Max-Margin Classification

Avi Pfeffer; Revised by David Parkes and Ryan P. Adams

Support vector machines (SVMs) are an approach to classification that has received a lot of interest in the past few years and the SVM framework is currently one the most popular approaches to supervised learning. This makes it worthwhile to try to understand some of the underlying mathematics behind SVMs. The purpose of these notes is not for you to understand all the details of SVMs, but to know enough about them that you can apply them intelligently to a problem that you might encounter.

SVMs are based on three big ideas.[1] The first is maximizing the *margin*. Intuitively, maximizing the margin means that when we learn a linear decision boundary, we should try to choose the decision boundary that maximizes the distance to the "hardest" examples – those that data that are closest to the boundary. The second big idea is *duality*. This is an idea that is used many times in optimization problems. It allows one problem to be transformed into another problem that may be easier to solve. The third big idea is the "kernel trick." *Kernels* allow input vectors to be mapped into higher-dimensional, and therefore more expressive, feature spaces, but without necessarily incurring the full computational cost that one might expect.

1 Maximizing the Margin

We will mainly consider binary classification problems, where the training data are tuples $\{\boldsymbol{x}_n, t_n\}_{n=1}^N$, and we'll identify the target class via -1 and $+1$ so $t_n \in \{-1, +1\}$. As with other linear classifiers we have examined, SVMs seek to learn linear functions

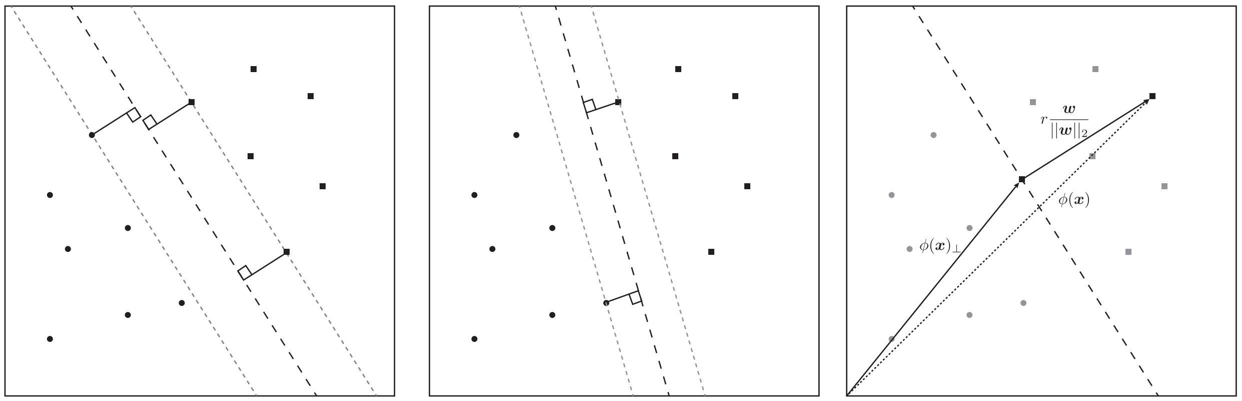
$$f(\boldsymbol{x}, \boldsymbol{w}, b) = \boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}) + b \tag{1}$$

where $\boldsymbol{w}$ is an J-dimensional column vector of weights, $\boldsymbol{\phi}(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}^J$ is a collection of J basis functions that take inputs and produce a useful feature representation. As before, think of $\boldsymbol{\phi}$ as transforming the original data set into a new and better representation of some kind. The jth of these functions is denoted by $\phi_j(\cdot)$. The inner product $\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x})$ weights these functions to produce a scalar output. The scalar value b is the bias, as in other linear classifiers.

As before, we classify using zero as the threshold between classes:

$$y(\boldsymbol{x}, \boldsymbol{w}, b) = \begin{cases} +1 & \text{if } \boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}) + b > 0 \\ -1 & \text{otherwise} \end{cases}. \tag{2}$$

[1]These notes are based, in part, on Bishop (1998).


$r\frac{\boldsymbol{w}}{||\boldsymbol{w}||_2}$
$\phi(\boldsymbol{x})$
$\phi(\boldsymbol{x})_\perp$

(a) Margin for a boundary in $\mathbb{R}^2$ (b) Boundary with a poor margin. (c) Decomposing into two vectors

Figure 1: Examples of decision boundaries and margins in $\mathbb{R}^2$.

