Support Vector Machines (SVM)

Anhthy Ngo

1 The Support Vector Machine

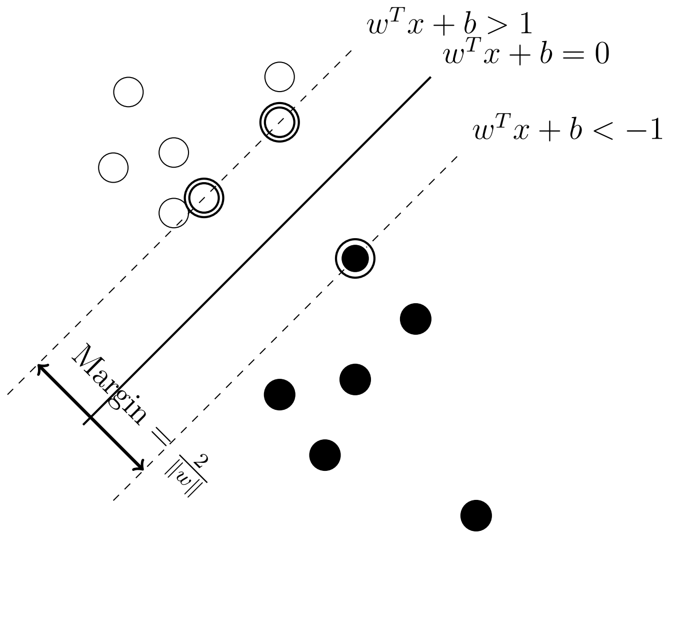
For a linear support vector machine (SVM), we use the hypothesis space of affine functions

$$\mathcal{F} = \{f(x) = w^T x + b \mid w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

and evaluate with respect to the **SVM loss function**, also known as the **hinge loss**. The hinge loss is a margin loss defined as $\ell(m) = (1 - m)_+$ where $m = yf(x)$ is the margin for the prediction function f on the example (x, y) and $(x)_+ = x1(x \geq 0)$ denotes the "positive part" of x. The SVM traditionally uses an $\ell 2$ regularization term, and the objective function is written as:

$$J(w, b) = \frac{1}{2}||w||^2 + \frac{c}{n}\sum_{i=1}^{n}(1 - y_i[w^T x_i + b])_+$$

1.1 Decision Rule



The class y is determined as follows:

$$y_i = \begin{cases} -1 & \text{if } w^T x_i + b \leq -1 \\ 1 & \text{if } w^T x_i + b \geq 1 \end{cases}$$

which can be more concisely written as $y_i(w^T x_i + b) \geq 1$.

1.2 Goal of SVM

1. Maximize the distance between the two decision boundaries. Mathematically, this means that we want to maximize the distance between the hyperplane defined by $w^T x + b = -1$ and the hyperplane defined by $w^T x + b = 1$. The distance between hyperplanes is equal to $\frac{2}{||w||}$. This means that we want to solve $\max\limits_x \frac{2}{\|w\|}$, which is equivalent to $\min\limits_x \frac{\|w\|}{2}^2$.

2. **Hard-Margin SVM:** The SVM should also correctly classify all x_i, which mathematically means
$$y_i(w^T x_i + b) \geq 1, \forall i \in \{1, \dots, N\}$$
This leads us to the following quadratic optimization problem:
$$\begin{aligned} \min_{w,b} \quad & \frac{\|w\|^2}{2} \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 \ , \forall i \in \{1, \dots, N\} \end{aligned} \tag{1}$$
We call this the **hard-margin SVM**, as this quadratic optimization problems only admits a solution iff the data is linearly separable. In other words, this variation of SVM <u>does not</u> allow for misclassifications.

3. **Soft-Margin SVM:** One can relax the constraints by adding **"slack variables"** ξ_i. Note that each sample of the training set has its own slack variable. This gives us the following quadratic optimization problem:
$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n}(1 - y_i[w_T x_i + b])_+ \tag{2}$$
The above equation is an unconstrained optimization problem (which is nice), but it's not differentiable. We can formulate an equivalent problem with a differentiable objective, but we will have to add new constraints to make this a constrained optimization problem. We get the following equivalent equation below:
$$\begin{aligned} \min_{w,b} \quad & \frac{\|w\|^2}{2} + \frac{c}{n}\sum_{i=1}^{n} \xi_i, \\ \text{s.t.} \quad & \xi_i \geq 1 - y_i(w^T x_i + b) \ , \forall i \in \{1, \dots, N\} \\ & \xi_i \geq 0 \ , \forall i \in \{1, \dots, N\} \end{aligned} \tag{3}$$

What is C? C is a hyperparameter called **penalty of the error term**. The C parameter tells the SVM how much to avoid misclassifying each training example. For large C, the optimization will choose smaller-margin hyperplane even if the hyperplane does a better job of getting all the training points classified correctly. Conversely, for small C, the optimization will look for a larger-margin separating hyperplane even if the hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples even if the training data is linearly separable.

