A Tutorial for Support Vector Machine

Wei-Lun Chao
weilunchao760414@gmail.com
Graduate Institute of Communication Engineering, National Taiwan University
Draft version: Dec. 30, 2011

Abstract

Support vector machine (SVM), proposed by V. Vapnik in mid 1990, is probably the most popular machine learning algorithm in the last decade. SVM has a solid theoretical background, an intuitive geometrical interpretation, and several interesting properties that link the development of kernel space and convex optimization. At the early age of SVM, it suffered from the computational complexity, coming from a large-scale quadratic programming problem, and therefore seemed to be an ideal tool. This problem brought many researchers in, and several efficient algorithms and fast SVM packages, such as SVM^{light} [1] and LIBSVM [2], have been developed and released since the late 1990. Thanks to these efforts, SVM has now been widely applied and even served as the baseline in computer vision, pattern recognition, information retrieval, and data mining, etc.

In this tutorial, we will provide the basic model and ideas of SVM, and then derive its variational forms such as the dual form and kernel extension. This tutorial is organized as follows:

Section 1: The geometrical interpretation of SVM
Section 2: The optimization form
Section 3: The dual form
Section 4: The soft-margin SVM
Section 5: The kernel extension of SVM
Section 6: Key points of SVM
Section 7: Practical package: LIBSVM
Section 8: Conclusion

1. The Geometrical Interpretation of SVM

SVM is originally a deterministic algorithm to find the *linear separating hyperplane* of a *binary* labeled dataset. Compared to the perceptron learning algorithm (PLA), which eventually finds a random separating hyperplane in a linear-separable dataset, SVM outputs a unique hyperplane in a given dataset. This hyperplane provided by SVM not only has an intuitive geometrical interpretation, but also been proved to balance the in-sample error (E_{in}) and the generalization error theoretically. In this Section, we will discuss the geometrical interpretation of SVM.

Given a binary labeled dataset as shown in Fig. 1, we found that there are many hyperplanes to separate the red (positive: 1) and blue (negative: -1) circles, such as the three gray lines plotted in Fig. 2 (a). These gray lines may come from the PLA, the Adaline algorithm, the least square regression algorithm, or the logistic regression algorithm, where the last three determine their separating hyperplane based on the corresponding objective functions but without a direct geometrical interpretation. SVM, on contrary, was originated from a geometrical view as shown in Fig. 2 (b), which seeks a separating hyperplane that stands as far as possible to the positive and negative samples without training errors. In other words, SVM desire a separating hyperplane that can maximize the *margin* between the positive and the negative samples. This interpretation can effectively tolerate the error of the unseen samples, and was claimed to have good generalization ability. The objective function of SVM is then derived from this interpretation and modeled as a constraint optimization problem.

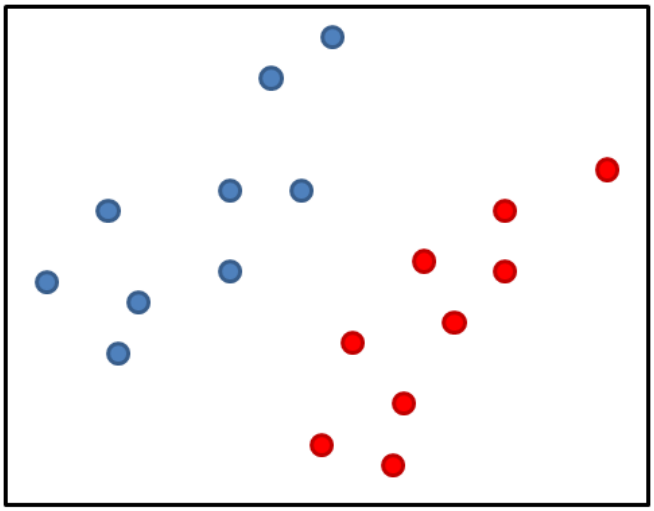
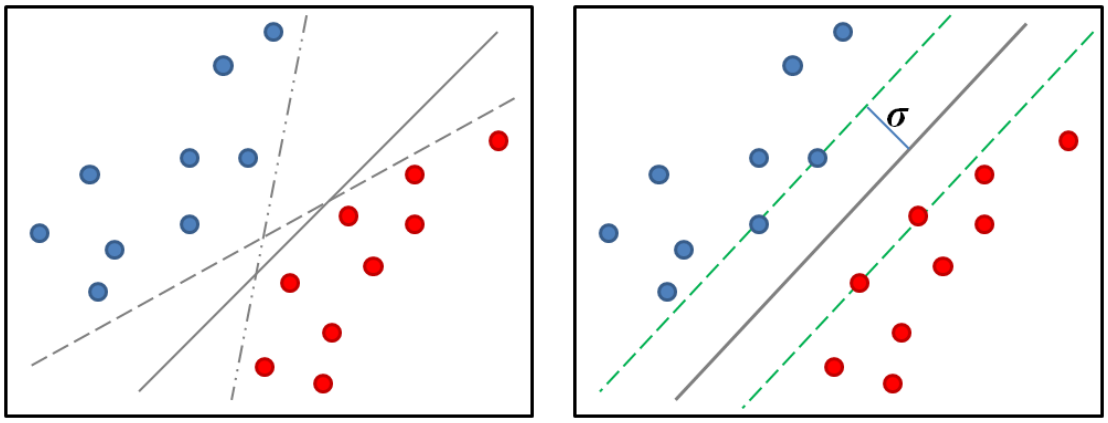


Fig. 1 A binary labeled dataset with red (positive) and blue (negative) samples.



(a) (b)

Fig. 2 Different separating hyperplanes resulted from different algorithms: (a) the hyperplanes (three gray lines) resulted from general linear classification algorithms, and (b) the hyperplane (gray line) resulted from linear SVM, where the margin σ between this hyperplane to each side of samples is marked.