We will assume for the most part that the training set is linearly separable in the feature space $\boldsymbol{\phi}(\boldsymbol{x})$, so that there exist parameters $\boldsymbol{w}$ and b such that

$$\begin{aligned} \boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b > 0 \qquad &\text{for} \qquad t_n = +1 \\ \boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b \leq 0 \qquad &\text{for} \qquad t_n = -1\,. \end{aligned}$$

The "margin" of a classifier measures, intuitively, the distance of the decision boundary to the closest examples. As in other classifiers we have examined, the (linear) decision boundary in the J-dimensional feature space is the line in $\mathbb{R}^J$ where

$$\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0\,. \tag{3}$$

This is the place where $y(\boldsymbol{x}, \boldsymbol{w}, b)$ changes from producing one class, to producing the other. Note that as a function of $\boldsymbol{x}$, this may not be a plane, as $\boldsymbol{\phi}(\cdot)$ may be complicated and nonlinear.

Definition 1 *The margin for a single example $\mathbf{x}_n$ is the orthogonal distance to the decision boundary in $\mathbb{R}^J$ if $\mathbf{x}_n$ is classified correctly, and the negated orthogonal distance if $\mathbf{x}_n$ is classified incorrectly.*

Definition 2 *The margin on a training set (or simply the "margin") is the minimum margin on all examples in the training set.*

Figure 1a shows the margin on the closest examples in $\mathbb{R}^2$ for a particular classifier (as indicated by its linear decision boundary.) One thing to note is that there are several closest examples to the hyperplane. This often happens in practice with SVMs. SVMs find a linear separator in possibly high dimensional feature space $\mathbb{R}^J$. The goal of an SVM is to find a linear separator that maximizes the margin given a particular input representation $\boldsymbol{x} \in \mathcal{X}$ and a feature mapping $\boldsymbol{\phi}(\cdot)$. Figure 1b shows a hypothesis with a smaller margin than Figure 1a. Intuitively, the second hypothesis is less ideal, because it has less robustness in being able to classify unseen examples that are distributed around the current examples. We might therefore expect it to generalize less well to unseen examples.

Given parameters $\boldsymbol{w}$ and b, vector $\boldsymbol{w}$ is orthogonal to every vector in $\mathbb{R}^J$ that lies on the decision boundary. To see this, consider two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, both on the decision boundary. Then, the vector $\boldsymbol{\phi}(\boldsymbol{x}_2) - \boldsymbol{\phi}(\boldsymbol{x}_1)$ is in the decision boundary hyperplane. We can examine how this vector projects onto $\boldsymbol{w}$:

$$\begin{aligned} \boldsymbol{w}^{\mathsf{T}}(\boldsymbol{\phi}(\boldsymbol{x}_2) - \boldsymbol{\phi}(\boldsymbol{x}_1)) &= \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) && (4) \\ &= (-b) - (-b) = 0\,. && (5) \end{aligned}$$

Thus vectors in the decision hyperplane are orthogonal to $\boldsymbol{w}$.

Recall that the Euclidean (or ℓ_2) norm is:

$$||\boldsymbol{w}||_2 = \sqrt{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}} \tag{6}$$

and that a *unit vector* $\boldsymbol{z}$ is defined as a vector with $||z||_2 = 1$. The orthogonal (normalized and signed) distance from an arbitrary point $\boldsymbol{\phi}(\boldsymbol{x})$ to the decision boundary, in space $\mathbb{R}^J$, is given by $r \in \mathbb{R}$, where

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})_{\perp} + r\frac{\boldsymbol{w}}{||\boldsymbol{w}||_2}, \tag{7}$$

and $\boldsymbol{\phi}(\boldsymbol{x})_{\perp}$ is the projection of $\boldsymbol{\phi}(\boldsymbol{x})$ onto the decision boundary, i.e., $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x})_{\perp} + b = 0$. Figure 1c shows what this decomposition looks like, with the origin in the bottom left corner.

Left-multiplying by $\boldsymbol{w}^{\mathsf{T}}$, we have

$$\begin{aligned} \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}) &= \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x})_{\perp} + r\frac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}}{||\boldsymbol{w}||_2} && (8) \\ &= -b + r||\boldsymbol{w}||_2\,. && (9) \end{aligned}$$

We can solve for r to get

$$r = \frac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b}{||\boldsymbol{w}||_2} = \frac{f(\boldsymbol{x}, \boldsymbol{w}, b)}{||\boldsymbol{w}||_2}\,. \tag{10}$$

Note that for $\boldsymbol{\phi}(\boldsymbol{x}) = 0$ this provides $r = b/||\boldsymbol{w}||_2$, so the bias b is the orthogonal distance from the decision boundary to the origin. In all cases, the (absolute) *distance* from the margin is simply $|r|$.

