9.520: Statistical Learning Theory and Applications February 17th, 2010

Several Views of Support Vector Machines

Lecturer: Charlie Frogner *Scribe: Nicholas Edelman, Matt Miller*

1 Introduction

Support vector machines (SVMs) are a popular learning tool for designing classifier functions. For the purposes of this class, we are only concerned with **binary classification**. Like regularized least squares (RLS) from the previous lecture, SVMs determine the classification function by solving Tikhonov regularization learning problem.

2 Setting up the SVM problem

We are given n examples $(x_1, y_1), \dots, (x_n, y_n)$, with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ for all i. We solve the Tikhonov regularization learning problem to find the classifier function:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(x_i)) + \lambda ||f||_{\mathcal{H}}^2.$$

For our classification function $f(x)$, negative values of $f(x)$ correspond to one class, and nonnegative values of $f(x)$ correspond to the other class. The final classification is thus:

$$\text{sign}(f(x)).$$

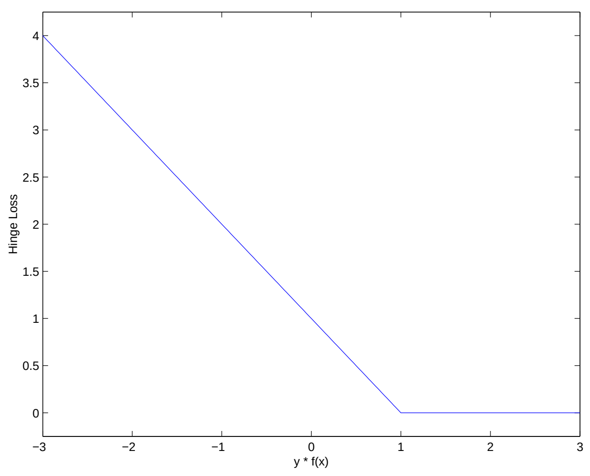
2.1 Designing an appropriate loss function $V(y_i, f(x_i))$

In the case where the classification function, $f(x)$, and the training point, y_i, have the same sign, we have classified correctly, $y_i f(x_i) \geq 0$, and want the loss function V to be small or zero. When there is an incorrect classification, $y_i f(x_i) < 0$, we want the loss function V to be larger. When first considering this problem, a natural initial choice is the step function.

$$V(y_i, f(x_i)) = \begin{cases} 1 & \text{for } y_i f(x_i) < 0 \\ 0 & \text{for } y_i f(x_i) \geq 0 \end{cases}.$$

However, the step function has a few problems making it not the ideal loss function for binary classification. For one, the derivative is undefined at $x = 0$, so thus the step function is not differentiable. The primary problem is the step function is not convex. Non-convex functions are difficult to use in minimization problems. On the other hand, for convex functions, we can leverage the field of convex optimization to help solve the minimization problem. To address the deficiencies of the step function, the classical SVM loss function is the hinge loss (Although the hinge loss is convex, it is not also not differentiable. We later introduce slack variables to deal with this shortcoming.):

$$V(f(x), y) \equiv (1 - yf(x))_+, \text{ where } (k)_+ \equiv \max(k, 0).$$



Using the hinge loss, the loss is linear for $yf(x) < 1$ and zero elsewhere. The loss is designed to be asymmetric: incorrect classifications, $y_if(x_i) < 0$, are assigned linearly increasing losses as $y_if(x_i)$ decreases, and any correct classification with $y_if(x_i) \geq 1$ is always zero loss. In addition, the loss function assigns a positive loss to correct classification for $0 < y_if(x_i) < 1$. The goal of this asymmetry is to hopefully create a classifier that not only classifies correctly, but also classifies most x_i inputs with at least a value of $y_if(x_i) \geq 1$. Assigning a linearly loss to anything below 1 instead of just below 0 reduces the likelihood that new data will perturb parts of the solution into incorrect classification.

Given the hinge loss function, the Tikhonov regularization expression becomes:

$$\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n}(1 - y_if(x_i))_+ + \lambda||f||^2_{\mathcal{H}}.$$

3 Solving SVM

To solve the SVM regularization problem, we need to make our constraint problem more tractable. We introduce the use of slack variables to make the problem differentiable. We then formulate the SVM regularization as the classic *primal* problem and discover it is easier to optimize when we convert the *primal* into the *dual* form.

