tag{73}$$

The dual variables $\eta, \hat{\eta}_i$ are eliminated in revising Eq. (64) into Eq. (70). Eq. (68) and (68) can be rewritten as follows.

$$w = \sum_i(\alpha_i - \hat{\alpha}_i)\mathbf{x}_i \tag{74}$$
$$\eta_i = C - \alpha_i \tag{75}$$
$$\hat{\eta}_i = C - \hat{\alpha}_i \tag{76}$$

where w is represented by a linear combination of the training vectors $\mathbf{x}_i$. Accordingly, the SVR function $F(\mathbf{x})$ becomes the following function.

$$F(\mathbf{x}) \Longrightarrow \sum_i(\alpha_i - \hat{\alpha}_i)\mathbf{x}_i\mathbf{x} + b \tag{77}$$

Eq.(77) can map the training vectors to target real values with allowing some errors but it cannot handle the nonlinear SVR case. The same kernel trick can be applied by replacing the inner product of two vectors $\mathbf{x}_i, \mathbf{x}_j$ with a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. The transformed feature space is usually high dimensional, and the SVR function in this space becomes nonlinear in the original input space. Using the kernel function K, The inner product in the transformed feature space can be computed as fast as the inner product $\mathbf{x}_i \cdot \mathbf{x}_j$ in the original input space. The same kernel functions introduced in Section 2.3 can be applied here.

Once replacing the original inner product with a kernel function K, the remaining process for solving the optimization problem is very similar to that for the linear SVR. The linear optimization function can be changed by using kernel function as follows.

$$\text{maximize: } L(\alpha) = \sum_i y_i(\alpha_i - \hat{\alpha}_i) - \varepsilon \sum_i (\alpha_i + \hat{\alpha}_i) - \frac{1}{2} \sum_i \sum_j (\alpha_i - \hat{\alpha}_i)(\alpha_i - \hat{\alpha}_i) K(\mathbf{x}_i, \mathbf{x}_j) \tag{78}$$

$$\text{subject to: } \sum_i (\alpha_i - \hat{\alpha}_i) = 0 \tag{79}$$

$$\hat{\alpha}_i \geq 0, \alpha_i \geq 0 \tag{80}$$

$$0 \leq \alpha, \hat{\alpha} \leq C \tag{81}$$

Finally, the SVR function $F(\mathbf{x})$ becomes the following using the kernel function.

$$F(\mathbf{x}) \Longrightarrow \sum_i (\hat{\alpha}_i - \alpha_i) K(\mathbf{x}_i, \mathbf{x}) + b \tag{82}$$

4 SVM Ranking

Ranking SVM, learning a ranking (or preference) function, has produced various applications in information retrieval [14, 16, 28]. The task of learning *ranking* functions is distinguished from that of learning *classification* functions as follows:

1. While a training set in classification is a set of data objects and their class labels, in *ranking*, a training set is an ordering of data. Let "A is preferred to B" be specified as "$A \succ B$". A training set for ranking SVM is denoted as $R = \{(\mathbf{x}_1, y_i), ..., (\mathbf{x}_m, y_m)\}$ where y_i is the ranking of $\mathbf{x}_i$, that is, $y_i < y_j$ if $\mathbf{x}_i \succ \mathbf{x}_j$.
2. Unlike a classification function, which outputs a distinct class for a data object, a ranking function outputs a *score* for each data object, from which a *global ordering* of data is constructed. That is, the target function $F(\mathbf{x}_i)$ outputs a score such that $F(\mathbf{x}_i) > F(\mathbf{x}_j)$ for any $\mathbf{x}_i \succ \mathbf{x}_j$.

If not stated, R is assumed to be strict ordering, which means that for all pairs $\mathbf{x}_i$ and $\mathbf{x}_j$ in a set D, either $\mathbf{x}_i \succ_R \mathbf{x}_j$ or $\mathbf{x}_i \prec_R \mathbf{x}_j$. However, it can be straightforwardly

generalized to weak orderings. Let R^* be the optimal ranking of data in which the data is ordered perfectly according to user's preference. A ranking function F is typically evaluated by how closely its ordering R^F approximates R^*.

Using the techniques of SVM, a global ranking function F can be learned from an ordering R. For now, assume F is a *linear* ranking function such that:

$$\forall\{(\mathbf{x}_i,\mathbf{x}_j) : y_i < y_j \in R\} : F(\mathbf{x}_i) > F(\mathbf{x}_j) \Longleftrightarrow \mathbf{w}\cdot\mathbf{x}_i > \mathbf{w}\cdot\mathbf{x}_j \tag{83}$$

A weight vector $\mathbf{w}$ is adjusted by a learning algorithm. We say an orderings R is *linearly rankable* if there exists a function F (represented by a weight vector $\mathbf{w}$) that satisfies Eq.(83) for all $\{(\mathbf{x}_i,\mathbf{x}_j) : y_i < y_j \in R\}$.