Now, to determine the margin from example $\boldsymbol{x}_n$ to the decision boundary (recalling that the margin is the positive, normalized orthogonal distance if the example is correctly classified, and the negative distance otherwise), we can observe that

- If $\boldsymbol{x}_n$ is correctly classified, then either $f(\boldsymbol{x}_n, \boldsymbol{w}, b) \geq 0$ and $t_n = +1$ or $f(\boldsymbol{x}_n, \boldsymbol{w}, b) < 0$ and $t_n = -1$; either way $t_n \cdot f(\boldsymbol{x}_n, \boldsymbol{w}, b) \geq 0$.
- If $\boldsymbol{x}_n$ is incorrectly classified, then either $f(\boldsymbol{x}_n, \boldsymbol{w}, b) \geq 0$ and $t_n = -1$ or $f(\boldsymbol{x}_n, \boldsymbol{w}, b) < 0$ and $t_n = +1$; either way $t_n \cdot f(\boldsymbol{x}_n, \boldsymbol{w}, b) \leq 0$.

Following from this, the *margin* for example $\boldsymbol{x}_n$, given feature space defined by $\boldsymbol{\phi}$ and parameters $\boldsymbol{w}$, b is exactly

$$\frac{t_n \cdot f(\boldsymbol{x}_n, \boldsymbol{w}, b)}{||\boldsymbol{w}||_2} = \frac{t_n \cdot (\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b)}{||\boldsymbol{w}||_2}\,. \tag{11}$$

Given this, the parameters that maximize the margin arise from:

$$\boldsymbol{w}^\star, b^\star = \underset{\boldsymbol{w},b}{\arg\max} \; \frac{1}{||\boldsymbol{w}||_2} \min_{n\in\{1,\dots,N\}} \left\{ t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) \right\} \tag{12}$$

The min is looking for the *worst* (smallest) margin over the N training data, and the max is trying to find $\boldsymbol{w}$ and b such that this is as large as possible. That is, it is trying to maximize the margin between the decision boundary and the "hardest" datum.

This is the optimization problem that must be solved in training an SVM. As formulated, the problem is hard to solve because it is has a non-linear objective and a large search space given that J may be large. In working to simplify this formulation, we can observe that the normalized, orthogonal distance from point $\boldsymbol{x}_n$ to the decision boundary is invariant to multiplying $\boldsymbol{w}$ and b by any constant $\beta > 0$, since

$$\frac{t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b)}{||\boldsymbol{w}||_2} = \frac{t_n \cdot (\beta\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + \beta b)}{\beta||\boldsymbol{w}||_2} = \frac{t_n \cdot (\widetilde{\boldsymbol{w}}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + \widetilde{w}_0)}{||\widetilde{\boldsymbol{w}}||_2} \tag{13}$$

where $\widetilde{\boldsymbol{w}} = \beta\boldsymbol{w}$ and $\widetilde{w}_0 = \beta b$.

Given this, and given that the data are linearly separable and so there exists a decision boundary with a non-zero (and positive) margin to every example, then we can impose on Equation 12, without any loss in generality, the constraint

$$t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) \geq 1, \quad \forall n \in \{1,\dots,N\} \tag{14}$$

without losing any solutions to optimization problem defined by Equation 12. So we now reformulate the problem as the following optimization:

$$\boldsymbol{w}^\star, b^\star = \underset{\boldsymbol{w},b}{\arg\min} \left\{ \frac{1}{2}||\boldsymbol{w}||_2^2 \right\} \quad \text{s.t.} \quad t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) \geq 1, \quad \forall n \in \{1,\dots,N\}\,. \tag{15}$$

What's going on here? First, notice that if we minimize $\frac{1}{2}||\boldsymbol{w}||_2^2$, we'll also be maximizing $\frac{1}{||\boldsymbol{w}||_2}$. Look back at Equation 12, and imagine that the inside minimum was fixed to some value; in that case we'd be trying to maximize $\frac{1}{||\boldsymbol{w}||_2}$, or alternatively minimize $||\boldsymbol{w}||_2^2$. But the min *wasn't* fixed, right? Well, the argument about β scaling above basically means that we can make this minimum any value we want. In the above constrained problem, we make it 1. So now we can focus entirely on making $||\boldsymbol{w}||_2^2$ small, subject to the margin being at least one for all of the data. The way this will work is that some of the constraints will be tight (binding) in an optimal solution. You can see that this will be true because if the constraints aren't tight, then we could always find a little bit better solution (i.e., with a smaller squared norm for $\boldsymbol{w}$) in which they were.