3.1 Slack Variables

The hinge loss is not differentiable due to the kink in V at $y_if(x_i) = 1$. We introduce slack variables ξ_i to make the objective function differentiable and thus easier to optimize. For each point in the training set, we introduce one ξ_i to replace $(1 - y_if(x_i))$. For each ξ_i, we require $\xi_i \geq (1 - y_if(x_i))_+$. The new problem statement becomes:

$$\begin{array}{rll} \min\limits_{f\in\mathcal{H}} & \frac{1}{n}\sum_{i=1}^{n}\xi_i + \lambda||f||^2_{\mathcal{H}} & \\ \text{subject to :} & y_if(x_i) \geq 1 - \xi_i & i = 1,\dots,n \\ & \xi_i \geq 0 & i = 1,\dots,n. \end{array}$$

3.2 Arriving at the Primal SVM

By the representer theorem, we know our optimal classifier function, the minimizer of the Tikhonov functional, can be written as follows:

$$f^*(x) = \sum_{i=1}^{n} c_i K(x, x_i).$$

Substituting the $f^*(x)$ for f, we arrive at a constrained quadratic programming problem:

$$\begin{array}{lll} \min\limits_{c\in\mathbb{R}^n, \xi\in\mathbb{R}^n} & \frac{1}{n}\sum_{i=1}^{n}\xi_i + \lambda c^T K c & \\ \text{subject to :} & y_i \sum_{j=1}^{n} c_j K(x_i, x_j) \geq 1 - \xi_i & i = 1, \ldots, n \\ & \xi_i \geq 0 & i = 1, \ldots, n. \end{array}$$

To make our problem appear as it does in the SVM literature, we need to make two modifications to our problem statement:

1. Add a bias term b.
2. Use parameter C instead of λ. To change to C notation, we have $C = \frac{1}{2\lambda n}$.

Applying these two modifications, we arrive at the "primal" SVM:

$$\begin{array}{lll} \min\limits_{c\in\mathbb{R}^n, b\in\mathbb{R}, \xi\in\mathbb{R}^n} & C\sum_{i=1}^{n}\xi_i + \frac{1}{2} c^T K c & \\ \text{subject to :} & y_i (\sum_{j=1}^{n} c_j K(x_i, x_j) + b) \geq 1 - \xi_i & i = 1, \ldots, n \\ & \xi_i \geq 0 & i = 1, \ldots, n. \end{array}$$

3.3 Optimizing the Primal SVM

Now that we have formulated the SVM problem into a *primal* problem, the approach to solving this constrained optimzation problem is as follows:

- *Lagrangian* - formulate from the primal as in Lagrange multipliers
- *Dual* - associate one dual variable to each primal constraint in the Lagrangian. Solve the dual problem. As we will show, it is easier to solve the dual rather than the primal problem.

We associate one slack variable for each constraint. For each constraint, $\xi_i \geq 0$, we associate a dual variable ζ_i. For each constraint, $y_i(\sum_{j=1}^{n} c_j K(x_i, x_j) + b) \geq 1 - \xi_i$, we associate a dual variable α_i. The Lagrangian is thus:

$$\begin{aligned} L(c, \xi, b, \alpha, \zeta) &= C\sum_{i=1}^{n}\xi_i + c^T K c \\ &\quad - \sum_{i=1}^{n}\alpha_i (y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} - 1 + \xi_i) \\ &\quad - \sum_{i=1}^{n}\zeta_i \xi_i. \end{aligned}$$

The dual problem is to find the maximum, with respect to the dual variables, of the infimum value of the Lagrangian with respect to the primal variables. Formally, we can write the dual problem as:

$$\arg\max_{\alpha,\zeta}\ \inf_{c,\xi,b} L(c,\xi,b,\alpha,\zeta).$$

To construct the dual problem, we need to determine the optimal c, ξ, and b in terms of the dual variables. We achieve this by differentiating the constraints with respect to the primal variables.