2. The Optimization Form

The geometrical pursuit of SVM on a given dataset can be directly formulized as the following optimization problem:

$$\max_{\boldsymbol{w},b} \min_{n} \frac{y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)}{\|\boldsymbol{w}\|_2} \tag{1}$$

, which is an unconstraint problem with a discrete variable n. To eliminate this hardly-solved variable, an equivalent problem of (1) with an additional variable σ is derived:

$$\begin{aligned} &\max_{\boldsymbol{w},b,\sigma} \ \sigma \\ &\text{s.t.} \ \frac{y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)}{\|\boldsymbol{w}\|_2} \geq \sigma, \ \forall n. \end{aligned} \tag{2}$$

This equivalent problem transfers the minimization problem over a discrete variable into several inequality constraints. To be noticed, an unconstraint problem is usually more compelling than a constraint problem, while in SVM the constraint form makes it possible to modify the complicated optimization problem in (1) to a widely-studied optimization form, quadratic problem.

Unfortunately, (2) is not a quadratic programming problem and is still hard to solve, and the main bottleneck comes from the division terms in the constraints. One possible solution in to restrict the denominator to be of unit length:

$$\max_{\boldsymbol{w},b,\sigma} \ \sigma$$
$$\text{s.t.} \ \begin{cases} y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b) \geq \sigma, \ \forall n \\ \|\boldsymbol{w}\|_2 = 1 \end{cases} \tag{3}$$

, while the induced equality constraint is not an affine function, and so (3) is yet neither a convex nor a quadratic programming problem. Another possible solution is to restrict the product $\|\boldsymbol{w}\|_2 \sigma$ to some fixed value (ex. 1), and change the objective function into $1/\|\boldsymbol{w}\|_2$:

$$\max_{\boldsymbol{w},b} \ \frac{1}{\|\boldsymbol{w}\|_2}$$
$$\text{s.t.} \ y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b) \geq 1, \ \forall n \tag{4}$$
$$\text{(with an implicit constraint that } \|\boldsymbol{w}\|_2 \sigma = 1).$$

This modification fits all the constraints to the requirement of a convex optimization problem (affine equality constraints and convex inequality constraints) and a quadratic programming problem (both the two types of constraints are affine), and we can further change the maximization problem into a minimization problem:

$$\min_{\boldsymbol{w},b} \ \|\boldsymbol{w}\|_2$$
$$\text{s.t.} \ y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b) \geq 1, \ \forall n. \tag{5}$$

Finally, to make it convenient for computation and taking derivatives, the objective function is conventionally modified into a quadratic term:

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2$$
$$\text{s.t.} \ y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b) \geq 1, \ \forall n \tag{6}$$

, and now the optimization problem is exactly a quadratic programming problem. To be noticed, the optimization problems (1)-(6) are all equivalent to each other, and so by solving (6), we can find the separating hyperplane that meet the pursuit of the geometrical interpretation shown in Fig. 2 (b).

3. The Dual Form

The complexity of the quadratic optimization problem in (6) is controlled by the dimensionality of the features *d*, not by the training size *N*. If we only desire to find a linear hyperplane in the original feature space (usually *N*>>*d*), then directly using quadratic programming algorithm to solve (6) seems reasonable. While in most cases,

a linear hyperplane is not flexible enough to describe the underlying data distribution. To increase the flexibility without changing the optimization problem, applying feature transform is probably the most straightforward way. However, applying feature transform is accompanied by other questions, what kind of feature transform to use and how the complexity, controlled by the transformed feature dimensionality, will increase. It is not surprising that even using a low order of feature transform, say 3 to 6 order, may still results in a higher feature dimensionality d_Φ than N. This means that we need to strike a balance between model flexibility and computational complexity, a common issue of machine learning.

To overcome this issue, researchers were thinking whether the optimization problem can be refined to have a complexity dependent on N rather than d (d_Φ), and the dual problem formulation, a widely-used trick in convex optimization, seems to provide the rescue. In this section, we will briefly introduce the primal and dual form of a constraint optimization, and present the dual form of SVM.

3.1 The Primal and Dual Form

Lagrange multipliers, a simple yet powerful trick in mathematics, provide a direct way to transfer a constraint problem into an unconstraint one by introducing several new variables, called the Lagrange multipliers. Not losing the generality, from this section on, we assume that the all the functions appearing in the optimization problems are differentiable (a true assumption for SVM) to make the derivations clear. The *equivalent* unconstraint problem of (6) through applying Lagrange multipliers is shown below:

$$p^* = \min_{\boldsymbol{w},b} \max_{\alpha_n \geq 0} \underbrace{\frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{n=1}^{N} \alpha_n \left[1 - y^{(n)}(\boldsymbol{w}^T \boldsymbol{x}^{(n)} + b)\right]}_{\text{Lagrange dual function: } \mathcal{L}(\boldsymbol{w},b,\boldsymbol{\alpha})} \tag{7}$$

, where each constraint term is now attached with a Lagrange multiplier α_n. To be noticed, (7) is exactly equivalent to (6) and usually called the *primal* problem. The primal problem here is hard to solve because to the implicit constraint $\alpha_n \geq 0$, so people usually further derive the *dual* problem of (6) as follows:

$$d^* = \max_{\alpha_n \geq 0} \min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{n=1}^{N} \alpha_n \left[1 - y^{(n)}(\boldsymbol{w}^T \boldsymbol{x}^{(n)} + b)\right] \tag{8}$$

The only difference between (6) and (7) is the reverse order of the "max" and "min" problem. Yet, the dual problem indeed provides several good properties for optimization:

- The dual problem is always convex on the Lagrange multipliers α_n.
- The optimal value p^* of the primal problem is lower-bounded by the optimal vale d^* of the dual problem.
- When $p^* = d^*$, we say that the *strong duality* is held for the underlying problem, which is true for SVM. Besides, the strong duality condition further indicates that with optimal solution $(\boldsymbol{w}^*, b^*)$ of the primal problem $L(\boldsymbol{w}^*, b^*, \boldsymbol{\alpha}^*)$ is equal to d^* as well as p^*, where $\boldsymbol{\alpha}^*$ is the optimal solution of the dual problem. This important property tells us that the primal and dual problems are the same problem.
- The complexity of the dual problem depends on N (# Lagrange variables).

The proofs of these properties are beyond the scope of this tutorial, and the readers can refer to [3] for more information and details.