Let's imagine that datum n that is binding in the optimal solution. (Remember, every datum has a constraint in the problem defined by Equation 15.) If the constraints are tight for example n, then that means

$$t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) = 1\,. \tag{16}$$

So for this example $t_n \cdot f(\boldsymbol{x}_n, \boldsymbol{w}, b) = 1$, and the margin on datum n is exactly $1/||\boldsymbol{w}||_2$, just by substituting into Equation 11. We conclude that formulation in Equation 15 is correct in that a solution will find a $\boldsymbol{w}$ and b that maximize the margin, noting that for all of the other non-binding examples

$$\frac{t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b)}{||\boldsymbol{w}||_2} > \frac{1}{||\boldsymbol{w}||_2}\,. \tag{17}$$

2 Duality

We're already feeling pretty good about things now in that we've managed to come up with a simple constrained optimization problem that maximizes the margin when the data are linearly separable. Moreover, it's a quite well-behaved optimization problem in that it is a convex problem subject to linear constraints. How can we actually solve this though? The feature space $\mathbb{R}^J$ may be very high dimensional.

The answer is that we can employ *duality*. Duality is a general principle that is often used to transform a difficult optimization problem into an equivalent problem that is simpler. The idea is to put the constraints into the objective function, with each constraint associated with a scalar multiplier that becomes a variable in the dual problem and indicates how important the constraint is in the solution. This will take advantage of mathematical tools that we've already been using in the course for handling constraints.

The first step is to introduce *Lagrange multipliers*, $\alpha_1, \ldots, \alpha_N \geq 0$, one for each inequality in Equation 15, i.e., one per datum, to obtain the Lagrangian function:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||_2^2 - \sum_{n=1}^{N} \alpha_n (t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) - 1)\,. \tag{18}$$

We can observe that the formulation in Equation 15 is equivalent to

$$\boldsymbol{w}^\star, b^\star = \arg\min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha}\geq 0} L(\boldsymbol{w}, b, \boldsymbol{\alpha}), \tag{19}$$

since if a constraint is violated in Equation 15 with the choice of $\boldsymbol{w}, b$ then

$$t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) < \; 1 \quad \text{and so} \quad t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) - 1 < \; 0,$$

and α_n can be selected arbitrarily high, providing an unbounded objective value and thus a large penalty. Moreover, in the case that $\boldsymbol{w}, b$ are such that

$$t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) \geq 1 \qquad \text{for all } n,$$

then

$$\max_{\boldsymbol{\alpha}\geq 0} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||_2^2$$

since $\alpha_n = 0$ unless $t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) = 1$, and $\alpha_n (t_n \cdot (\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) - 1) = 0$ for all n.

Moreover, we also have (weak duality)

$$\min_{\boldsymbol{w},b} \; \max_{\boldsymbol{\alpha}\geq 0} \; L(\boldsymbol{w}, b, \boldsymbol{\alpha}) \geq \max_{\boldsymbol{\alpha}\geq 0} \; \min_{\boldsymbol{w},b} \; L(\boldsymbol{w}, b, \boldsymbol{\alpha}) \tag{20}$$

and in fact, because the problem in Equation 19 has a quadratic objective and linear constraints we have (strong duality)

$$\min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha}\geq 0} L(\boldsymbol{w},b,\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}\geq 0} \min_{\boldsymbol{w},b} L(\boldsymbol{w},b,\boldsymbol{\alpha})\,. \tag{21}$$

By this, the order of min and max can be interchanged in (19). This is very helpful, because it now means that for a given $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_N)$, we can solve the first-order conditions for $\boldsymbol{w}$ and b, and achieve a further simplification. Specifically, we can require

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0, \qquad \frac{\partial L}{\partial b} = 0. \tag{22}$$

Solving these, we have

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{n=1}^{N} \alpha_n\, t_n\, \phi_j(\boldsymbol{x}_n) = 0, \quad \forall j \in \{1,\ldots,J\} \tag{23}$$

and require for optimality that:

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n\, t_n\, \boldsymbol{\phi}(\boldsymbol{x}_n)\,. \tag{24}$$

In addition, we have

$$\frac{\partial L}{\partial b} = -\sum_{n=1}^{N} \alpha_n t_n = 0 \tag{25}$$

and require for optimality that:

$$\sum_{n=1}^{N} \alpha_n t_n = 0 \tag{26}$$

Substituting Equations 24 and 26 into $L(\boldsymbol{w},b,\boldsymbol{\alpha})$, and working with the right-hand side of Equation 21, we obtain

$$\arg\max_{\boldsymbol{\alpha}} \tilde{L}(\boldsymbol{\alpha}) \quad \text{s.t.} \quad \alpha_n \geq 0 \quad \forall n \in \{1,\ldots,N\} \quad \text{and} \quad \sum_{n=1}^{N} \alpha_n t_n = 0 \tag{27}$$

where

$$\tilde{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{n'=1}^{N} \alpha_n \alpha_{n'} t_n t_{n'} \mathbf{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) \tag{28}$$

and

$$K(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\top}\boldsymbol{\phi}(\boldsymbol{x}') \tag{29}$$

is a *kernel function.* A kernel function is a scalar product on two vectors mapped by basis function $\boldsymbol{\phi}$ into a (possibly higher dimensional) feature space.

This we can solve! In particular, it is a *quadratic program* because it has a quadratic objective function (terms in the objective are zero, first or second-degree nomials) and linear inequality constraints. The number of decision variables is exactly the number of examples in the training data. One typical solution technique is to solve via "iterative conjugate gradient methods", but the particular details of this technique are outside the scope of this course.

The search space in the original formulation in Equation 12 has been transformed from feature space to example space, and now has dimension of the number of data examples. This may seem like a loss but is in fact a win because the potential high dimensionality of a feature space (as described through mapping $\boldsymbol{\phi}$) is now all handled through the kernel function.

When we solve the program we obtain an optimal vector $\boldsymbol{\alpha}^\star \in \mathbb{R}^J_+$. With this, we can substitute into Eq. (24) to obtain

$$\boldsymbol{w}^\star = \sum_{n=1}^{N} \alpha_n^\star \, t_n \, \boldsymbol{\phi}(\boldsymbol{x}_n) \tag{30}$$

which together with $f(\boldsymbol{x}, \boldsymbol{w}^\star, b^\star) = (\boldsymbol{w}^\star)^\top \boldsymbol{\phi}(\boldsymbol{x}) + b^\star$, where $b^\star$ is also optimized, yields

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{w}^\star, b^\star) &= \sum_{n=1}^{N} \alpha_n^\star \, t_n \, \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}) + b^\star & (31)\\ &= \sum_{n=1}^{N} \alpha_n^\star \, t_n \, K(\boldsymbol{x}_n, \boldsymbol{x}) + b^\star. & (32) \end{aligned}$$

The support vectors for a trained SVM classifier are those examples that define the margin of the hypothesis on the training data, that is, the examples closest to the decision boundary in the high dimensional space. Let $\mathcal{S} = \{n : \alpha_n > 0, n \in \{1, \ldots, N\}\}$ denote the index set of *support vectors* in the solution to formulation (27). The prediction $f(\boldsymbol{x}, \boldsymbol{w}, b)$ for example $\boldsymbol{x}$ depends only on the weights associated with these examples; i.e., the examples that define the margin of the classifier. These examples are said to "support" the margin.

This comes from interpreting the Karush-Kuhn-Tucker (KKT) conditions on the solution, which require

$$\alpha_n \left(t_n \, f(\boldsymbol{x}_n, \boldsymbol{w}, b) - 1\right) = 0, \quad \forall n \in \{1, \ldots, N\} \tag{33}$$

in an optimal solution, and so $\alpha_n > 0 \;\Rightarrow\; t_n \, f(\boldsymbol{x}_n, \boldsymbol{w}, b) = 1$. We see that for the support vectors, the classifier provides $t_n \, f(\boldsymbol{x}_n, \boldsymbol{w}, b) = 1$, and so these are the examples for which the constraints are tight in formulation Equation 15 and thus define the margin.