$$\begin{aligned}
\frac{\partial L}{\partial b} = 0 &\implies \sum_{i=1}^{n} \alpha_i y_i = 0 \\
\frac{\partial L}{\partial \xi_i} = 0 &\implies C - \alpha_i - \zeta_i = 0 \\
&\implies 0 \le \alpha_i \le C \\
\frac{\partial L}{\partial c} = 0 &\implies c_i = \alpha_i y_i
\end{aligned}$$

Substituting $\alpha_i y_i$ for c_i and applying the new constraints, the dual problem becomes:

$$\begin{array}{rll}
\max\limits_{\alpha\in\mathbb{R}^{\mathbf{n}}} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\alpha^T Q \alpha & \\
\text{subject to :} & \sum_{i=1}^{n} y_i \alpha_i = 0 & \\
& 0 \le \alpha_i \le C & i = 1, \ldots, n.
\end{array}$$

where $Q = y^T K y$.

This dual form is much easier to solve than the primal. In the primal, we must minimize over c, b and ξ, with two inequality constraints per training point. In the dual, we must maximize over α with one box constraint and one simple inequality per point. In practice, the problem is typically solved using the dual form for exactly this reason. Once the dual is maximized, the resulting α can be used to calculate both c and b.

$$\begin{aligned}
c_i &= \alpha_i y_i, \\
b &= y_i - \sum_{j=1}^{n} c_j K(x_i, x_j).
\end{aligned}$$

We showed earlier how the condition for c_i arises naturally out of $\frac{\partial L}{\partial c} = 0$. In the next section, we prove the condition for b.

3.4 Optimality Conditions

We have formulated the SVM as a quadratic programming problem. We can therefore use results from the field of optimization to derive certain properties of an optimal SVM solution. Specifically, the Karush-Kuhn-Tucker (KKT) conditions are necessary conditions for an optimal solution to any non-linear programming problem. They are sufficient conditions when the primal objective and inequality constraints are convex and continuously differentiable, and each equality constraint is an affine function. This holds for the SVM problem, so the KKT conditions are both necessary and sufficient conditions for any optimal SVM solution.

The KKT conditions can be grouped into four categories: stationarity, primal feasibility, dual feasibility and complementary slackness. Stationarity requires that the gradient of the Lagrangian be zero. Primal and dual feasibility say that the solution must satisfy both the primal and dual constraints (i.e. any optimal solution must be a feasible one). Complementary slackness relates the value of each Lagrangian multiplier with its corresponding constraint. Recall that the variables α_i and ζ_i were introduced to incorporate the primal constraints into the dual objective.

$$\begin{aligned}
\alpha_i &\rightarrow y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} - 1 + \xi_i \geq 0, \\
\zeta_i &\rightarrow \xi_i \geq 0.
\end{aligned}$$

The complementary slackness condition states that either the primal constraint is satisfied with equality, or its corresponding Lagrangian multiplier is zero. More formally, if c, ξ, b, α and ζ are optimal solutions to the primal and dual, then

$$\begin{aligned}
\alpha_i(y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} - 1 + \xi_i) &= 0, \\
\zeta_i \xi_i &= 0.
\end{aligned}$$

Therefore, for any training point x_i, either $\alpha_i = 0$ or $y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} - 1 + \xi_i = 0$. Since $\xi_i \geq 0$, the latter case occurs when $y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} \leq 1$ (so the point is not classified 'as correctly' as we would like) and $\alpha_i = 0$ when $y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} > 1$ (by complementary slackness). We showed above that, at the optimum, $c_i = y_i \alpha_i$, so when $\alpha_i = 0$, the coefficient in the solution that corresponds to the ith training point will be zero – the solution is said to be "sparse." The points x_i for which $\alpha_i > 0$ are called the "support vectors," which gives the SVM its name. The role of the support vectors will be discussed shortly. But first, we can add the stationarity and feasibility constraints to obtain all of the KKT conditions for the SVM problem.