3.2 The KKT Condition

To solve an optimization problem, the most popular way is to check the *optimality condition*, the condition occurring if and only of the optimal solution is achieved. For example, the optimality condition of a convex function $f(\boldsymbol{x})$ without constraint is $\nabla_x f(\boldsymbol{x}) = 0$. Assume that the strong duality condition is held, the KKT conditions, the most well-known optimality condition for convex optimization problem, provide several simple rules to check if the optimal solution is attained. Furthermore, if the optimal solution is attained, the KKT conditions show several important properties of the underlying optimization problem. The KKT conditions of (7) and (8) are summarized as below:

- $$y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)} + b^*) \geq 1, \ \forall n \tag{9}$$
- $$\alpha_n^* \geq 0, \ \forall n \tag{10}$$
- $$\alpha_n^*\left[1 - y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)} + b^*)\right] = 0, \ \forall n \equiv \begin{cases} \text{if } \left[1 - y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)} + b^*)\right] < 0, \ \alpha_n^* = 0 \\ \text{if } \alpha_n^* > 0, \ \left[1 - y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)} + b^*)\right] = 0 \end{cases} \tag{11}$$
- $$\begin{aligned} &\nabla_{\boldsymbol{w}}\left\{\frac{1}{2}\left\|\boldsymbol{w}^*\right\|_2^2 + \sum_{n=1}^{N}\alpha_n^*\left[1 - y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)} + b^*)\right]\right\} = 0 \\ &\nabla_b\left\{\frac{1}{2}\left\|\boldsymbol{w}^*\right\|_2^2 + \sum_{n=1}^{N}\alpha_n^*\left[1 - y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)} + b^*)\right]\right\} = 0 \end{aligned} \tag{12}$$

3.3 The Dual Form of SVM

According to the KKT conditions, now we can derive, or say simplify, the dual form of SVM based on (8). At first, we use the forth condition (12) to solve the "min" problem of (8) over $\boldsymbol{w}$ and b, which results in the following derivations:

$$\nabla_{\boldsymbol{w}}\left\{\frac{1}{2}\|\boldsymbol{w}\|_2^2+\sum_{n=1}^{N}\alpha_n\left[1-y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)\right]\right\}=\boldsymbol{w}+\sum_{n=1}^{N}\alpha_n(-y^{(n)}\boldsymbol{x}^{(n)})=0, \tag{13}$$

$$\boldsymbol{w}=\sum_{n=1}^{N}\alpha_n(y^{(n)}\boldsymbol{x}^{(n)}), \tag{14}$$

$$\nabla_{b}\left\{\frac{1}{2}\|\boldsymbol{w}\|_2^2+\sum_{n=1}^{N}\alpha_n\left[1-y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)\right]\right\}=-\sum_{n=1}^{N}\alpha_n y^{(n)}=0, \tag{15}$$

$$\sum_{n=1}^{N}\alpha_n y^{(n)}=0. \tag{16}$$

Plugging (14) and (16) back into (8), we can achieve an optimization problem that depends only on the Lagrange multipliers, more precisely the dual variables α_n:

$$\begin{aligned}
&\max_{\alpha_n\ge 0}\ \min_{\boldsymbol{w},b}\ \frac{1}{2}\|\boldsymbol{w}\|_2^2+\sum_{n=1}^{N}\alpha_n\left[1-y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)\right]\\
&=\max_{\alpha_n\ge 0}\ \frac{1}{2}\left\|\sum_{n=1}^{N}\alpha_n(y^{(n)}\boldsymbol{x}^{(n)})\right\|_2^2+\sum_{n=1}^{N}\alpha_n\left\{1-y^{(n)}\left[\left(\sum_{n=1}^{N}\alpha_n(y^{(n)}\boldsymbol{x}^{(n)})\right)^T\boldsymbol{x}^{(n)}+b\right]\right\} \qquad (17)\\
&=\max_{\alpha_n\ge 0}\ \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}+\sum_{n=1}^{N}\alpha_n-\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}-b\sum_{n=1}^{N}\alpha_n y^{(n)}\\
&=\min\ \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}-\mathbf{1}^T\boldsymbol{\alpha},\ \text{ s.t. }\sum_{n=1}^{N}\alpha_n y^{(n)}=0\text{ and }\alpha_n\ge 0\ \forall n \qquad (18)
\end{aligned}$$

, where Q stands for the following matrix:

$$\begin{aligned}
Q&=\begin{bmatrix}(y^{(1)}y^{(1)})\boldsymbol{x}^{(1)T}\boldsymbol{x}^{(1)} & \cdots & y^{(N)})\boldsymbol{x}^{(1)T}\boldsymbol{x}^{(N)}\\ \vdots & \ddots & \vdots\\ (y^{(N)}y^{(1)})\boldsymbol{x}^{(N)T}\boldsymbol{x}^{(1)} & \cdots & y^{(N)})\boldsymbol{x}^{(N)T}\boldsymbol{x}^{(N)}\end{bmatrix}\\
&=\begin{bmatrix}y^{(1)} & 0 & 0\\ 0 & \ddots & \\ 0 & 0 & y^{(N)}\end{bmatrix}\begin{bmatrix}- & \boldsymbol{x}^{(1)T} & -\\ & \vdots & \\ - & \boldsymbol{x}^{(N)T} & -\end{bmatrix}\begin{bmatrix}| & & |\\ \boldsymbol{x} & \cdots & \boldsymbol{x}\\ | & & |\end{bmatrix}\begin{bmatrix}y^{(1)} & 0 & 0\\ & \ddots & \\ 0 & 0 & y^{(N)}\end{bmatrix} \qquad (19)\\
&=Y^T(X^TX)Y=Y^TKY.
\end{aligned}$$

The matrix K is usually named the kernel matrix. As shown in (18), the dual problem is a quadratic programming problem with variables α_n. The necessary condition that the optimal solution of (18) is attainable, the optimal value of (18) is finite, is to ensure that Q is positive semi-definite, which is equivalent to ensure that K to be *positive semi-definite* (valid kernel). From (19), $K = (X^T X)$ does indicate that K is a valid kernel, and therefore by solving (18), with the complexity dependent on N, we equivalently solve the original optimization problem (6) of SVM.