The goal is to learn F which is concordant with the ordering R and also generalize well beyond R. That is to find the weight vector $\mathbf{w}$ such that $\mathbf{w}\cdot\mathbf{x}_i > \mathbf{w}\cdot\mathbf{x}_j$ for most data pairs $\{(\mathbf{x}_i,\mathbf{x}_j) : y_i < y_j \in R\}$.

Though this problem is known to be NP-hard [10], The solution can be approximated using SVM techniques by introducing (non-negative) slack variables ξ_{ij} and minimizing the upper bound $\sum\xi_{ij}$ as follows [14]:

$$\text{minimize:} \qquad L_1(\mathbf{w},\xi_{ij}) = \tfrac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum\xi_{ij} \tag{84}$$

$$\text{subject to: } \forall\{(\mathbf{x}_i,\mathbf{x}_j) : y_i < y_j \in R\} : \mathbf{w}\cdot\mathbf{x}_i \geq \mathbf{w}\cdot\mathbf{x}_j + 1 - \xi_{ij} \tag{85}$$

$$\forall(i,j) : \xi_{ij} \geq 0 \tag{86}$$

By the constraint (85) and by minimizing the upper bound $\sum\xi_{ij}$ in (84), the above optimization problem satisfies orderings on the training set R with minimal error. By minimizing $\mathbf{w}\cdot\mathbf{w}$ or by maximizing the margin ($= \frac{1}{||\mathbf{w}||}$), it tries to maximize the generalization of the ranking function. We will explain how maximizing the margin corresponds to increasing the generalization of *ranking* in Section 4.1. C is the soft margin parameter that controls the trade-off between the margin size and training error.

By rearranging the constraint (85) as

$$\mathbf{w}(\mathbf{x}_i - \mathbf{x}_j) \geq 1 - \xi_{ij} \tag{87}$$

The optimization problem becomes equivalent to that of *classifying SVM* on *pairwise difference vectors* $(\mathbf{x}_i - \mathbf{x}_j)$. Thus, we can extend an existing SVM implementation to solve the problem.

Note that the support vectors are the data pairs $(\mathbf{x}_i^s,\mathbf{x}_j^s)$ such that constraint (87) is satisfied with the *equality* sign, i.e., $\mathbf{w}(\mathbf{x}_i^s - \mathbf{x}_j^s) = 1 - \xi_{ij}$. Unbounded support vectors are the ones on the margin (i.e., their slack variables $\xi_{ij} = 0$), and bounded support vectors are the ones within the margin (i.e., $1 > \xi_{ij} > 0$) or misranked (i.e., $\xi_{ij} > 1$). As done in the classifying SVM, a function F in ranking SVM is also expressed only by the support vectors.

Similarily to the classifying SVM, the primal problem of ranking SVM can be transformed to the following dual problem using the Lagrange multipliers.

$$\text{maximize: } L_2(\alpha) = \sum_{ij} \alpha_{ij} - \sum_{ij}\sum_{uv} \alpha_{ij}\alpha_{uv} K(\mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_u - \mathbf{x}_v) \tag{88}$$

$$\text{subject to: } C \geq \alpha \geq 0 \tag{89}$$

Once transformed to the dual, the kernel trick can be applied to support nonlinear ranking function. $K(\cdot)$ is a kernel function. α_{ij} is a coefficient for a pairwise difference vectors $(\mathbf{x}_i - \mathbf{x}_j)$. Note that the kernel function is computed for $P^2(\sim m^4)$ times where P is the number of data pairs and m is the number of data points in the training set, thus solving the ranking SVM takes $O(m^4)$ at least. Fast training algorithms for ranking SVM have been proposed [17] but they are limited to linear kernels.

Once α is computed, $\mathbf{w}$ can be written in terms of the pairwise difference vectors and their coefficients such that:

$$\mathbf{w} = \sum_{ij} \alpha_{ij}(\mathbf{x}_i - \mathbf{x}_j) \tag{90}$$

The ranking function F on a new vector $\mathbf{z}$ can be computed using the kernel function replacing the dot product as follows:

$$F(\mathbf{z}) = \mathbf{w} \cdot \mathbf{z} = \sum_{ij} \alpha_{ij}(\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{z} = \sum_{ij} \alpha_{ij} K(\mathbf{x}_i - \mathbf{x}_j, \mathbf{z}). \tag{91}$$

4.1 Margin-Maximization in Ranking SVM

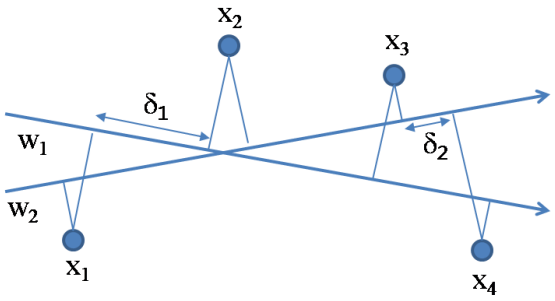


Fig. 6 Linear projection of four data points

We now explain the margin-maximization of the ranking SVM, to reason about how the ranking SVM generates a ranking function of *high generalization*. We first establish some essential properties of ranking SVM. For convenience of explana-

tion, we assume a training set R is linearly rankable and thus we use hard-margin SVM, i.e., $\xi_{ij} = 0$ for all (i, j) in the objective (84) and the constraints (85).