$$\begin{aligned}
\sum_{j=1}^{n} c_j K(x_i, x_j) - \sum_{j=1}^{n} y_i \alpha_j K(x_i, x_j) = 0 &\qquad i = 1, \ldots, n \\
\sum_{i=1}^{n} \alpha_i y_i = 0 & \\
C - \alpha_i - \zeta_i = 0 &\qquad i = 1, \ldots, n \\
y_i(\sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j) + b) - 1 + \xi_i \geq 0 &\qquad i = 1, \ldots, n \\
\alpha_i[y_i(\sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j) + b) - 1 + \xi_i] = 0 &\qquad i = 1, \ldots, n \\
\zeta_i \xi_i = 0 &\qquad i = 1, \ldots, n \\
\xi_i, \alpha_i, \zeta_i \geq 0 &\qquad i = 1, \ldots, n.
\end{aligned}$$

Using these optimality conditions, we can calculate the bias term b. Suppose that there exists some i satisfying $0 < \alpha_i < C$ (this happens in practice).

$$\begin{aligned}
\alpha_i < C &\implies \zeta_i > 0 \\
&\implies \xi_i = 0 \\
&\implies y_i(\sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j) + b) - 1 = 0 \\
&\implies b = y_i - \sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j).
\end{aligned}$$

Thus, if we solve the dual problem for an optimal α, we can go back and solve for b.

3.5 Support Vectors

We showed earlier that $c_i = y_i \alpha_i$. Substituting for c_i, the optimal classification function may be written as:

$$f(x) = \sum_{i=1}^{n} y_i \alpha_i K(x, x_i) + b$$

.

Notice that, in order to classify a novel point x, we need only calculate the value of $K(x, x_i)$ when $\alpha_i > 0$. Therefore, once an optimal α is learned, all of the training points x_i where $\alpha_i = 0$ can be discarded. New points can be classified using only those training points with positive Lagrangian multipliers. This is why a sparse solution is desirable. If the classification function is expressed in terms of very few training points, it reduces the required memory and computation for classification.

But why is this possible? Why can certain points be ignored for purposes of classification? Suppose that for some training point x_i, the loss $V(y_i, f(x_i))$ is zero, so

$$\begin{array}{rc}
 & V(y_i, f(x_i)) = 0 \\
\Rightarrow & y_i f(x_i) \geq 1 \\
\Rightarrow & y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} \geq 1 \\
\Rightarrow & y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} - 1 + \xi_i \geq 0
\end{array}$$

with the last step following because $\xi_i \geq 0$. If $y_i f(x_i) > 1$ (so excluding the case that $y_i f(x_i) = 1$), then

$$\begin{array}{rc}
 & y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} - 1 + \xi_i > 0 \\
\Rightarrow & \alpha_i = 0
\end{array}$$

($\alpha_i = 0$ by complementary slackness.)

This holds in the reverse direction, as well: $\alpha_i = 0 \Rightarrow V(y_i, f(x_i)) = 0$. First, assuming $\alpha_i = 0$:

$$\begin{aligned}
\alpha_i = 0 \Rightarrow &\zeta_i = C \\
\Rightarrow &\xi_i = 0
\end{aligned}$$

($\zeta_i = C$ because at the optimum $C - \alpha_i - \zeta_i = 0$, and $\xi_i = 0$ by complementary slackness.)

We defined (way back in the primal problem) that $y_i\{\sum_{j=1}^n c_j K(x_i, x_j) + b\} - 1 + \xi_i \geq 0$. Plugging $\xi_i = 0$ into this constraint:

$$\begin{aligned}
&\xi_i = 0 \\
&\Rightarrow y_i\{\sum_{j=1}^{n} c_j K(x_i, x_j) + b\} - 1 \geq 0 \\
&\Rightarrow y_i f(x_i) \geq 1 \\
&\Rightarrow V(y_i, f(x_i)) = 0
\end{aligned}$$

.

So we see that if the training point x_i incurs no loss given the optimal classification function f, then the corresponding α_i is zero – except in the case that $y_i f(x_i) = 1$, where α_i is constrained only to be between 0 and C – and vice versa: if α_i is zero, then the optimal classification function incurs no loss at the point x_i. Since the classification function can be written

$$\begin{aligned}
f(x) &= \text{sign}\left(\sum_{j=1}^{n} c_j K(x, x_j) + b\right) \\
&= \text{sign}\left(\sum_{j=1}^{n} y_j \alpha_j K(x, x_j) + b\right)
\end{aligned}$$

the *points for which* $\alpha_j = 0$ *don't appear in the solution function* – it's based only on those points x_j for which $\alpha_j > 0$. Intuitively, the classification function defines a decision boundary based on a weighted sum over the training points. Those for which the weight is 0 are far enough inside the decision boundary defined by other points to incur no loss (recall that the hinge loss penalizes points that are near the decision boundary, even if they are classified correctly). Those points for which $\alpha_i > 0$ define the decision boundary, and are therefore the only points required for classification. These are the support vectors.