3.4 The Relationship Between the Primal and Dual Optimal Solutions

The above conclusion may confuse the readers because even the optimal solution $\boldsymbol{\alpha}^*$ of (18) is found, without $\boldsymbol{w}^*$ and b^* of the primal problem (7) we still cannot predict the label of an unseen sample. Before answering this question, let's first review the linear model of SVM. Given a sample vector $\boldsymbol{x}$, SVM predicts the label y by the following equation:

$$y = \boldsymbol{w}^{*T}\boldsymbol{x} + b^*. \tag{20}$$

According to (17) and the properties of the dual problem in Section 3.1, we can directly compute $\boldsymbol{w}^*$ based on $\boldsymbol{\alpha}^*$. And to find the relationship, we should go back to the KKT conditions mentioned in Section 3.2. As described in (11), $\boldsymbol{\alpha}_n^* > 0$ only occurs when $1 = y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)} + b^*)$ for some n. So by randomly pick a sample $\boldsymbol{x}^{(m)}$ with $\boldsymbol{\alpha}_m^* > 0$, b^* can be computed as:

$$b^* = (1 - y^{(n)}\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)}) / y^{(n)}. \tag{21}$$

To alleviate the possible rounding error during computation, people usually compute b for each sample with $\boldsymbol{\alpha}_n^* > 0$ and then taking the average to achieve a much stable b^*.

3.5 The Meaning of Support Vectors

After these complicated equations and derivations, now you may ask an interesting question why this linear classification method is called the *support vector* machine. To answer this question, let's take a second look at Fig. 2 (b), and you may

suddenly realized that if the four samples passed by the green lines have been determined from the dataset, then the separating hyperplane as well as the margin can be directly computed. Surprisingly, by combining the results in (1)-(6) and (11), it can be proved that the samples with $\alpha_n^* > 0$ are exactly the ones that can directly determine the margin as well as separating hyperplane of SVM. In other words, both $\boldsymbol{w}^*$ and b^* can be derived only by the sample vectors with $\alpha_n^* > 0$, and these samples are usually called the *support vectors*.

4. The Soft-Margin SVM

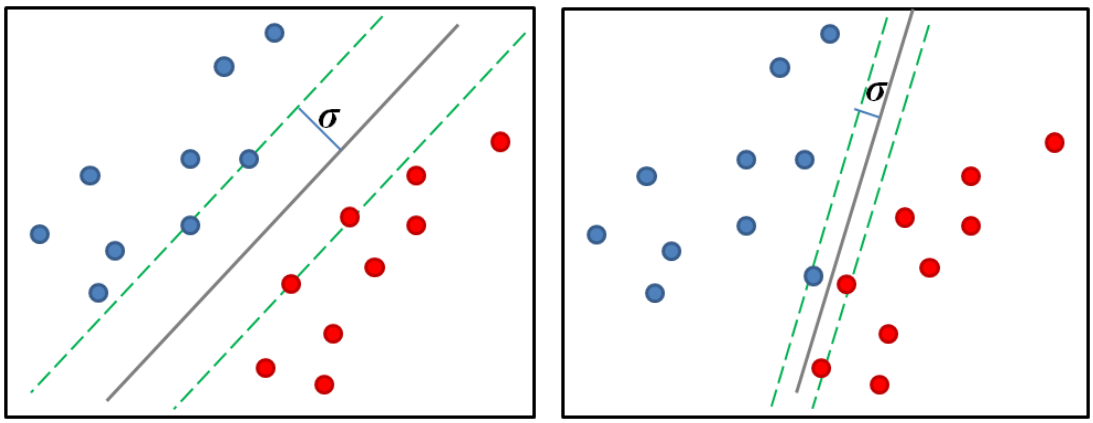
In the previous sections, SVM always assumes that the dataset is linear-separable, and therefore a separating hyperplane without training errors can be found in the dataset (This requirement is embedded in the constraints of (6)). Unfortunately, in most of the practical cases, the dataset is not linear-separable. One possible solution, mentioned in the beginning of Section 3, is to perform feature transform, hoping that a separating hyperplane without training errors exists in the transformed space (usually with higher dimensionality). However, according to the learning theory, increasing the feature dimensionality through feature transform simultaneously enlarges the model complexity and the generalization error; so even if the learned SVM model can achieve zero E_{in} in the high dimensional transformed space, we cannot guarantee the model to have a low E_{out} because of the possible *over-fitting* problem. In addition, there is no clue that what kinds of feature transform are doomed to produce a linear-separable transformed space.

Even though the dataset is linear-separable, noise may still occur in it. When some samples, affected by the noise, invade the margin computed by the other samples, SVM has to rerun the optimization process to fit these samples and define a new hyperplane as well as margin. This observation shows the likelihood that SVM *over-fits* to the noise and results in a poor separating hyperplane with a smaller margin. Fig. 3 shows the case.

To get away from the *over-fitting* and *noise-fitting* problem while maintain the geometrical interpretation and the beautiful formulation of SVM, another possible solution is to relax the hard constraint of SVM, to allow the violation of margin. This desire can be accomplished by introducing the *slack* variables and *loss terms* into the constraints of (6), and the refined SVM is called the soft-margin SVM; on contrary, SVM derived from (6) is called the hard-margin SVM.

4.1 Slack variables

The basic idea of soft-margin SVM is to allow the violation of margin, meaning that data samples can locate in the margin defined by SVM, while punish these violating samples with a loss term. This relaxation can therefore result in a bigger margin (with better generalization ability) and still restrict the number of violating samples to ensure a small training error. Fig. 4 illustrates the resulting hyperplane by the soft-margin SVM on the dataset shown in fig. 3 (b).



(a) (b)

Fig. 3 The case of noise-fitting of SVM: (a) the original dataset with a larger margin, and (b) the refined smaller margin after a noisy sample coming in.

