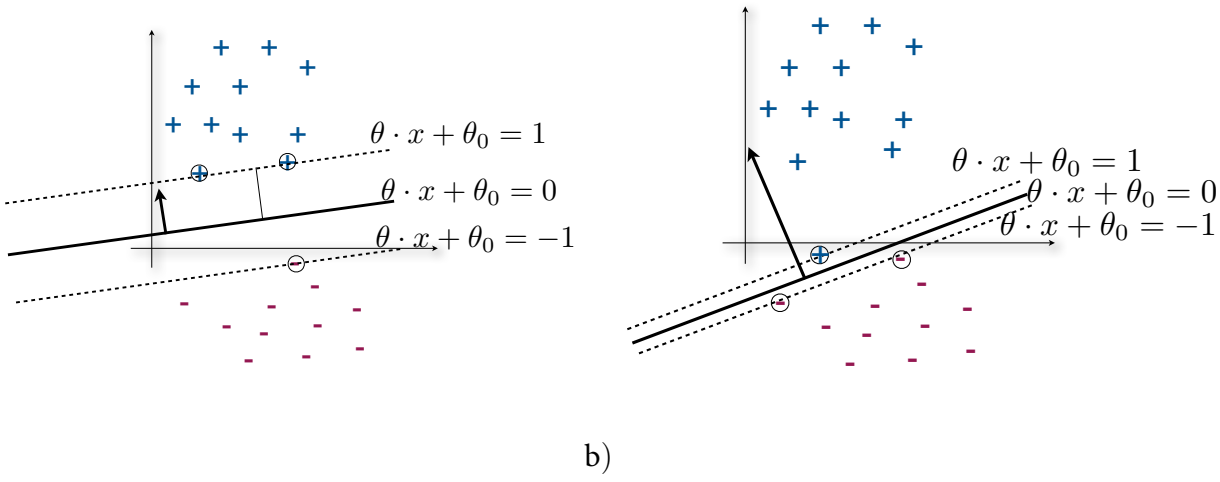
Support Vector Machines (Part 2)

Support vector machines (with errors)

If the labels for training examples contain errors, finding the maximum margin linear classifier may be problematic. The resulting decision boundary is potentially drastically affected by a single mislabeled point. Consider, for example, how the decision boundary changes by the single positive (perhaps mislabeled) point included in Figure 1b).



a) b)

Figure 1: The maximum margin separator is strongly affected by individual points

In order to remedy the situation, we should allow for misclassified points, yet maximize the margin. How do we achieve this? The simplest way is to treat the classification constraints as "soft" constraints rather than hard constraints. In other words, we permit them to be violated but specify the cost of such discrepancies in relation to increasing the margin (decreasing the norm $\|\theta\|$). Specifically, we will add "slack" variables $\xi_t \geq 0$ to the primal optimization problem as follows

$$\text{(primal)} \qquad \min \ \frac{\lambda}{2}\|\theta\|^2 + \frac{1}{n}\sum_{t=1}^{n}\xi_t \tag{6}$$

$$\text{subject to } \ y^{(t)}(\theta \cdot x^{(t)} + \theta_0) \geq 1 - \xi_t, \ \ \xi_t \geq 0, \ \ t = 1, \ldots, n \tag{7}$$

We now minimize with respect to the parameters θ, θ_0 as well as the slack variables $\xi_t \geq 0$. Note that we had to introduce a parameter[1] λ (regularization parameter) to balance how much we favor increasing the margin over satisfying the classification constraints. Larger values of λ will push the margin boundaries and potentially the decision boundary past the examples. The margin boundaries are still defined as points satisfying $\theta \cdot x + \theta_0 = 1$ or $\theta \cdot x + \theta_0 = -1$. Figure 2 illustrates the choice of λ and the resulting maximum margin solutions when the examples are still linearly separable.

[1] In the literature you will often see a parameter C multiplying the sum of slack variables instead. This is the same formulation so long as $C = 1/\lambda$.