3.6 Reduced Optimality Conditions

It is convenient to rewrite the KKT conditions in a "reduced" form that expresses them more clearly. Specifically, we would like to define them in terms of our classification function $f(x)$. We can write the optimality conditions in terms of the values of α_i, ξ_i and ζ_i given $y_i f(x_i)$ and vice versa. For example, it is easy to show that

$$\begin{aligned}
y_i f(x_i) < 1 &\implies \xi_i > 0 \\
&\implies \zeta_i = 0 \\
&\implies \alpha_i = C
\end{aligned}$$

and conversely if $\alpha_i = C$

$$\begin{aligned}
\alpha_i = C &\implies y_i f(x_i) - 1 + \xi_i = 0 \\
&\implies y_i f(x_i) \leq 1.
\end{aligned}$$

Proceeding accordingly, we can rewrite all of the optimality conditions as relations of this sort (we'll avoid proving them here.)

$$\begin{aligned}
\alpha_i = 0 &\implies y_i f(x_i) \geq 1 \\
0 < \alpha_i < C &\implies y_i f(x_i) = 1 \\
\alpha_i = C &\impliedby y_i f(x_i) < 1 \\
\\
\alpha_i = 0 &\impliedby y_i f(x_i) > 1 \\
\alpha_i = C &\implies y_i f(x_i) \leq 1
\end{aligned}$$

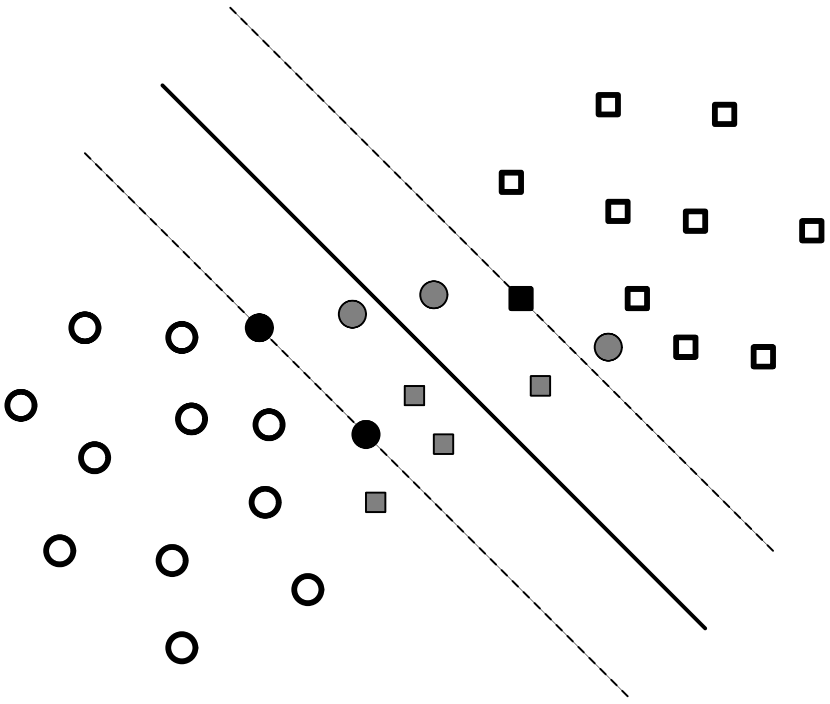


Figure 1: A linear decision boundary on the training set. The support vectors are darkened and filled.

4 The Geometric Approach

So far we have presented the SVM as a special case of Tikhonov Regularization. The solution of the resulting quadratic programming problem is a set of (possibly sparse) α_i's, which specify the weighted sum that becomes the classification function $f(x)$. While this formulation is elegant and rigorously satisfying, it is not how the SVM was originally conceived or derived. Instead, it started with the concepts of separating hyperplanes, and margin.