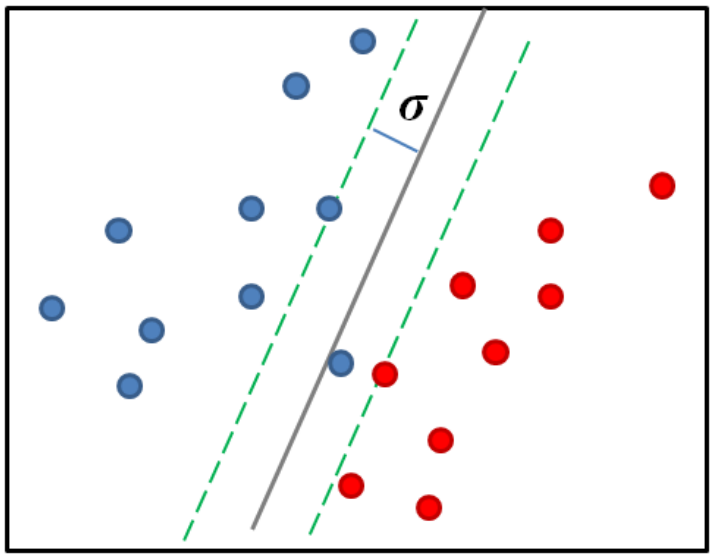


Fig. 4 The separating hyperplane resulted from the soft-margin SVM. The soft-margin SVM allows the added sample in fig. 3 (b) to invade the margin (marked by the green lines), and so a wider margin can be achieved.

To model this relaxation, slack variables ξ_n and a penalty function $f(\xi_n)$ are introduced into the constraints in (6):

$$\begin{aligned} &\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{n=1}^{N} f(\xi_n) \\ &\text{s.t. } y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + b) \geq 1 - \xi_n, \ \forall n \\ &\quad \xi_n \geq 0, \ \forall n \end{aligned} \tag{22}$$

, where ξ_n are constrained to be non-negative to avoid model degradation. The parameter C is a trade-off parameter to balance the violation of margin (E_{in}) and the width of margin (generalization ability). The penalty function $f(.)$ can be any convex function on ξ_n, which ensure the global optima to be reached: when $f(.)$ is convex, the objective function of (22) is then convex, and any local optima achieved is exactly the global optima. In fact, there are other ways to model the margin violation of SVM, and the form in (22) is usually called the C-SVM.

The most widely-used penalty function in electrical engineering and computer science is probably the least-square error (L_2 error), a convex function, while in soft-margin SVM, the L_1 error is more popular because it is claimed to achieve better error tolerance (robust).

4.2 The Hinge Loss

Plugging the L_1 loss penalty function into (22), we can get the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{n=1}^{N} \xi_n \\ &\text{s.t. } y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + b) \geq 1 - \xi_n, \ \forall n \\ &\quad \xi_n \geq 0, \ \forall n \end{aligned} \tag{23}$$

, where the $|.|$ sign on ξ_n has been ignored because of the non-negative constraint on it. According to (23), now we can derive the dual form for the soft-margin SVM. While at first, we make a bypass to transfer (23) into an unconstraint form — not by the Lagrange multipliers — that can be directly compared with other kinds of linear classification:

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{n=1}^{N} \max\left\{0, \ 1 - y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)} + b)\right\}. \tag{24}$$

This unconstraint form is composed of a regularization term (for model complexity) and a loss term (for training error), which is exactly the standard form of linear classification modeling. And to be clear, (23) and (24) are equivalent optimization problems. The error term used in (24):

$$f(t) = \max(0, \ t), \ t \in R, \tag{25}$$

is called the Hinge loss, and so some people calls the soft-margin SVM with the L_1 loss the Hinge loss SVM.

4.3 The Dual Form of Soft-Margin SVM

To derive the dual form of (23), we first transfer it into the unconstraint form, the primal problem, by Lagrange multipliers:

$$\min_{\boldsymbol{w},b,\xi_n}\max_{\alpha_n\geq 0,\beta_n\geq 0}\ \frac{1}{2}\|\boldsymbol{w}\|_2^2+C\sum_{n=1}^{N}\xi_n+\sum_{n=1}^{N}\alpha_n\left[1-\xi_n-y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)\right]-\sum_{n=1}^{N}\beta_n\xi_n. \tag{26}$$

By changing the order of "max" and "min", we can get the dual problem:

$$\max_{\alpha_n\geq 0,\beta_n\geq 0}\min_{\boldsymbol{w},b,\xi_n}\ \frac{1}{2}\|\boldsymbol{w}\|_2^2+C\sum_{n=1}^{N}\xi_n+\sum_{n=1}^{N}\alpha_n\left[1-\xi_n-y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)\right]-\sum_{n=1}^{N}\beta_n\xi_n. \tag{27}$$

The optimization form of soft-margin SVM does hold the strong duality property, so we can further simplify (27) by the KKT conditions. Taking the derivative over $\boldsymbol{w}$ and b and $\boldsymbol{\xi}$ for the minimization problem in (27), the following equations can be achieved:

$$\begin{aligned}&\nabla_{\boldsymbol{w}}\left\{\frac{1}{2}\|\boldsymbol{w}\|_2^2+C\sum_{n=1}^{N}\xi_n+\sum_{n=1}^{N}\alpha_n\left[1-\xi_n-y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)\right]-\sum_{n=1}^{N}\beta_n\xi_n\right\}\\&=\boldsymbol{w}-\sum_{n=1}^{N}\alpha_n y^{(n)}\boldsymbol{x}^{(n)}=0\end{aligned} \tag{28}$$

$$\boldsymbol{w}=\sum_{n=1}^{N}\alpha_n y^{(n)}\boldsymbol{x}^{(n)} \tag{29}$$

$$\begin{aligned}&\nabla_{b}\left\{\frac{1}{2}\|\boldsymbol{w}\|_2^2+C\sum_{n=1}^{N}\xi_n+\sum_{n=1}^{N}\alpha_n\left[1-\xi_n-y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)\right]-\sum_{n=1}^{N}\beta_n\xi_n\right\}\\&=-\sum_{n=1}^{N}\alpha_n y^{(n)}=0\end{aligned} \tag{30}$$

$$\sum_{n=1}^{N}\alpha_n y^{(n)}=0 \tag{31}$$

$$\begin{aligned}&\nabla_{\xi_n}\left\{\frac{1}{2}\|\boldsymbol{w}\|_2^2+C\sum_{n=1}^{N}\xi_n+\sum_{n=1}^{N}\alpha_n\left[1-\xi_n-y^{(n)}(\boldsymbol{w}^T\boldsymbol{x}^{(n)}+b)\right]-\sum_{n=1}^{N}\beta_n\xi_n\right\}\\&=C-\alpha_n-\beta_n=0.\end{aligned} \tag{32}$$