In our ranking formulation, from Eq.(83), the linear ranking function $F_{\mathbf{w}}$ projects data vectors onto a weight vector $\mathbf{w}$. For instance, Fig. 6 illustrates linear projections of four vectors $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ onto two different weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ respectively in a two-dimensional space. Both $F_{\mathbf{x}_1}$ and $F_{\mathbf{x}_2}$ make the same ordering R for the four vectors, that is, $\mathbf{x}_1 >_R \mathbf{x}_2 >_R \mathbf{x}_3 >_R \mathbf{x}_4$. The ranking difference of two vectors $(\mathbf{x}_i, \mathbf{x}_j)$ according to a ranking function $F_{\mathbf{w}}$ is denoted by the geometric distance of the two vectors projected onto $\mathbf{w}$, that is, formulated as $\frac{\mathbf{w}(\mathbf{x}_i - \mathbf{x}_j)}{||\mathbf{w}||}$.

Corollary 1. *Suppose $F_{\mathbf{w}}$ is a ranking function computed by the hard-margin ranking SVM on an ordering R. Then, the support vectors of $F_{\mathbf{w}}$ represent the data pairs that are closest to each other when projected to* $\mathbf{w}$ *thus closest in ranking.*

Proof. The support vectors are the data pairs $(\mathbf{x}_i^s, \mathbf{x}_j^s)$ such that $\mathbf{w}(\mathbf{x}_i^s - \mathbf{x}_j^s) = 1$ in constraint (87), which is the smallest possible value for all data pairs $\forall(\mathbf{x}_i, \mathbf{x}_j) \in R$. Thus, its ranking difference according to $F_{\mathbf{w}}(= \frac{\mathbf{w}(\mathbf{x}_i^s - \mathbf{x}_j^s)}{||\mathbf{w}||})$ is also the smallest among them [24].

Corollary 2. *The ranking function F, generated by the hard-margin ranking SVM, maximizes the minimal difference of any data pairs in ranking.*

Proof. By minimizing $\mathbf{w} \cdot \mathbf{w}$, the ranking SVM maximizes the margin $\delta = \frac{1}{||\mathbf{w}||} = \frac{\mathbf{w}(\mathbf{x}_i^s - \mathbf{x}_j^s)}{||\mathbf{w}||}$ where $(\mathbf{x}_i^s, \mathbf{x}_j^s)$ are the support vectors, which denotes, from the proof of Corollary 1, the minimal difference of any data pairs in ranking.

The soft margin SVM allows bounded support vectors whose $\xi_{ij} > 0$ as well as unbounded support vectors whose $\xi_{ij} = 0$, in order to deal with noise and allow small error for the R that is not completely linearly rankable. However, the objective function in (84) also minimizes the amount of the slacks and thus the amount of error, and the support vectors are the close data pairs in ranking. Thus, maximizing the margin generates the effect of maximizing the differences of close data pairs in ranking.

From Corollary 1 and 2, we observe that ranking SVM improves the generalization performance by maximizing the minimal ranking difference. For example, consider the two linear ranking functions $F_{\mathbf{w}_1}$ and $F_{\mathbf{w}_2}$ in Fig. 6. Although the two weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ make the same ordering, intuitively $\mathbf{w}_1$ generalizes better than $\mathbf{w}_2$ because the distance between the closest vectors on $\mathbf{w}_1$ (i.e., δ_1) is larger than that on $\mathbf{w}_2$ (i.e., δ_2). SVM computes the weight vector $\mathbf{w}$ that maximizes the differences of close data pairs in ranking. Ranking SVMs find a ranking function of high generalization in this way.

5 Ranking Vector Machine: An Efficient Method for Learning the 1-norm Ranking SVM

This section presents another rank learning method, *Ranking Vector Machine (RVM)*, a revised 1-norm ranking SVM that is better for feature selectoin and more scalable to large data sets than the standard ranking SVM.

We first develop a 1-norm ranking SVM, a ranking SVM that is based on *1-norm* objective function. (The standard ranking SVM is based on 2-norm objective function.) The 1-norm ranking SVM learns a function with much less support vectors than the standard SVM. Thereby, its testing time is much faster than 2-norm SVMs and provides better feature selection properties. (The function of 1-norm SVM is likely to utilize a less number of features by using a less number of support vectors [11].) Feature selection is also important in ranking. Ranking functions are relevance or preference functions in document or data retrieval. Identifying key features increases the interpretability of the function. Feature selection for nonlinear kernel is especially challenging, and the fewer the number of support vectors are, the more efficiently feature selection can be done [12, 20, 6, 30, 8].

We next present RVM which revises the 1-norm ranking SVM for fast training. The RVM trains much faster than standard SVMs while not compromising the accuracy when the training set is relatively large. *The key idea of RVM is to express the ranking function with "ranking vectors" instead of support vectors.* Support vectors in ranking SVMs are pairwise difference vectors of the closest pairs as discussed in Section 4. Thus, the training requires investigating every *data pair* as potential candidates of support vectors, and the number of data pairs are quadratic to the size of training set. On the other hand, the ranking function of the RVM utilizes each training data object instead of data pairs. Thus, the number of variables for optimization is substantially reduced in the RVM.

