Support Vector Machines

Kristian Wichmann

January 23, 2017

1 Overview

At its heart, a *support vector machine* (SVM) is a binary classifaction algorithm. The basic version creates a linear decision boundary, but using a strategy known as the *kernel trick*, we can make SVM's produce non-linear decision boundaries.

2 Linearly separable data

Initially, we assume that the data can be perfectly classified by a linear decision boundary - a hyperplane in configuration space such that all positives are on one side and all negatives on the other side. Generally, this can be done in many ways. So we will specifically search for the one which maximizes the perpendicular distance to the closest data points. Such closest point are known as *support vectors*. See figure 1.

2.1 Classifying a new data point

If we have a new data point $\vec{u}$, we might ask whether or not our algorithm classifies $\vec{u}$ as a positive or a negative.

The separation boundary is a hyperplane. Let $\vec{w}$ be a normal vector to this hyperplane[1]. We want to know how far in the direction of $\vec{w}$ our new data point $\vec{u}$ goes. In other words, we want to know the projection of $\vec{u}$ on $\vec{w}$:

$$\vec{u}_{\vec{w}} = \frac{\vec{u} \cdot \vec{w}}{|\vec{w}|^2} \vec{w} \tag{2.1}$$

So the signed distance from the origin to $\vec{u}$ is proportional to the inner product $\vec{u} \cdot \vec{w}$. Using more compact notation, this can also be written $u^T w =$

[1] The chosen vector is sometimes referred to as the *weight vector*.

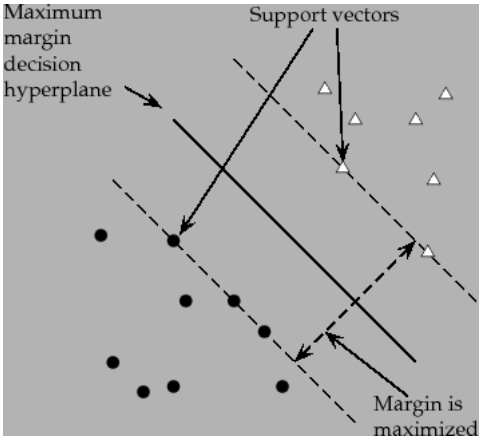

Figure 1: We're searching for the widest possible separation "road".

$w^t u$. Since the decision boundary is perpendicular to w, this inner product will have the same value for all points on it. We will call this value $-b$. Thus, the expression $w^T u + b$ will be exactly zero on the boundary, and its sign will otherwise decide which side the new data point is on. Hence, we may write the classifier as a function f:

$$f(u) = \text{sign}(w^T u + b) \tag{2.2}$$

Here, positives are identifies as $+1$ and negatives as -1[2].

2.2 A trick: introducing y_i

Consider two samples in the training set, x_+ and x_-, an arbitrary positive and ditto negative. So the following must be true:

$$w^T x_+ + b \geq 0, \qquad w^T x_- + b \leq 0 \tag{2.3}$$

However, since we assume there is a margin separating the positives and negatives, we may strengthen this condition to:

$$w^T x_+ + b \geq 1, \qquad w^T x_- + b \leq -1 \tag{2.4}$$

It is now convenient to introduce the following function of the sample x_i:

$$y_i = \begin{cases} +1 & \text{if } x_i \text{ positive} \\ -1 & \text{if } x_i \text{ negative} \end{cases} \tag{2.5}$$

[2]If these happen to be switched around, $-w$ will do the trick instead

Now, if we multiply inequalities 2.4 with y_i, we end up with the same inequality for an arbitrary sample x_i:

$$y_i(w^T x_i + b) \geq 1 \tag{2.6}$$

This may be rewritten:

$$y_i(w^T x_i + b) - 1 \geq 0 \tag{2.7}$$

Specifically, the equality holds for the support vectors - the vectors that are located on the margins:

$$y_i(w^T x_i + b) - 1 = 0 \quad \text{(for support vectors)} \tag{2.8}$$

2.3 Width of the "road"

Now, consider two support vectors, a positive and a negative one: x_+ and x_-. In order to get the width of the road, we can find the difference of these and project them unto the normal of the separation plane. The length of this projection is simply the dot product, assuming the normal has unit length:

$$\frac{1}{||w||} w^T (x_+ - x_-) \tag{2.9}$$

Here, w has been normalized to assure unit length. But we know that x_+ and x_- must satisfy equation 2.8, since they're both support vectors:

$$w^T x_+ + b - 1 = 0, \quad -w^T x_- - b - 1 = 0 \tag{2.10}$$

Adding the two we get:

$$w^T (x_+ - x_-) - 2 = 0 \Leftrightarrow w^T (x_+ - x_-) = 2 \tag{2.11}$$

Inserting into equation 2.9, we get that the width is $\frac{2}{||w||}$.

2.4 Maximizing the width

Since the width is inversely proportional to the length $||w||$, this means we should minimize the length. Since the length is non-negative, we might as well minimize $\frac{1}{2}||w||^2$. We will see in a minute why this is convenient.

The optimization happens with respect to the boundary conditions 2.8. So the natural approach is to use the method of Lagrange multipliers. The Lagrangian in this case is:

$$\mathcal{L} = \frac{1}{2}||w||^2 - \sum_i \alpha_i \left[y_i(w^T x_i + b) - 1 \right] \tag{2.12}$$

Here strictly, the sum is over the support vectors only. But for non-support vectors, the term will be greater than zero. Hence, if we include all training set vectors in the sum, when we minimize, the corresponding Lagrange multipliers will turn out to be zero.

We now seek to minimize $\mathcal{L}$ by differentiating with respect to the weight vector w:

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_i \alpha_i y_i x_i \tag{2.13}$$

Setting this equal to zero, we get the following formula for w:

$$w = \sum_i \alpha_i y_i x_i \tag{2.14}$$

Differentiation after b gives:

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_i \alpha_i y_i \tag{2.15}$$

So we have the following constraint on the Lagrange multipliers:

$$\sum_i \alpha_i y_i = 0 \tag{2.16}$$

2.5 Rewriting the Lagrangrian

Now, let's rewrite the Lagranagian in terms of equations 2.14 and 2.16:

$$\frac{1}{2}\left(\sum_i \alpha_i y_i x_i\right)^T \left(\sum_j \alpha_j y_j x_j\right) - \sum_i \alpha_i \left[y_i \left(\left(\sum_j \alpha_j y_j x_j\right)^T x_i + b \right) - 1 \right] \tag{2.17}$$

The first term reduces to:

$$\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{2.18}$$

The second term distributes into two terms. First:

$$\sum_i \alpha_i y_i \left(\left(\sum_j \alpha_j y_j x_j\right)^T x_i + b \right) = \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_j^T x_i + b \sum_i \alpha_i y_i \tag{2.19}$$

By switching the indices, the first term is equal to twice equation 2.18, and the second is zero according to equation 2.16. Finally, the second part of the second term is:

$$\sum_i \alpha_i \tag{2.20}$$

Combining everything (and keeping track of signs), we get:

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{2.21}$$

The important thing here is, that this depends on the training set **only** through the inner products between individual sample vectors: $x_i^T x_j$.

2.6 Revisiting the classifier

Reconsider equation 2.2 in terms of equation 2.14:

$$f(u) = \text{sign}(w^T u + b) = \text{sign}\left(\sum_i \alpha_i y_i x_i^T u + b \right) \tag{2.22}$$

Again, the only dependence on sample vectors is through inner products.

3 The kernel trick

If we have a set of data that is not linearly separable, we might try a transformation to a higher-dimensional space $\phi : x_i \mapsto \phi(x_i)$ where there's a better chance of such a separation.

But, as noted above, all dependence on sample vectors in the problem is though inner products! Hence, we don't really need to know anything else about ϕ than the inner products after the transformation. In other words, all we need is:

$$\phi(x_i)^T \phi(x_j) \equiv K(x_i, x_j) \tag{3.1}$$

Here, we've defined *the kernel function* K.