Given a training set $S \subseteq \mathbb{R}^n \times \{-1, 1\}$ the separating hyperplane is an d-dimensional hyperplane that separates the positive and negative examples in S. We can denote the hyperplane with a weight vector $w \in \mathbb{R}^d$, and the corresponding classification function $f(x) = \text{sign } (w \cdot x)$.

This is the definition of a linear classifier. In high dimensional input spaces, there are often many hyperplanes that can separate a given training set. Many of the possible separating hyperplanes might overfit the training data, and generalize poorly. To compensate, we pick the hyperplane that maximizes the margin, where the margin is defined as the distance from the separating hyperplane to the closest point in the training set. In other words, select the boundary that separates the positive and negative examples by the largest distance. The hope is that this hyperplane will do well at classifying future examples, because, in some sense, it separates the training data in the best possible way.

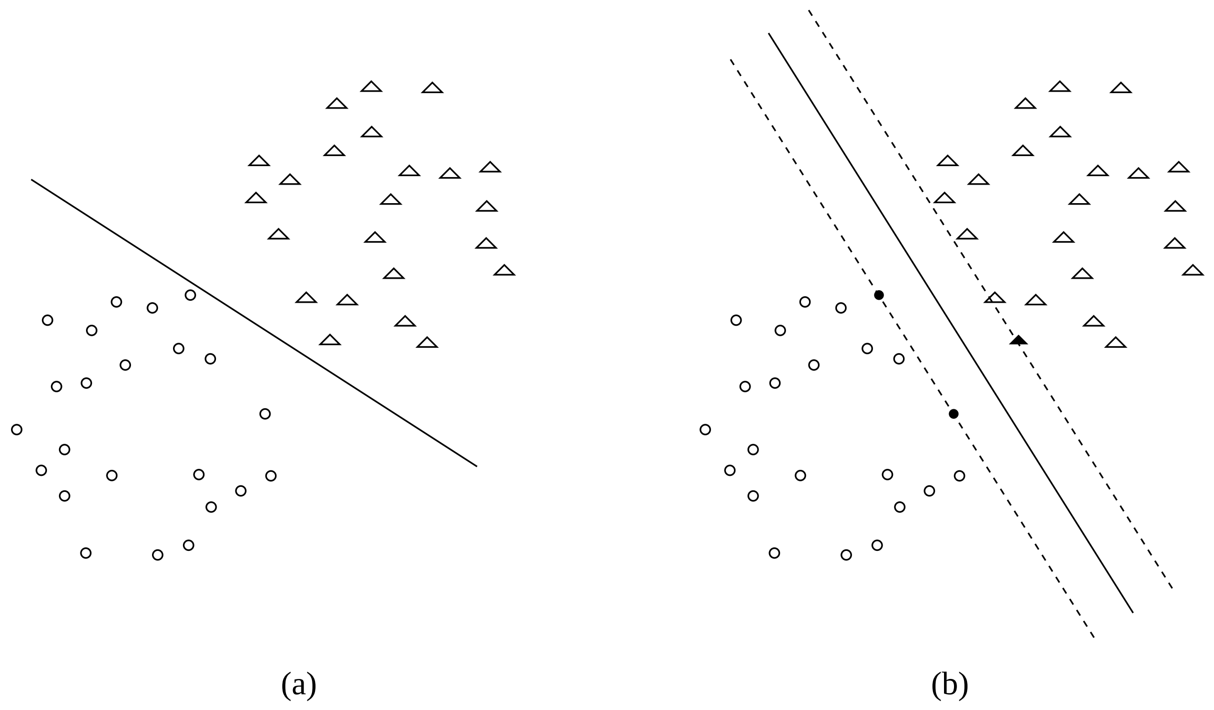


Figure 2: An illustration of the max-margin principle. The boundary is chosen that separates the data by the largest distance on both sides.

Let x be the training datapoint closest to w, and x^w be the point on the hyperplane closest to x. Then, maximizing the margin becomes equivalent to maximizing $\|x - x^w\|$. We can do a little bit of math to construct an optimization problem to do just this.