5.1 1-norm Ranking SVM

The goal of 1-norm ranking SVM is the same as that of the standard ranking SVM, that is, to learn F that satisfies Eq.(83) for most $\{(\mathbf{x}_i, \mathbf{x}_j) : y_i < y_j \in R\}$ and generalize well beyond the training set. In the 1-norm ranking SVM, we express Eq.(83) using the F of Eq.(91) as follows.

$$F(\mathbf{x}_u) > F(\mathbf{x}_v) \Longrightarrow \sum_{ij}^{P} \alpha_{ij}(\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{x}_u > \sum_{ij}^{P} \alpha_{ij}(\mathbf{x}_i - \mathbf{x}_j) \cdot \mathbf{x}_v \tag{92}$$

$$\Longrightarrow \sum_{ij}^{P} \alpha_{ij}(\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{x}_u - \mathbf{x}_v) > 0 \tag{93}$$

Then, replacing the inner product with a kernel function, the 1-norm ranking SVM is formulated as:

$$minimize: \qquad L(\alpha,\xi) = \sum_{ij}^{P} \alpha_{ij} + C \sum_{ij}^{P} \xi_{ij} \tag{94}$$

$$s.t.: \sum_{ij}^{P} \alpha_{ij} K(\mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_u - \mathbf{x}_v) \geq 1 - \xi_{uv}, \; \forall \{(u,v) : y_u < y_v \in R\} \tag{95}$$

$$\alpha \geq 0, \; \xi \geq 0 \tag{96}$$

While the standard ranking SVM suppresses the weight $\mathbf{w}$ to improve the generalization performance, the 1-norm ranking suppresses α in the objective function. Since the weight is expressed by the sum of the coefficient times pairwise ranking difference vectors, suppressing the coefficient α corresponds to suppressing the weight $\mathbf{w}$ in the standard SVM. (Mangasarian proves it in [18].) C is a user parameter controlling the tradeoff between the margin size and the amount of error, ξ, and K is the kernel function. P is the number of pairwise difference vectors ($\sim m^2$).

The training of the 1-norm ranking SVM becomes a linear programming (LP) problem thus solvable by LP algorithms such as the Simplex and Interior Point method [18, 11, 19]. Just as the standard ranking SVM, K needs to be computed P^2 ($\sim m^4$) times, and there are P number of constraints (95) and α to compute. Once α is computed, F is computed using the same ranking function as the standard ranking SVM, i.e., Eq.(91).

The accuracies of 1-norm ranking SVM and standard ranking SVM are comparable, and both methods need to compute the kernel function $O(m^4)$ times. In practice, the training of the standard SVM is more efficient because fast decomposition algorithms have been developed such as sequential minimal optimization (SMO) [21] while the 1-norm ranking SVM uses common LP solvers.

It is shown that *1-norm SVMs use much less support vectors that standard 2-norm SVMs*, that is, the number of positive coefficients (i.e., $\alpha > 0$) after training is much less in the 1-norm SVMs than in the standard 2-norm SVMs [19, 11]. It is because, unlike the standard 2-norm SVM, the support vectors in the 1-norm SVM are not bounded to those close to the boundary in classification or the minimal ranking difference vectors in ranking. Thus, the testing involves much less kernel evaluations, and it is more robust when the training set contains noisy features [31].

5.2 Ranking Vector Machine

Although the 1-norm ranking SVM has merits over the standard ranking SVM in terms of the testing efficiency and feature selection, its training complexity is very high w.r.t. the number of data points. In this section, we present Ranking Vector Machine (RVM), which revises the 1-norm ranking SVM to reduce the training time substantially. The RVM significantly reduces the number of variables in the optimization problem while not compromizing the accuracy. *The key idea of RVM is to express the ranking function with "ranking vectors" instead of support vectors.*

The support vectors in ranking SVMs are chosen from pairwise difference vectors, and the number of pairwise difference vectors are quadratic to the size of training set. On the other hand, the ranking vectors are chosen from the training vectors, thus the number of variables to optimize is substantially reduced.

To theoretically justify this approach, we first present the Representer Theorem.

Theorem 1 (Representer Theorem [22]).

Denote by Ω: $[0,\infty) \rightarrow \mathscr{R}$ a strictly monotonic increasing function, by $\mathscr{X}$ a set, and by $c : (\mathscr{X} \times \mathscr{R}^2)^m \rightarrow \mathscr{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $F \in \mathscr{H}$ of the regularized risk

$$c((x_1,y_1,F(x_1)),...,(x_m,y_m,F(x_m))) + \Omega(||F||_{\mathscr{H}}) \tag{97}$$

admits a representation of the form

$$F(x) = \sum_{i=1}^{m} \alpha_i K(x_i,x) \tag{98}$$

The proof of the theorem is presented in [22].