Looking at Figure 1a, the support vectors are the examples for which there is a line between them and the decision boundary. We can classify any example using only the support vectors:

$$y(\boldsymbol{x}, \boldsymbol{w}^\star, b^\star) = \operatorname{sign}\left\{ \sum_{n \in \mathcal{S}} \alpha_n^\star \, t_n \, K(\boldsymbol{x}_n, \boldsymbol{x}) + b^\star \right\} \tag{34}$$

where $\operatorname{sign}(z)$ returns $+1$ if $z \geq 0$ and -1 otherwise. The sum is over support vectors. Note that this requires computing the kernel of each support vector and the instance to be classified.

Essentially, the classification process compares a new instance $\boldsymbol{x}$ with each of the support vectors. The scalar product $\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}_n) = K(\boldsymbol{x}_n, \boldsymbol{x})$ measures how similar the new instance $\boldsymbol{x}$ is to the training instance $\boldsymbol{x}_n$. Weight $\alpha_n^\star$ measures the contribution of support vector $\boldsymbol{x}_n$, i.e. $\alpha_n^\star$ measures how important the given support vector is. We multiply by t_n to take into account the influence of the given support vector on the classification.

Finally, we need to solve for $b^\star$. For this, take any example $n \in \mathcal{S}$ and note that $t_n\, y(\boldsymbol{x}_n, \boldsymbol{w}, b) = 1$ (since the support vectors are those for which Eq. 14 is tight). From this,

$$t_n \cdot \sum_{n' \in \mathcal{S}} (\alpha_{n'}^\star t_{n'} K(\boldsymbol{x}_{n'}, \boldsymbol{x}_n) + b^\star) = 1\,. \tag{35}$$

Multiplying both sides by t_n (which is $t_n \in \{-1, +1\}$, so that $t_n^2 = 1$) then we have

$$b^\star = t_n - \sum_{n' \in \mathcal{S}} \alpha_{n'}^\star\, t_{n'} K(\boldsymbol{x}_{n'}, \boldsymbol{x}_n)\,. \tag{36}$$

Note: in practice, given that the optimization problem is solved to some degree of accuracy, it is standard to set $b^\star$ to the average such value computed over all examples in the support:

$$b^\star = \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{n' \in \mathcal{S}} \alpha_{n'}\, t_{n'}\, K(\boldsymbol{x}_{n'}, \boldsymbol{x}_n) \right)\,. \tag{37}$$

3 Kernels

The wonderful thing that happened in formulating optimization (27) is that the inputs $\boldsymbol{x}$ only enter into the formulation via kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$. This allows for a very high dimensional feature space to be handled implicitly and without first computing a feature vector and then taking the scalar product.

In working with SVMs, we are in fact seeking classifiers that provide linear separation of data in a higher dimensional space. Kernels are a clever way of mapping data into a higher dimensional feature space. First, notice the form of the dual problem. It contains instances of the data multiplied with each other through an inner product. This inner product between instances allows us to use a kernel function.

Definition 3 *A* kernel *is a function $K(\cdot, \cdot)$ such that $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}')$, where $\boldsymbol{\phi}$ is a mapping from a space $\mathcal{X}$ to a feature space $\mathbb{R}^J$.*

Figure 2 illustrates this definition. $\boldsymbol{\phi}$ maps examples in the input space $\mathcal{X}$ into the feature space $\mathbb{R}^J$. So $\boldsymbol{\phi}$ takes $\boldsymbol{x}$ and produces $\boldsymbol{\phi}(\boldsymbol{x})$, and similarly takes $\boldsymbol{x}'$ and produces $\boldsymbol{\phi}(\boldsymbol{x}')$.

Now we can compute the inner (scalar) product in the feature space. But with a kernel function, we don't actually have to do the mapping and compute the inner product in $\mathbb{R}^J$. We can compute the kernel function $K(\cdot, \cdot)$ directly on examples in $\mathcal{X}$.

When we compute $K(\boldsymbol{x}, \boldsymbol{x}')$, that immediately tells us the scalar product between $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x}')$ – but we don't actually have to compute $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x}')$ to compute $K(\boldsymbol{x}, \boldsymbol{x}')$. This is the important property of kernels – we can map inputs into a higher dimensional feature space, without doing all the computations in that higher dimensional space.