Plugging (29), (31), and (32) back into (27), we can have the following derivations:

$$\max_{\alpha_n\geq 0,\beta_n\geq 0}\ \frac{1}{2}\left\|\sum_{n=1}^{N}\alpha_n y^{(n)}\boldsymbol{x}^{(n)}\right\|_2^2+\sum_{n=1}^{N}(C-\beta_n-\alpha_n)\xi_n+\sum_{n=1}^{N}\alpha_n\left\{1-y^{(n)}\left[\left(\sum_{n=1}^{N}\alpha_n(y^{(n)}\boldsymbol{x}^{(n)})\right)^T\boldsymbol{x}^{(n)}+b\right]\right\}$$

$$=\max_{\alpha_n\geq 0,\beta_n\geq 0}\ -\frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}+\sum_{n=1}^{N}(C-\beta_n-\alpha_n)\xi_n+\sum_{n=1}^{N}\alpha_n-b\sum_{n=1}^{N}\alpha_n y^{(n)}$$

$$=\max_{\alpha_n\geq 0,\beta_n\geq 0}\ -\frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}+\sum_{n=1}^{N}\alpha_n$$

$$=\min\ \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}+\sum_{n=1}^{N}\alpha_n \tag{33}$$

$$\text{s.t.}\begin{cases}\alpha_n\geq 0,\ \beta_n\geq 0,\ C-\alpha_n-\beta_n=0\ \forall n\\ \sum_{n=1}^{N}\alpha_n y^{(n)}=0\end{cases}$$

$$=\min\ \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}+\mathbf{1}^T\boldsymbol{\alpha}$$

$$\text{s.t.}\begin{cases}C\geq\alpha_n\geq 0,\ \forall n\\ \sum_{n=1}^{N}\alpha_n y^{(n)}=0\end{cases} \tag{34}$$

, and finally a quadratic programming over α_n is reached, which is very similar to (18) while with an additional constraint term $C\geq\alpha_n\geq 0$. The definition of Q follows the one in (19).

After simplifying the dual problem, again we need to find the relationship between the primal solutions $\boldsymbol{w}^*$ and b^* and the dual solutions $\boldsymbol{\alpha}^*$. Apparently, (29) gives us the relationship between $\boldsymbol{w}^*$ and $\boldsymbol{\alpha}^*$; however, we cannot directly follow the procedure in Section 3.4 to find the relationship between b^* and $\boldsymbol{\alpha}^*$ because of the additional (and unknown) ξ_n^* in (27). To solve this problem, we first list the terms in (27) that should follow the KKT condition in (11):

$$\forall n,\ \begin{cases}\alpha_n^*\left[1-\xi_n^*-y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)}+b^*)\right]=0\\ \beta_n^*\xi_n^*=0\end{cases}. \tag{35}$$

From (11), we know that $\beta_n^*>0$ if and only if $\xi_n^*=0$, which means that the corresponding samples do not violate the margin. Combining this observation with (32), we immediately find that:

$$\text{for } C>\alpha_n^*>0,\ \alpha_n^*\left[1-y^{(n)}(\boldsymbol{w}^{*T}\boldsymbol{x}^{(n)}+b^*)\right]=0. \tag{36}$$

According to (36), we can choose any sample with $C > \alpha_n^* > 0$ and then apply the same procedure in Section 3.4 to compute b^*.

4.4 The Type of Support Vectors

The observations in (35) and (36) indicate that there are two kinds of support vectors in the soft-margin SVM: one is with $C > \alpha_n^* > 0$ and one is with $C = \alpha_n^*$. The first kind of support vectors implies $\beta_n^* > 0$ and $\xi_n^* = 0$, meaning that these vectors do not violate the margin; indeed, they lie on the margin boundary (the parallel green lines in fig. 4) because of $\alpha_n^* > 0$. The second kind of support vectors implies $\beta_n^* > 0$, which occurs either when $\xi_n^* > 0$ or $\xi_n^* = 0$, so they may either lies on the margin boundary or violate the margin: the sample violating the margin in fig. 4 must have $C = \alpha_n^*$.

5. The Kernel Extension of SVM

In the previous two sections, both the dual problem of hard-margin SVM and soft-margin SVM with Hinge loss are derived. The dual problem transfers the complexity dependent on dimensionality to the one dependent on the number of samples, and the main purpose for doing so is to allow the usage of feature transform into a higher dimensional space: Whatever feature transforms are applied, after building the kernel matrix *K* in (19), the complexity of the quadratic programming problem in (18) or (34) only depends on *N*. However, the complexity for computing *K* is still dependent on the dimensionality of the transformed features. From (19), we have to first transform the original feature vector $\boldsymbol{x}^{(n)}$ into $\Phi(\boldsymbol{x}^{(n)})$, and then take the product computation between two d_Φ-dimensional vectors for building *K*. This procedure spends not only time, but also memory, and will be intractable for some kinds of feature transforms. In order to solve this problem, the kernel trick — having been widely researched and used for making a linear model nonlinear — is introduced for soft-margin SVM (for convenience, the term “soft-margin” is neglected in the later sections).

