Support Vector Machine

Cornell CS 4/5780 — Spring 2022

The Support Vector Machine (SVM) is a linear classifier that can be viewed as an extension of the Perceptron developed by Rosenblatt in 1958. The Perceptron guaranteed that you find **a** hyperplane if it exists. The SVM finds the **maximum margin** separating hyperplane.

Setting: We define a linear classifier: $h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$ and we assume a binary classification setting with labels $\{+1, -1\}$.

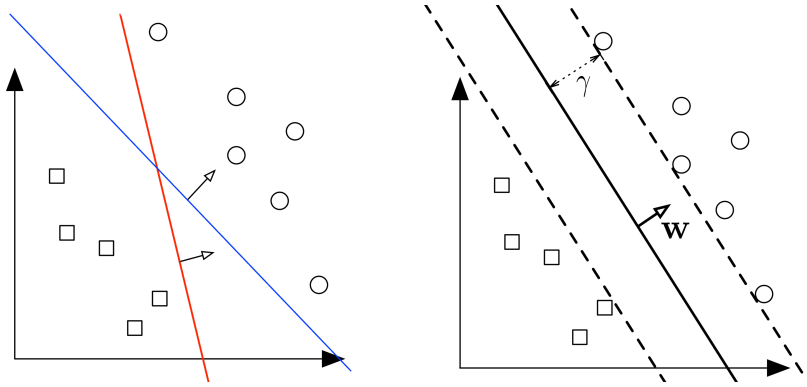
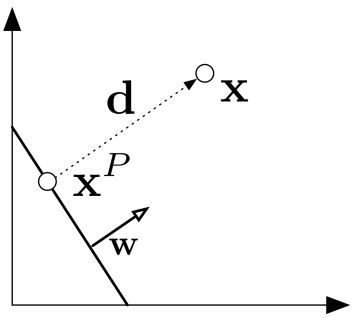


Figure 1: (Left:) Two different separating hyperplanes for the same data set. (Right:) The maximum margin hyperplane. The margin, γ, is the distance from the hyperplane (solid line) to the closest points in either class (which touch the parallel dotted lines).

Typically, if a data set is linearly separable, there are infinitely many separating hyperplanes. A natural question to ask is: **Question: What is the best separating hyperplane? SVM Answer: The one that maximizes the distance to the closest data points from both classes. We say it is the hyperplane with maximum margin.**

We already saw the definition of a *margin* in the context of the Perceptron. A hyperplane is defined through $\mathbf{w}, b$ as a set of points such that $\mathcal{H} = \{\mathbf{x} | \mathbf{w}^T\mathbf{x} + b = 0\}$. Let the margin γ be defined as the distance from the hyperplane to the closest point across both classes.

What is the distance of a point $\mathbf{x}$ to the hyperplane $\mathcal{H}$? Consider some point $\mathbf{x}$. Let $\mathbf{d}$ be the vector from $\mathcal{H}$ to $\mathbf{x}$ of minimum length. Let $\mathbf{x}^P$ be the projection of $\mathbf{x}$ onto $\mathcal{H}$. It follows then that: $\mathbf{x}^P = \mathbf{x} - \mathbf{d}$. $\mathbf{d}$ is parallel to $\mathbf{w}$, so $\mathbf{d} = \alpha\mathbf{w}$ for some $\alpha \in \mathbb{R}$. $\mathbf{x}^P \in \mathcal{H}$ which implies $\mathbf{w}^T\mathbf{x}^P + b = 0$ therefore $\mathbf{w}^T\mathbf{x}^P + b = \mathbf{w}^T(\mathbf{x} - \mathbf{d}) + b = \mathbf{w}^T(\mathbf{x} - \alpha\mathbf{w}) + b = 0$ which implies $\alpha = \frac{\mathbf{w}^T\mathbf{x}+b}{\mathbf{w}^T\mathbf{w}}$. The length of $\mathbf{d}$ is then: $\|\mathbf{d}\|_2 = \sqrt{\mathbf{d}^T\mathbf{d}} = \sqrt{\alpha^2\mathbf{w}^T\mathbf{w}} = \frac{|\mathbf{w}^T\mathbf{x}+b|}{\sqrt{\mathbf{w}^T\mathbf{w}}} = \frac{|\mathbf{w}^T\mathbf{x}+b|}{\|\mathbf{w}\|_2}$ Margin of $\mathcal{H}$ with respect to D: $\gamma(\mathbf{w}, b) = \min_{\mathbf{x}\in D} \frac{|\mathbf{w}^T\mathbf{x}+b|}{\|\mathbf{w}\|_2}$



By definition, the margin and hyperplane are scale invariant: $\gamma(\beta\mathbf{w}, \beta b) = \gamma(\mathbf{w}, b), \forall \beta \neq 0$

Note that if the hyperplane is such that γ is maximized, it must lie right in the middle of the two classes. In other words, γ must be the distance to the closest point within both classes. (If not, you could move the hyperplane towards data points of the class that is further away and increase γ, which contradicts that γ is maximized.)