Note that, in the theorem, the loss function c is *arbitrary* allowing *coupling* between data points $(\mathbf{x}_i, y_i)$, and the regularizer Ω has to be monotonic.

Given such a loss function and regularizer, the representer theorem states that although we might be trying to solve the optimization problem in an infinite-dimensional space $\mathscr{H}$, containing linear combinations of kernels centered on *arbitrary* points of $\mathscr{X}$, the solution lies in the span of m particular kernels – those centered on the *training* points [22].

Based on the theorem, we define our ranking function F as Eq.(98), which is based on the *training points* rather than arbitrary points (or pairwise difference vectors). Function (98) is similar to function (91) except that, unlike the latter using pairwise difference vectors $(\mathbf{x}_i - \mathbf{x}_j)$ and their coeifficients (α_{ij}), the former utilizes the training vectors $(\mathbf{x}_i)$ and their coefficients (α_i). With this function, Eq.(92) becomes the following.

$$F(\mathbf{x}_u) > F(\mathbf{x}_v) \Longrightarrow \sum_{i}^{m} \alpha_i K(\mathbf{x}_i,\mathbf{x}_u) > \sum_{i}^{m} \alpha_i K(\mathbf{x}_i,\mathbf{x}_v) \tag{99}$$

$$\Longrightarrow \sum_{i}^{m} \alpha_i (K(\mathbf{x}_i,\mathbf{x}_u) - K(\mathbf{x}_i,\mathbf{x}_v)) > 0. \tag{100}$$

Thus, we set our loss function c as follows.

$$c = \sum_{\forall\{(u,v):y_u<y_v \in R\}} (1 - \sum_{i}^{m} \alpha_i (K(\mathbf{x}_i,\mathbf{x}_u) - K(\mathbf{x}_i,\mathbf{x}_v))) \tag{101}$$

The loss function utilizes couples of data points penalizing misranked pairs, that is, it returns higher values as the number of misranked pairs increases. Thus, the loss function is order sensitive, and it is an instance of the function class c in Eq.(97).

We set the regularizer $\Omega(||f||_{\mathscr{H}}) = \sum_i^m \alpha_i$ ($\alpha_i \geq 0$), which is strictly monotonically increasing. Let P is the number of pairs $(u,v) \in R$ such that $y_u < y_v$, and let $\xi_{uv} = 1 - \sum_i^m \alpha_i (K(\mathbf{x}_i, \mathbf{x}_u) - K(\mathbf{x}_i, \mathbf{x}_v))$. Then, our RVM is formulated as follows.

$$\text{minimize:} \qquad L(\alpha, \xi) = \sum_i^m \alpha_i + C \sum_{ij}^P \xi_{ij} \tag{102}$$

$$\text{s.t.:} \sum_i^m \alpha_i (K(\mathbf{x}_i, \mathbf{x}_u) - K(\mathbf{x}_i, \mathbf{x}_v)) \geq 1 - \xi_{uv}, \forall \{(u,v) : y_u < y_v \in R\} \tag{103}$$

$$\alpha, \xi \geq 0 \tag{104}$$

The solution of the optimization problem lies in the span of kernels centered on the training points (i.e., Eq.(98)) as suggested in the representer theorem. Just as the 1-norm ranking SVM, the RVM suppresses α to improve the generalization, and forces Eq.(100) by constraint (103). Note that there are only m number of α_i in the RVM. Thus, the kernel function is evaluated $O(m^3)$ times while the standard ranking SVM computes it $O(m^4)$ times.

Another rationale of RVM or rationale of using training vectors instead of pairwise difference vectors in the ranking function is that the support vectors in the 1-norm ranking SVM are not the closest pairwise difference vectors, thus expressing the ranking function with pairwise difference vectors becomes not as beneficial in the 1-norm ranking SVM. To explain this further, consider *classifying* SVMs. Unlike the 2-norm (classifying) SVM, the support vectors in the 1-norm (classifying) SVM are *not* limited to those close to the decision boundary. This makes it possible that the 1-norm (classifying) SVM can express the similar boundary function with less number of support vectors. Directly extended from the 2-norm (classifying) SVM, the 2-norm *ranking* SVM improves the generalization by maximizing the closest pairwise ranking difference that corresponds to the margin in the 2-norm (classifying) SVM as discussed in Section 4. Thus, the 2-norm ranking SVM expresses the function with the closest pairwise difference vectors (i.e., the support vectors). However, the 1-norm ranking SVM improves the generalization by suppressing the coefficients α just as the 1-norm (classifying) SVM. Thus, the support vectors in the 1-norm ranking SVM are not the closest pairwise difference vectors any more, and thus expressing the ranking function with pairwise difference vectors becomes not as beneficial in the 1-norm ranking SVM.