5.1 The Definition of Kernels

In (19), we need to compute $\Phi(\boldsymbol{x}^{(i)})^T \Phi(\boldsymbol{x}^{(j)})$ for each sample pair (i, j), which is with a complexity $O(d_\Phi)$. However, if we can find a function k (.) for a specific feature transform like the following:

$$k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \Phi(\boldsymbol{x}^{(i)})^T \Phi(\boldsymbol{x}^{(j)}) \tag{37}$$

, the computational complexity can be drastically reduced: even to the desired condition $O(d)$. The following shows the kernel function for two-dimensional vectors with the second-order feature transform:

$$\Phi(\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 & \sqrt{2}x_1 & \sqrt{2}x_2 & 1 \end{bmatrix}^T \tag{38}$$

$$\begin{aligned} &\Phi(\begin{bmatrix} x_1^{(i)} & x_2^{(i)} \end{bmatrix}^T)^T \Phi(\begin{bmatrix} x_1^{(j)} & x_2^{(j)} \end{bmatrix}^T) \\ &= (x_1^{(i)2}x_1^{(j)2} + 2x_1^{(i)}x_2^{(i)}x_1^{(j)}x_2^{(j)} + x_2^{(i)2}x_2^{(j)2} + 2x_1^{(i)}x_1^{(j)} + 2x_2^{(i)}x_2^{(j)} + 1) \\ &= (x_1^{(i)}x_1^{(j)} + x_2^{(i)}x_2^{(j)} + 1)^2 \\ &= (1 + \boldsymbol{x}^{(i)T}\boldsymbol{x}^{(j)})^2 \\ &= k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}). \end{aligned} \tag{39}$$

In fact, for any feature transform, we can find the corresponding kernel function; for any function defined on $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$, we may not find the corresponding feature transform. However, it is much harder to define the kernel function for a specific feature transform than to directly use some popular functions for measuring the relationship between $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$, without proving that these functions are valid kernel functions.

To release this problem, a relaxation for the definition of kernel functions called the *Mercer's rule* is introduced: A function can be used as a kernel function if and only if the resulting kernel matrix K, by substituting $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ for $\boldsymbol{x}^{(i)T}\boldsymbol{x}^{(j)}$ in (19), is positive semi-definite. And the transformed space, implicitly corresponding to a kernel function, is called the reproducing Hilbert space.

5.2 Some Popular Kernels

In this subsection, we introduce the two most popular kernel functions: the polynomial kernel and the radial basis kernel (RBF kernel).

- Polynomial kernel: The polynomial kernel is defined as:

$$k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = (\gamma + c\boldsymbol{x}^{(i)T}\boldsymbol{x}^{(j)})^{\tau} \tag{40}$$

, where τ is the polynomial degree, while γ and c jointly determine the trade-off between high-order features and low-order features.

- RBF kernel: The RBF kernel is defined as:

$$k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \exp(-\gamma \left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\|_2^2) \tag{41}$$

, where γ again controls the trade-off between high-order features and low-order features. To be more practical, increasing γ is equivalent to increasing the model complexity of SVM. The RBF kernel corresponds to transforming the original features into an infinite-dimensional space and can results in highly flexible separating hyperplane, so it is the most widely-used kernel type nowadays.

6. Key Points of SVM

In this section, we list and discuss some important or interesting points of SVM.

6.1 The Power of SVM

As discussed in the learning theory, a good classification model should have a suitable model complexity to strike a balance between the training error and the generalization error, and to avoid over-fitting and under-fitting. The standard form of linear classification modeling, composed of a regularized term and a loss term, does provide an objective function that balances these two errors; however, there is no effective and efficient way to adjust the model complexity.

The standard form of soft-margin SVM, defined in (22), also has the error balancing formulation. Furthermore, the dual form and the kernel trick can effectively and efficiently adjust the model complexity by trying different kinds of kernels and different value of *C* in (22). In other words, the kernel trick (existing in the dual form) provides the flexibility of separating boundaries, the $\|\boldsymbol{w}\|_2^2$ term in (22) provides the mechanism of restricting the complexity, and the parameter *C* and the kernel parameters provide the freedom of adjusting the model. These factors collaborate with each other and result in the robustness and flexibility of SVM. In Fig. 5, we show how

SVM with the kernel trick (RBF kernel) can produce flexible separating boundaries for the underlying task.

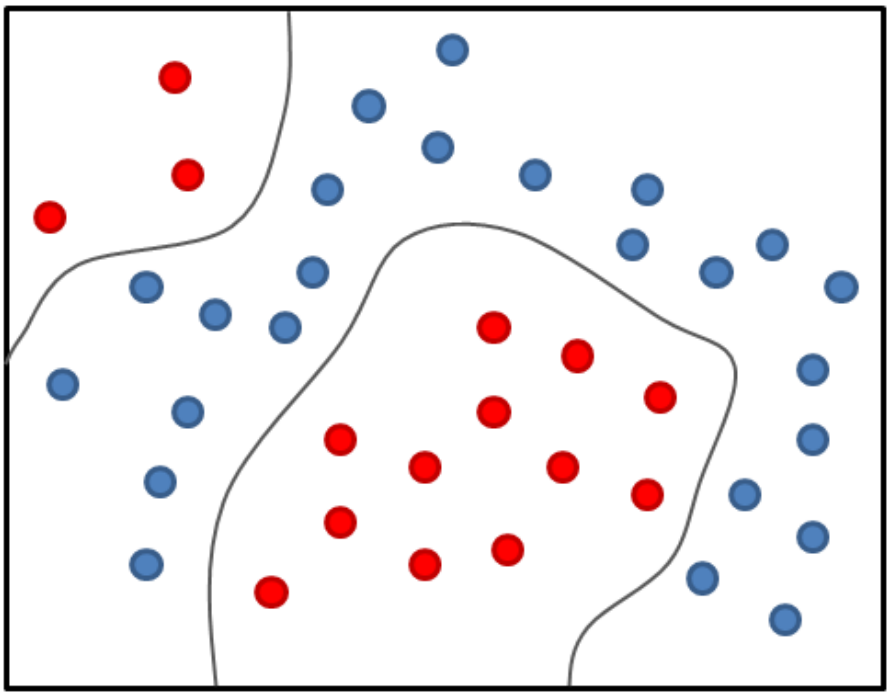


Fig. 5 The flexible separating boundaries (gray curves) that can be achieved by using the RBF kernel.