2 Solving Non-Linear Problems with SVM

One can add even more flexibility by introduction a function ϕ that maps the original feature space to a higher dimensional feature space. This allows to solve non-linear problems with linear separators. The quadratic optimization problem becomes:

$$\begin{aligned} \min_{w,b} \quad & \frac{\|w\|^2}{2} + \frac{c}{n}\sum_{i=1}^{n} \xi_i, \\ \text{s.t.} \quad & \xi_i \geq 1 - y_i(w^T\phi(x_i) + b) \ , \forall i \in \{1, \dots, N\} \\ & \xi_i \geq 0 \ , \forall i \in \{1, \dots, N\} \end{aligned} \tag{4}$$

3 Optimizing SVM Dual Problem

The quadratic optimization problem can be transformed to another optimization problem called the Lagrangian dual problem (the previous problem is called the **primal**). The first step is to compute the Langrangian.

$$L(w, b, \xi, \alpha, \lambda) = \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha(1 - y_i[w^T x_i + b] - \xi_i) + \sum_{i+1}^{n}\lambda_i(-\xi_i)$$

From our original study of Lagrangian duality, we know that original problem can now be expressed as:

$$\inf_{w,b,\xi}\sup_{\alpha,\lambda\succeq 0} L(w, b, \xi, \alpha, \lambda)$$

Since our constraints are affine and this is a convex problem, by Slater's condition we have strong duality so long as the problem is feasible (i.e., so long as there is at least one point in the feasible set). The constraints are satisfied by $w = 0$ and $\xi_i = 1$ for $i = 1, \dots, n$, so we have **strong duality**. Thus we get the same result if we solve the following dual problem:

$$\sup_{\alpha,\lambda\succeq 0}\inf_{w,b,\xi} L(w, b, \xi, \alpha, \lambda)$$

The Lagrange dual is the inf over primal variables of the Lagrangian:

$$\begin{aligned} g(\alpha, \lambda) &= \inf_{w,b,\xi} L(w, b, \xi, \alpha, \lambda) \\ &= \inf_{w,b,\xi} [\frac{1}{2} w^T w + \sum_{i=1}^{n} \xi_i (\frac{c}{n} - \alpha_i - \lambda_i) + \sum_{i=1}^{n} \alpha_i (1 - y_i [w^T x_i + b])] \end{aligned} \tag{5}$$

By taking the partial derivatives with respect to each primal variable, we get the following:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\begin{aligned} w &= \sum_{i=1}^{n} \alpha_i y_i = 0 \\ w &= \alpha_i + \lambda_i = \frac{c}{n} \end{aligned}$$

Now, by substituting these conditions back into L, we get:

$$g(\alpha, \lambda) = \begin{cases} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i & \sum_{i=1}^{n} \alpha_i yi = 0 \\ & \alpha_i + \lambda_i = \frac{c}{n}, \text{ all i} \\ -\infty & \text{otherwise.} \end{cases}$$

And we get that the dual problem is $\sup\limits_{\alpha, \lambda \succcurlyeq 0} g(\alpha, \lambda)$:

$$\begin{aligned} \sup_{\alpha, \lambda} \quad & \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i, \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i + \lambda_i = \frac{c}{n}, \text{ all i} \end{aligned} \tag{6}$$

But wait.. we can actually eliminate the λ variables, replacing the last constraint by $0 \leq \alpha_i \leq \frac{c}{n}$ to get our final dual equation:

$$
\begin{aligned}
\sup_{\alpha} \quad & \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j x_j^T x_i, \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\
& \alpha_i \in [0, \frac{c}{n}]
\end{aligned} \tag{7}
$$

We learn a_i using the (x_i, y_i) of the training set.

4 Making Predictions with SVM

Once α_i is learned, one can predict the class of a new sample with the feature vector x^{test} as follows:

$$
\begin{aligned}
y^{\text{test}} &= \text{sign}(w^T\phi(x^{\text{test}}) + b) \\
&= \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \phi(x_i)^T \phi(x^{\text{test}}) + b)
\end{aligned} \tag{8}
$$

The summation $\sum_{i=1}^{N}$ can seem intimidating, since it means we have to sum over all training examples, but the vast majority of a_i are actuallly 0 (i.e., a_i is sparse). We conclude that x_i is a support vector iff the corresponding $a_i > 0$. Know that the fact that most $a_i > 0$ is a direct consequence of Karush-Kuhn-Tucker (KTT) dual complementarity conditions.

5 Kernel Trick

Soft-Margin SVM can handle non-linearly separable data caused by noise, but what if the non-separability is not caused by noise? Can we still separate using SVM? Yes, the technique "kernel trick" solves this problem. Recall the dual optimization problem (7), in which we only care about the dot product $x_j^T x_i$. The function that maps (x, y) to the inner product $(\phi(x)^T\phi(y))$ is called a kernel function, often denoted by k:

$$k(x, y) = \langle \phi(x), \phi(y) \rangle$$

One can choose a k s.t. the inner product is efficient to compute. We define the kernel trick as:

$$\begin{aligned}
\sup_{\alpha} \quad & \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(x_i, x_j), \\
\text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\
& \alpha_i \in [0, \frac{c}{n}]
\end{aligned} \tag{9}$$

This small change is a powerful trick and we now have the ability to calculate the result of a dot product performed in another space (i.e., we now have the ability to change the kernel function in order to classify non-linearly separable data). This allows to use potentially high dimensional feature spaces at a low computational cost.