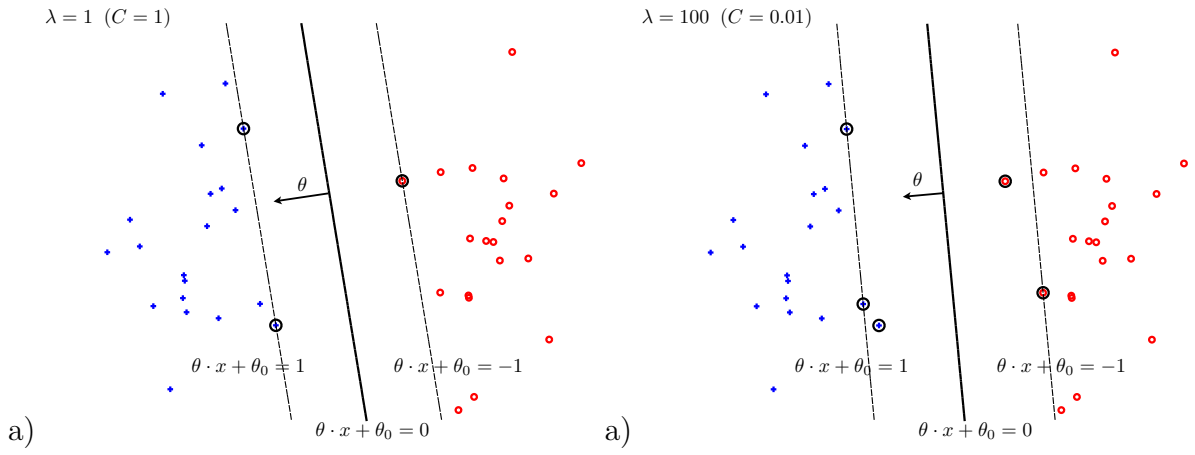


Figure 2: The effect of slack when examples are still linearly separable

The other advantage of the slack variables is that we can now solve problems that are no longer linearly separable. This is illustrated in Figure 3 with different values of the regularization parameter λ.

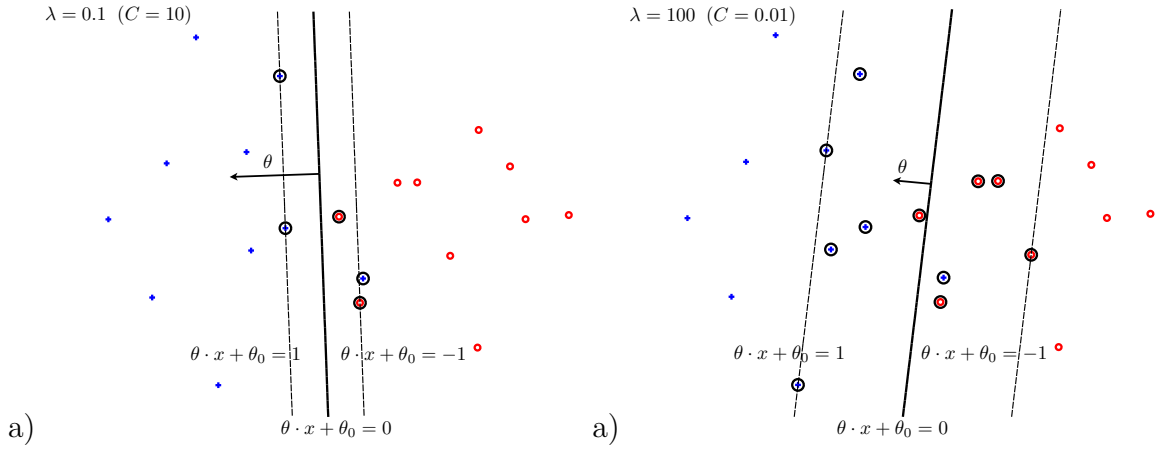


Figure 3: The effect of slack when examples are no longer linearly separable

Non-linear support vector machines – kernels

Let's start by considering how we can use linear classifiers to make non-linear predictions. The easiest way is to first map all the examples $x \in \mathcal{R}^d$ into a different feature