5.3 Experiment

This section evaluates the RVM on synthetic datasets (Section 5.3.1) and a real-world dataset (Secton 5.3.2). The RVM is compared with the state-of-the-art ranking SVM provided in SVM-light. Experiment results show that the RVM trains substantially faster than the SVM-light for nonlinear kernels while their accura-

cies are comparable. More importantly, the number of ranking vectors in the RVM is multiple orders of magnitudes smaller than the number of support vectors in the SVM-light. Experiments are performed on a Windows XP Professional machine with a Pentium IV 2.8GHz and 1GB of RAM. We implemented the RVM using C and used CPLEX [1] for the LP solver. The source codes are freely available at "http://iis.postech.ac.kr/rvm" [29].

Evaluation metric: MAP (mean average precision) is used to measure the ranking quality when there are only two classes of ranking [26], and NDCG is used to evaluate ranking performance for IR applications when there are multiple levels of ranking [2, 4, 7, 25]. Kendall's τ is used when there is a global ordering of data and the training data is a subset of it. Ranking SVMs as well as the RVM minimize the amount of error or mis-ranking, which is corresponding to optimizing the Kendall's τ [16, 27]. Thus, we use the Kendall's τ to compare their accuracy.

Kendall's τ computes the overall accuracy by comparing the similarity of two orderings R^* and R^F. (R^F is the ordering of D according to the learned function F.) The Kendall's τ is defined based on the number of concordant pairs and discordant pairs. If R^* and R^F agree in how they order a pair, $\mathbf{x}_i$ and $\mathbf{x}_j$, the pair is concordant, otherwise, it is discordant. The accuracy of function F is defined as the number of concordant pairs between R^* and R^F per the total number of pairs in D as follows.

$$F(R^*, R^F) = \frac{\text{\# of concordant pairs}}{\binom{|R|}{2}}$$

For example, suppose R^* and R^F order five points $\mathbf{x}_1, \ldots, \mathbf{x}_5$ as follow:

$$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5) \in R^*$$
$$(\mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5) \in R^F$$

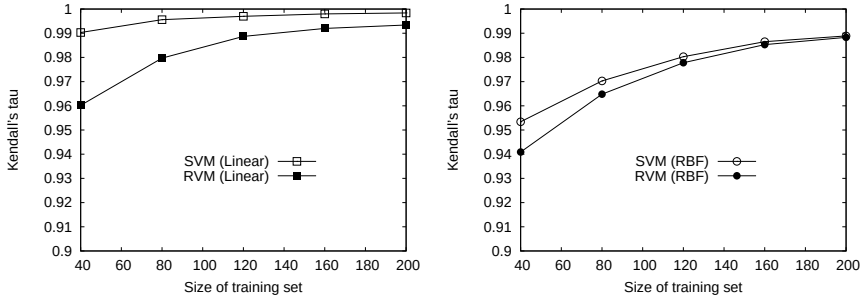
Then, the accuracy of F is 0.7, as the number of discordant pairs is 3, i.e.,$\{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_1, \mathbf{x}_3\}, \{\mathbf{x}_2, \mathbf{x}_3\}$ while all remaining 7 pairs are concordant.

5.3.1 Experiments on Synthetic Dataset

Below is the description of our experiments on synthetic datasets.

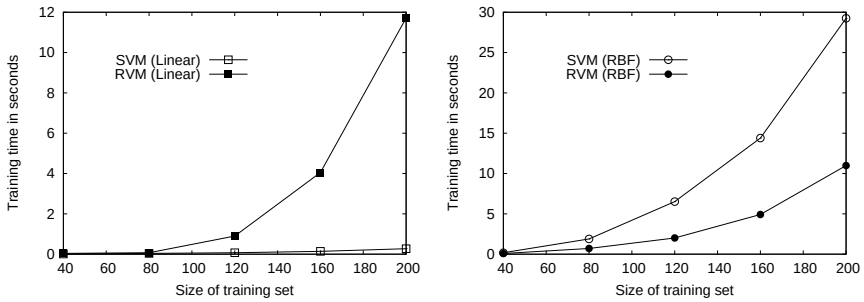
1. We randomly generated a training and a testing dataset D_{train} and D_{test} respectively, where D_{train} contains m_{train} (= 40, 80, 120, 160, 200) data points of n (e.g., 5) dimensions (i.e., m_{train}-by-n matrix), and D_{test} contains m_{test} (= 50) data points of n dimensions (i.e., m_{test}-by-n matrix). Each element in the matrices is a random number between zero and one. (We only did experiments on the data set