Here are some examples of kernel functions:

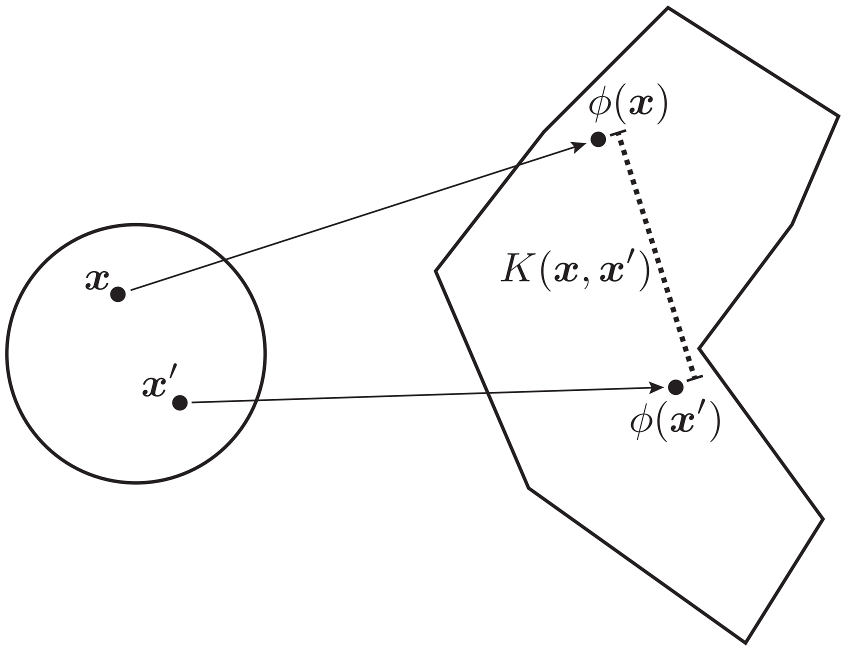


Figure 2: Illustration of a kernel function.

Example 1

$$K(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}^\top \boldsymbol{z} \tag{38}$$

This is the trivial kernel function. The input space and the feature space are the same, and the kernel simply computes the scalar product of the two vectors.

Example 2 *Consider a mapping from $\mathcal{X} = \mathbb{R}^D$ to $\mathbb{R}^J$ where $J = D^2$. $\boldsymbol{\phi}(\boldsymbol{x})$ maps a point $\boldsymbol{x} = (x_1, ..., x_D)$ to $x_1^2, x_1x_2, x_1x_3, ..., x_1x_D, x_2x_1, ..., x_2x_D, ..., x_Dx_{D-1}, x_D^2$.*

$$\begin{aligned}
K(\boldsymbol{x}, \boldsymbol{z}) &= (\boldsymbol{x}^\top \boldsymbol{z})^2 && (39)\\
&= \left(\sum_{d=1}^{D} x_d z_d\right)^2 && (40)\\
&= \left(\sum_{d=1}^{D} x_d z_d\right)\left(\sum_{d=1}^{D} x_d z_d\right) && (41)\\
&= \sum_{d=1}^{D}\sum_{d'=1}^{D} x_d x_{d'} z_d z_{d'} && (42)\\
&= \sum_{d=1}^{D}\sum_{d'=1}^{D} (x_d z_d)(x_{d'} z_{d'}) && (43)\\
&= \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{z}) && (44)
\end{aligned}$$

We see here the advantage of a kernel. Instead of having to compute the entire feature map (which is quadratic in this case) and then computing the scalar product directly, you only have to compute the kernel function, which is $(\boldsymbol{x}^\top \boldsymbol{x}')^2$, and takes linear time.

Example 3 *The previous example generalizes to polynomial kernel functions:*

$$K(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^{\mathsf{T}}\boldsymbol{z} + c)^M \tag{45}$$

which consist of features comprised of all nomials up to the Mth degree (and are therefore exponential in M.)

Example 4 *The Gaussian kernel is*

$$K(\boldsymbol{x}, \boldsymbol{z}) = \exp\left\{-\frac{1}{2\sigma^2}||\boldsymbol{x} - \boldsymbol{z}||_2^2\right\} \tag{46}$$

While this doesn't look much like an inner product, it is a legal kernel. It compares the distances between two examples, with the importance of $\boldsymbol{z}$ *to* $\boldsymbol{x}$ *decaying exponentially with the squared distance from* $\boldsymbol{z}$ *to* $\boldsymbol{x}$*. In fact, the corresponding feature space for this kernel has an infinite dimension!*

A key point is that we can describe this kernel without describing the inner product space explicitly. When we come to computing with this kernel, we will be comparing training instances and computing their distance from each other.