6.2 The Key Parameters

Having discussed the power of SVM, now you may ask about what parameters a user can choose and what their meanings are. When using SVM, the first thing you can choose is the kernel type (in fact, you can choose other optimization form of SVM rather than C-SVM). Take the RBF kernel as an example, before training the SVM model, two parameters, C and γ, need defined. The parameter C balances the regularized term and the loss term: when it increases, the loss term is more important and the model tends to be over-fitting; when it decreases, the regularized term is more important and the model tends to be under-fitting. The parameter γ balance the priority of higher-order features and lower-level features: a large γ is proved to have a higher model complexity. In practice, we often pre-select a range of C and γ, and perform the brute-force search to select the based parameters based on the cross validation error.

6.3 SVM for Prediction

Having trained the SVM model and achieved the dual optimal solutions $\boldsymbol{\alpha}^*$, we

can use the relationship defined in Section 3.4 or 4.3 to compute the primal optimal solutions $\boldsymbol{w}^*$ and b^*, and then apply the linear classification model (20) to predict the label for a new feature vector. However, when the kernel trick is used, the dimensionality of $\boldsymbol{w}^*$ will be d_Φ and a d_Φ-dimensional product has to be performed in (20), making the prediction process rather slow. To overcome this drawback, let's take a twice look at the relationships between primal and dual solution:

$$\boldsymbol{w}^* = \sum_{n=1}^{N} \alpha_n^* (y^{(n)} \boldsymbol{x}^{(n)}) \tag{42}$$

$$b^* = (1 - y^{(m)} \boldsymbol{w}^{*T} \boldsymbol{x}^{(m)}) / y^{(m)} \text{ with } C > \alpha_m^* > 0. \tag{43}$$

When applying the feature transform, these two equations will be come:

$$\boldsymbol{w}^* = \sum_{n=1}^{N} \alpha_n^* (y^{(n)} \Phi(\boldsymbol{x}^{(n)})) \tag{43}$$

$$b^* = (1 - y^{(m)} \boldsymbol{w}^{*T} \Phi(\boldsymbol{x}^{(m)})) / y^{(m)} \text{ with } C > \alpha_m^* > 0. \tag{44}$$

It is clear that a d_Φ-dimensional product has to be computed in (44), but this operation can be simplified by the kernel trick as follows:

$$\begin{aligned} b^* &= (1 - y^{(m)} \left[\sum_{n=1}^{N} \alpha_n^* (y^{(n)} \Phi(\boldsymbol{x}^{(n)})) \right]^T \Phi(\boldsymbol{x}^{(m)})) / y^{(m)} \\ &= (1 - y^{(m)} \sum_{\alpha_n^* \neq 0} \alpha_n^* y^{(n)} k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)})) / y^{(m)}. \end{aligned} \tag{45}$$

Furthermore, when considering the classification model in (20), the kernel trick can also be applied:

$$\begin{aligned} y &= \boldsymbol{w}^{*T} \Phi(\boldsymbol{x}^{(n)}) + b^* \\ &= \left[\sum_{n=1}^{N} \alpha_n^* (y^{(n)} \Phi(\boldsymbol{x}^{(n)})) \right]^T \Phi(\boldsymbol{x}^{(n)}) + b^* \\ &= \sum_{\alpha_n^* \neq 0} \alpha_n^* y^{(n)} k(\boldsymbol{x}^{(n)}, \boldsymbol{x}) + b^*. \end{aligned} \tag{46}$$

By this reformulation, SVM can easily achieve real-time prediction even with a complicated feature transform.

6.4 Multi-class problems

The SVM model described above are only for the binary classification case, and to extend it for the multi-class problem, the most straightforward way is to apply the one-versus-one or one-versus all extension.

6.5 Interesting Points of SVM

Although there is no guarantee that how many zeros the solution $\boldsymbol{\alpha}^*$ contains, it is usually the case that a great portion of elements in $\boldsymbol{\alpha}^*$ will be zero. As a consequence, only a portion of training samples and their corresponding α_n^* need remained for testing. In addition, even if the samples with zero α_n^* are missed in the training phase, the model of SVM won't be changed. Moreover, the leave-one-out error (LOO) of SVM is upper-bounded by the proportion of nonzero elements in $\boldsymbol{\alpha}^*$.

7. Practical Package: LIBSVM (Source Code)

There have been several released SVM packages that can be freely downloaded on-line, and among them, LIBSVM [2] — developed and maintained by the laboratory of C.J. Lin in National Taiwan University — is the most popular one because of its multiple interfaces and highly efficiency. LIBSVM is written in C, and utilizes the sequential minimal optimization (SMO), an optimization method specifically designed for SVM, to solve the dual problem of SVM. LIBSVM provides several models of SVM, such as C-SVM, SVR (support vector regression), and one-class SVM. Furthermore, it provides the Matlab interface and so users familiar with this programming tool can easily apply SVM in their work. Besides, LIBSVM allows the multi-class input, and deal with it by the one-versus-one extension.

The compressed package of LIBSVM is attached with a comprehensive readme that gives a clear description of how to use LIBSVM. Besides, the introduction papers [4], [5] written by the same laboratory also provide good introductions of SVM. So if you are a notice of SVM, just try LIBSVM first.

8. Conclusion

In this tutorial, we provide a broad introduction of SVM, yet some key points and details also be discussed. After getting familiar with how SVM works, the free

packages available on-line are worth a try. Because the maturity of SVM, it has become the baseline for most of the researches requiring the use of machine learning techniques. As a result, compared to the early age of SVM, now we cannot just collect a couple of features and use SVM for prediction to publish a paper. However, the way how SVM is built is still a popular method to model the classification or regression tasks which are challenging nowadays.

9. Reference

[1] SVM^{light}: http://svmlight.joachims.org/.

[2] LIBSVM: www.csie.ntu.edu.tw/~cjlin/libsvm.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, 2004.

[4] C. W. Hsu, C.C. Chang, and C. J. Lin, "A practical guide to support vector machine classification," available at: www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] C.C. Chang and C.J. Lin, "LIBSVM: a library for support vector machines," ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.