[1] http://www.ilog.com/products/cplex/



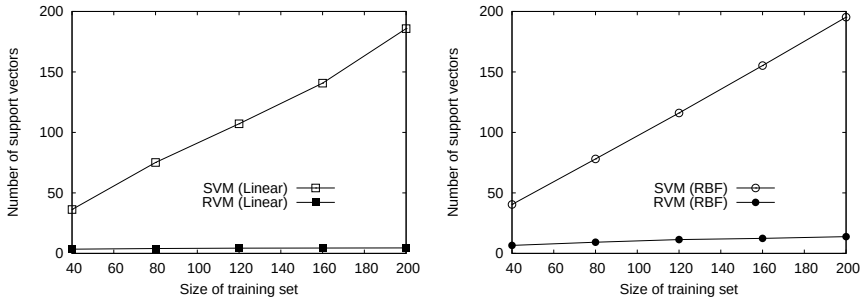
(a) Linear (b) RBF

Fig. 7 Accuracy



(a) Linear Kernel (b) RBF Kernel

Fig. 8 Training time

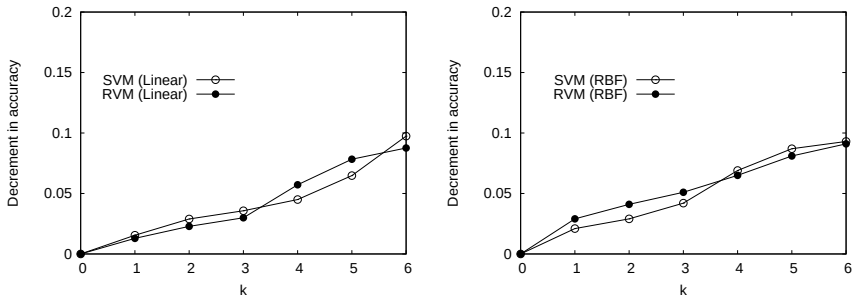


(a) Linear Kernel (b) RBF Kernel

Fig. 9 Number of support (or ranking) vectors

of up to 200 objects due to performance reason. Ranking SVMs run intolerably slow on data sets larger than 200.)

2. We randomly generate a global ranking function F^*, by randomly generating the weight vector $\mathbf{w}$ in $F^*(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ for linear, and in $F^*(\mathbf{x}) = exp(-||\mathbf{w} - \mathbf{x}||)^2$ for RBF function.



(a) Linear (b) RBF

Fig. 10 Sensitivity to noise ($m_{train} = 100$).

3. We rank D_{train} and D_{test} according to F^*, which forms the global ordering R^*_{train} and R^*_{test} on the training and testing data.
4. We train a function F from R^*_{train}, and test the accuracy of F on R^*_{test}.

We tuned the soft margin parameter C by trying $C = 10^{-5}$, $10^{-5}, ..., 10^5$, and used the highest accuracy for comparison. For the linear and RBF functions, we used linear and RBF kernels accordingly. We repeat this entire process 30 times to get the mean accuracy.

Accuracy: Figure 7 compares the accuracies of the RVM and the ranking SVM from the SVM-light. The ranking SVM outperforms RVM when the size of data set is small, but their difference becomes trivial as the size of data set increases. This phenomenon can be explained by the fact that when the training size is too small, the number of potential ranking vectors becomes too small to draw an accurate ranking function whereas the number of potential support vectors is still large. However, as the size of training set increases, RVM becomes as accurate as the ranking SVM because the number of potential ranking vectors becomes large as well.

Training Time: Figure 8 compares the training time of the RVM and the SVM-light. While the SVM light trains much faster than RVM for linear kernel (SVM light is specially optimized for linear kernel.), *the RVM trains significantly faster than the SVM light for RBF kernel.*

Number of Support (or Ranking) Vectors: Figure 9 compares the number of support (or ranking) vectors used in the function of RVM and the SVM-light. RVM's model uses a significantly smaller number of support vectors than the SVM-light.

Sensitivity to noise: In this experiment, we compare the sensitivity of each method to noise. We insert noise by switching the orders of some data pairs in R^*_{train}. We set the size of training set $m_{train} = 100$ and the dimension $n = 5$. After we make R^*_{train} from a random function F^*, we randomly picked k vectors from the R^*_{train} and switched it with its adjacent vector in the ordering to implant noise in the training set. Figure 10 shows the decrements of the accuracies as the number of misorderings

increases in the training set. Their accuracies are moderately decreasing as the noise increases in the training set, and their sensitivities to noise are comparable.

5.3.2 Experiment on Real Dataset

In this section, we experiment using the OHSUMED dataset obtained from the LETOR, the site containing benchmark datasets for ranking [1]. OHSUMED is a collection of documents and queries on medicine, consisting of 348,566 references and 106 queries. There are in total 16,140 query-document pairs upon which relevance judgements are made. In this dataset the relevance judgements have three levels: "definitely relevant", "partially relevant", and "irrelevant". The OHSUMED dataset in the LETOR extracts 25 features. We report our experiments on the first three queries and their documents. We compare the performance of RVM and SVM-light on them. We tuned the parameters 3-fold cross validation with trying C and $\gamma = 10^{-6}, 10^{-5}, ..., 10^{6}$ for the linear and RBF kernels and compared the highest performance. The training time is measured for training the model with the tuned parameters. We repeated the whole process three times and reported the mean values.

		query 1 $\|D\| = 134$			query 2 $\|D\| = 128$			query 3 $\|D\| = 182$		
		Acc	Time	#SV or #RV	Acc	Time	#SV or #RV	Acc	Time	#SV or #RV
RVM	linear	**.5484**	**.23**	**1.4**	**.6730**	**.41**	**3.83**	.6611	1.94	1.99
	RBF	.5055	.85	4.3	.6637	.41	2.83	**.6723**	**4.71**	**1**
SVM	linear	**.5634**	**1.83**	**92**	.6723	1.03	101.66	.6588	4.24	156.55
	RBF	.5490	3.05	92	**.6762**	**3.50**	**102**	**.6710**	**55.08**	**156.66**