The bottom line is that by adopting a non-trivial kernel function, SVMs can be used to learn max-margin classifiers in some high-dimensional space. In fact, one can also just ask directly whether a $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a valid kernel, in the sense that there exists some feature mapping $\boldsymbol{\phi}$ so that $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}')$ for all $\boldsymbol{x}, \boldsymbol{x}'$. The result (due to Mercer), and out of scope of this class, is that a *Kernel matrix*, constructed by making entry $K_{n,n'} = K(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$ for example n and n' from a set of examples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ must be *symmetric positive semi-definite*. This is the requirement that $\boldsymbol{z}^{\mathsf{T}}K\boldsymbol{z} \geq 0, \forall \boldsymbol{z} \in \mathbb{R}^N$.[2] Note also that positive semi-definite matrices have all nonnegative eigenvalues.

In addition, if $K(\cdot,\cdot), K'(\cdot,\cdot)$ are kernels then $cK(\cdot,\cdot)$ is a kernel for $c > 0$, and $aK(\cdot,\cdot) + bK'(\cdot,\cdot)$ is a kernel for $a, b > 0$. So, can make complex kernels from simple ones.

4 Properties of SVMs

- Accurate classification.
- With appropriate kernel function, can express very complex decision boundaries.
- No problem with local minima: quadratic objective function and thus convex minimization problem.
- Decision boundary directly represented as set of support vectors: independent of all other non-support vectors. Fairly succinct representation of a hypothesis and thus reasonable classification speed.
- Very slow training. This is particularly a problem for large training sets. The objective function to the quadratic program (27) contains the kernel on all pairs of instances.
- Exactly separating examples in a high-dimensional space can lead to strange looking decision boundaries. This can lead to over-fitting. Need to use regularization (see below).

[2]Positive-definite vs. positive semi-definite regards whether or not this is a strict inequality.

5 Extensions

We briefly discuss some extensions to the basic approach introduced so far.

Non separable data, regularization and model selection. What if the training data are not linearly separable, even with a high dimensional feature space? What if the data are noisy, and so exactly fitting the training data is not the best way to generalize?

The SVM method can be easily extended to act as a "soft-margin" method in which some examples are allowed to be misclassified. In particular, the new (primal) training problem is

$$\begin{aligned} \boldsymbol{w}^\star, b^\star = \arg\min_{\boldsymbol{w},b} &\left\{ \frac{1}{2}||\boldsymbol{w}||_2^2 + C\sum_{n=1}^{N} \xi_n \right\} \qquad (47) \\ \text{s.t.} \quad & t_n \cdot (\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b) \geq 1 - \xi_n, \quad \forall n \in \{1, \ldots, N\} \\ & \xi_n \geq 0, \quad \forall n \in \{1, \ldots, N\} \end{aligned}$$

for some $C > 0$. We can actually view the first term in the objective, which was derived as the component that provides for a large margin, as providing "regularization" in that it is preferring simpler weights.

The second component can be interpreted as an upper-bound on the empirical training error (set to 1 if an example is incorrectly classified, 0 otherwise). To see this, consider a positive example with $t_n = 1$. Then, if $\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b = 0$, and the point is about to be misclassified, we have $\xi_n = 1$. For smaller $\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b < 0$ then ξ_n increases. For larger $\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b > 0$ then ξ_n decreases and is zero for $y' \geq 1$. This is a convex upper bound on the error.

In this new training problem, we adopt $C > 0$ as the regularization parameter, with a larger C placing more emphasis on reducing training error and thus less regularization. Parameter C can be tuned through cross-validation.

What is very nice about this alternate "soft-margin" formulation is that the dual formulation of Equation 27 is modified in a very small way: the constraint on $\alpha_n \geq 0$ just becomes

$$0 \leq \alpha_n \leq C, \quad \forall n \in \{1, \ldots, N\} \qquad (48)$$

and everything follows (except for a slight change in the formula for $b^\star$.) Another practical issue that can occur is selecting the appropriate kernel. This can be determined through the use of (cross-)validation, in order to avoid overfitting.

Regression. SVMs extend easily to regression problems. The function becomes

$$y(\boldsymbol{x}, \boldsymbol{w}, b) = \sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) K(\boldsymbol{x}_n, \boldsymbol{x}) + b, \qquad (49)$$

where $\alpha_n, \hat{\alpha}_n$ are the weights on support vectors on either side of the hypothesis, outside of an "ϵ-tube" of training data that is accurately fit.

A parameter $\epsilon > 0$, which is part of the training problem, is be determined through cross-validation, with smaller $\epsilon > 0$ leading to more complex hypotheses.