representation $\phi(x) \in \mathcal{R}^p$ where typically p is much larger than d. We would then simply use a linear classifier on the new (higher dimensional) feature vectors, pretending that they were the original input vectors. There are many ways to create such feature vectors. For example, we can build $\phi(x)$ by concatenating polynomial terms of the original coordinates. For example, in two dimensions, we could map $x = [x_1, x_2]^T$ to a five dimensional feature vector

$$\phi(x) = [x_1, x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]^T \tag{16}$$

We can then train a "linear" classifier (linear in the new ϕ-coordinates)

$$y = \text{sign}(\theta \cdot \phi(x) + \theta_0) \tag{17}$$

by mapping each training example to the corresponding feature vector. In other words, our training set is now $S_n^\phi = \{(\phi(x^{(t)}), y^{(t)}), t = 1, \ldots, n\}$. The resulting parameter estimates $\hat{\theta}$, $\hat{\theta}_0$ define a linear decision boundary in the ϕ-coordinates but a non-linear boundary in the original x-coordinates

$$\hat{\theta} \cdot \phi(x) + \hat{\theta}_0 = 0 \quad \Leftrightarrow \quad \hat{\theta}_1 x_1 + \hat{\theta}_2 x_2 + \hat{\theta}_3 \sqrt{2} x_1 x_2 + \hat{\theta}_4 x_1^2 + \hat{\theta}_5 x_2^2 + \hat{\theta}_0 = 0 \tag{18}$$

The non-linear boundary can represent, e.g., an ellipse in the original two dimensional space.

The main problem with the above procedure is that the feature vectors $\phi(x)$ can become quite high dimensional. For example, if we start with $x \in \mathcal{R}^d$, where $d = 1000$, then compiling $\phi(x)$ by concatenating polynomial terms up to the 2nd order would have dimension d+d(d-1)/2 or about 500, 000. Using higher order polynomial terms in such situations becomes quickly infeasible. However, it may still be possible to *implicitly* use such feature vectors. If training and prediction problems can be formulated only in terms of inner products between examples, then the relevant computation for us is $\phi(x) \cdot \phi(x')$. Depending on how we define $\phi(x)$, this computation can be carried out efficiently even if using $\phi(x)$ explicitly is not. For example, when $\phi(x) = [x_1, x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]^T$, we see that (check!)

$$\phi(x) \cdot \phi(x') = (x \cdot x') + (x \cdot x')^2 \tag{19}$$

So the inner product is obtained easily from the original input vectors.

One of the main advantages from considering the dual form of support vector machines is that it can be expressed entirely in terms of inner products. For example, in solving for the Lagrange multipliers $\hat{\alpha}_t$, the dual problem in Eq.(11) only requires us to evaluate inner products between the training examples, i.e., $(x^{(t)} \cdot x^{(t')})$. Similarly, the parameter $\hat{\theta}_0$ can be reconstructed from the margin constraints in Eq.(14) based on inner products. Finally, we can predict labels for new examples x with access only to the inner products between the training examples $x^{(t)}$ and the new points:

$$y = \text{sign}\left(\hat{\theta} \cdot x + \hat{\theta}_0\right) = \text{sign}\left(\sum_{t=1}^{n} \hat{\alpha}_t y^{(t)} (x^{(t)} \cdot x) + \hat{\theta}_0\right) \tag{20}$$

When we map examples into feature vectors, we replace $(x^{(t)} \cdot x^{(t')})$ with $\phi(x^{(t)}) \cdot \phi(x^{(t')})$. Instead of explicitly using feature vectors, we will focus on evaluating *kernel functions* $K(x^{(t)}, x^{(t')}) = \phi(x^{(t)}) \cdot \phi(x^{(t')})$.