For some k (assume $k > 0$)

$$\begin{aligned} w \cdot x &= k \\ w \cdot x^w &= 0 \\ \implies \quad w \cdot (x - x^w) &= k \end{aligned}$$

Since $x - x^w$ is parallel to normal vector w we can proceed:

$$\begin{aligned} w \cdot (x - x^w) &= w \cdot \left(\frac{||x - x^w||}{||w||} w \right) \\ &= ||w||^2 \frac{||x - x^w||}{||w||} \\ &= ||w|| \; ||x - x^w|| \\ &\Longrightarrow ||w|| \; ||(x - x^w)|| = k \\ &\Longrightarrow ||x - x^w|| = \frac{k}{||w||}. \end{aligned}$$

So here we have derived that the margin $\|x - x^w\|$ is actually equal to some constant k divided by the norm of w. Therefore, we can maximize the margin by simply minimizing $\|w\|$ or, more conveniently, $\|w\|^2$. Essentially, the numerical value of the margin is unimportant. If we assume that the points in our training set are linearly separable, then we can define the max-margin hyperplane as the one that solves

$$\begin{array}{rl} \min\limits_{w \in \mathbb{R}^n} & ||w||^2 \\ \text{subject to :} & y_i(w \cdot x) \geq 1 \quad i = 1, \ldots, n. \end{array}$$

Of course, this is still a little bit too simple. We would like to be able to learn hyperplanes that don't pass through the origin. We can do this by adding a bias term b to the classification function

$$f(x) = \text{sign } (w \cdot x + b).$$

Also, note that *we've assumed that the classes are perfectly separated by a linear decision boundary.* Most datasets are not linearly separable. So we must add slack variables ξ_i for each training point. These allow a point to be misclassified, but they incur a penalty in the objective function. With these two additions, the primal form of the SVM becomes the familiar

$$\begin{array}{rll} \min\limits_{w \in \mathbb{R}^n, \xi \in \mathbb{R}^n, b \in \mathbb{R}} & C \sum_{i=1}^n \xi_i + \frac{1}{2} ||w||^2 & \\ \text{subject to :} & y_i(w \cdot x + b) \geq 1 - \xi_i & i = 1, \ldots, n \\ & \xi_i \geq 0 & i = 1, \ldots, n. \end{array}$$

The only thing this formulation is missing is the kernel function. It is easily added, using the kernel "trick". Remember that, when we write down the dual form, the points $\{x_i\}$ appear only in the quadratic penalty term $\alpha^T Q \alpha$: in the linear case we have $Q = y^T (X^T X) y$, i.e. $[Q]_{ij} = y^T \langle x_i, x_j \rangle y$, so we can replace these inner products between the input points with the kernel function evaluated at the same points. The penalty term becomes $\alpha^T \text{diag} Y K \text{diag} Y \alpha$, just as we derived before.

It is interesting to note that, historically, the kernel "trick" was added at the end of the derivation. Conceptually, the SVM was derived as an optimal linear classifier. However, it was only linear because the dot product was used to define the similarity metric between training points. The kernel was substituted in order to take the dot product in a (possibly infinite-dimensional) feature space, with a linear decision boundary in this feature space corresponding to a nonlinear decision boundary in the original input space.

This formulation also makes explicit the idea of a "support vector." Recall that the support vectors were those training points for which the corresponding coefficients in the solution function were non-zero. Under the geometric interpretation the support vectors are simply the training points that are on the 'wrong side' of the margin line – either they are misclassified or they are correctly classified, but lie within the margin.

5 Practical Issues

There are several practical tidbits that might help with the implementation of an SVM, or the choice of model. For instance, it was noted that Regularized Least Squares regression typically works about as well as an SVM in most cases. However, RLS requires the calculation of the entire kernel matrix. When there are too many training points, this becomes intractable. An SVM, on the other hand, only requires calculating the value of the kernel function between an input point and each support vector. This makes it possible to “chunk” the training set when learning an SVM. That is, to start with a small subset of the training points, learn an SVM, then discard all training points that are far from the decision boundary, add more of the training points and repeat. This effectively reduces the memory required to learn the classifier, allowing SVMs to scale potentially to much larger datasets than can RLS.

Finally, several good large scale SVM solvers were suggested:

- SVM Light: `http://svmlight.joachims.org`
- SVM Torch: `http://www.torch.ch`
- libSVM: `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`