Table 2 Experiment results: Accuracy (Acc), Training Time (Time), and Number of Support or Ranking Vectors (#SV or #RV)

Table 5.3.2 show the results. The accuracies of the SVM and RVM are comparable overall; SVM shows a little high accuracy than RVM for query 1, but for the other queries, their accuracy differences are not statistically significant. More importantly, *the number of ranking vectors in RVM is significantly smaller than that of support vectors in SVM*. For example, for query 3, the RVM having just one ranking vector outperformed the SVM with over 150 support vectors. *The training time of RVM is significantly shorter than that of SVM-light.*

References

1. Letor: Learning to rank for information retrieval. Http://research.microsoft.com/users/LETOR/
2. Baeza-Yates, R., Ribeiro-Neto, B. (eds.): Modern Information Retrieval. ACM Press (1999)
3. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1995)

4. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proc. Int. Conf. Machine Learning (ICML'04) (2004)
5. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2**, 121–167 (1998)
6. Cao, B., Shen, D., Sun, J.T., Yang, Q., Chen, Z.: Feature selection in a kernel space. In: Proc. Int. Conf. Machine Learning (ICML'07) (2007)
7. Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., Hon, H.W.: Adapting ranking svm to document retrieval. In: Proc. ACM SIGIR Int. Conf. Information Retrieval (SIGIR'06) (2006)
8. Cho, B., Yu, H., Lee, J., Chee, Y., Kim, I.: Nonlinear support vector machine visualization for risk factor analysis using nomograms and localized radial basis function kernels. IEEE Transactions on Information Technology in Biomedicine ((Accepted))
9. Christianini, N., Shawe-Taylor, J.: An Introduction to support vector machines and other kernel-based learning methods. Cambridge University Press (2000)
10. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. In: Proc. Advances in Neural Information Processing Systems (NIPS'98) (1998)
11. Fung, G., Mangasarian, O.L.: A feature selection newton method for support vector machine classification. Computational Optimization and Applications (2004)
12. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research (2003)
13. Hastie, T., Tibshirani, R.: Classification by pairwise coupling. In: Advances in Neural Information Processing Systems (1998)
14. Herbrich, R., Graepel, T., Obermayer, K. (eds.): Large margin rank boundaries for ordinal regression. MIT-Press (2000)
15. J.H.Friedman: Another approach to polychotomous classification. Tech. rep., Standford University, Department of Statistics, 10:1895-1924 (1998)
16. Joachims, T.: Optimizing search engines using clickthrough data. In: Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'02) (2002)
17. Joachims, T.: Training linear svms in linear time. In: Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'06) (2006)
18. Mangasarian, O.L.: Generalized Support Vector Machines. MIT Press (2000)
19. Mangasarian, O.L.: Exact 1-norm support vector machines via unconstrained convex differentiable minimization. Journal of Machine Learning Research (2006)
20. Mangasarian, O.L., Wild, E.W.: Feature selection for nonlinear kernel support vector machines. Tech. rep., University of Wisconsin, Madison (1998)
21. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: A.S. B. Scholkopf C. Burges (ed.) Advances in Kernel Methods: Support Vector Machines. MIT Press, Cambridge, MA (1998)
22. Scholkopf, B., Herbrich, R., Smola, A.J., Williamson, R.C.: A generalized representer theorem. In: Proc. COLT (2001)
23. Smola, A.J., Scholkopf, B.: A tutorial on support vector regression. Tech. rep., NeuroCOLT2 Technical Report NC2-TR-1998-030 (1998)
24. Vapnik, V.: Statistical Learning Theory. John Wiley and Sons (1998)
25. Xu, J., Li, H.: Adarank: A boosting algorithm for information retrieval. In: Proc. ACM SIGIR Int. Conf. Information Retrieval (SIGIR'07) (2007)
26. Yan, L., Dodier, R., Mozer, M.C., Wolniewicz, R.: Optimizing classifier performance via the wilcoxon-mann-whitney statistics. In: Proc. Int. Conf. Machine Learning (ICML'03) (2003)
27. Yu, H.: SVM selective sampling for ranking with application to data retrieval. In: Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'05) (2005)
28. Yu, H., Hwang, S.W., Chang, K.C.C.: Enabling soft queries for data retrieval. Information Systems (2007)
29. Yu, H., Kim, Y., Hwang, S.W.: RVM: An efficient method for learning ranking SVM. Tech. rep., Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea, http://iis.hwanjoyu.org/rvm (2008)

30. Yu, H., Yang, J., Wang, W., Han, J.: Discovering compact and highly discriminative features or feature combinations of drug activities using support vector machines. In: IEEE Computer Society Bioinformatics Conf. (CSB'03), pp. 220–228 (2003)
31. Zhu, J., Rosset, S., Hastie, T., Tibshriani, R.: 1-norm support vector machines. In: Proc. Advances in Neural Information Processing Systems (NIPS'00) (2003)

Index