Max Margin Classifier

We can formulate our search for the maximum margin separating hyperplane as a constrained optimization problem. The objective is to maximize the margin under the constraints that all data points must lie on the correct side of the hyperplane:

$$\underbrace{\max_{\mathbf{w},b} \gamma(\mathbf{w}, b)}_{\text{maximize margin}} \text{ such that } \underbrace{\forall i, \quad y_i(\mathbf{w}^T x_i + b) \geq 0}_{\text{separating hyperplane}}$$

If we plug in the definition of γ we obtain:

$$\underbrace{\max_{\mathbf{w},b} \underbrace{\frac{1}{\|\mathbf{w}\|_2} \min_{\mathbf{x}_i \in D} |\mathbf{w}^T\mathbf{x}_i + b|}_{\gamma(\mathbf{w},b)}}_{\text{maximize margin}} \text{ such that } \underbrace{\forall i \;\; y_i(\mathbf{w}^T x_i + b) \geq 0}_{\text{separating hyperplane}}.$$

Because the hyperplane is scale invariant, we can fix the scale of $\mathbf{w}, b$ anyway we want. Let's be clever about it, and choose it such that

$$\min_{\mathbf{x}\in D} |\mathbf{w}^T\mathbf{x} + b| = 1.$$

We can add this re-scaling as an equality constraint. Then our objective becomes:

$$\max_{\mathbf{w},b} \frac{1}{||\mathbf{w}||_2} \cdot 1 = \min_{\mathbf{w},b} ||\mathbf{w}||_2 = \min_{\mathbf{w},b} \mathbf{w}^\top \mathbf{w}$$

(Where we made use of the fact $f(z) = z^2$ is a monotonically increasing function for $z \geq 0$ and $||\mathbf{w}|| \geq 0$; i.e. the $\mathbf{w}$ that maximizes $||\mathbf{w}||_2$ also maximizes $\mathbf{w}^\top \mathbf{w}$.) The new optimization problem becomes:

$$\begin{aligned} &\min_{\mathbf{w},b} \mathbf{w}^\top \mathbf{w} \\ \text{s.t.} \quad &\forall i,\ y_i(\mathbf{w}^T \mathbf{x}_i + b) \quad \geq 0 \\ &\min_i \left|\mathbf{w}^T \mathbf{x}_i + b\right| \quad = 1 \end{aligned}$$

These constraints are still hard to deal with, however luckily we can show that (for the optimal solution) they are equivalent to a much simpler formulation. (Makes sure you know how to prove that the two sets of constraints are equivalent.)

$$\begin{aligned} &\min_{\mathbf{w},b} \mathbf{w}^T \mathbf{w} \\ &\text{s.t.} \quad \forall i\ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \end{aligned}$$

This new formulation is a quadratic optimization problem. The objective is *quadratic* and the constraints are all *linear*. We can be solve it efficiently with any QCQP (Quadratically Constrained Quadratic Program) solver. It has a unique solution whenever a separating hyper plane exists. It also has a nice interpretation: Find the simplest hyperplane (where simpler means smaller $\mathbf{w}^\top \mathbf{w}$) such that all inputs lie at least 1 unit away from the hyperplane on the correct side.

Support Vectors

For the optimal $\mathbf{w}, b$ pair, some training points will have tight constraints, i.e. $y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$. (This must be the case, because if for all training points we had a strict $>$ inequality, it would be possible to scale down both parameters $\mathbf{w}, b$ until the constraints are tight and obtained an even lower objective value.) We refer to these training points as **support vectors**. Support vectors are special because they are the training points that define the maximum margin of the hyperplane to the data set and they therefore determine the shape of the hyperplane. If you were to move one of them and retrain the SVM, the resulting hyperplane would change. The opposite is the case for non-support vectors (provided you don't move them too much, or they would turn into support vectors themselves). This will become particularly important in the dual formulation for Kernel-SVMs.

SVM with soft constraints

If the data is low dimensional it is often the case that there is no separating hyperplane between the two classes. In this case, there is no solution to the optimization problems stated above. We can fix this by allowing the constraints to be violated ever so slight with the introduction of slack variables:

$$\begin{gathered} \min_{\mathbf{w},b} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i \\ s.t.\ \forall i\ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\ \forall i\ \xi_i \geq 0 \end{gathered}$$

The slack variable ξ_i allows the input $\mathbf{x}_i$ to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such "slack". If C is very large, the SVM becomes very strict and tries to get all points to be on the right side of the hyperplane. If C is very small, the SVM becomes very loose and may "sacrifice" some points to obtain a simpler (i.e. lower $\|\mathbf{w}\|_2^2$) solution.

Unconstrained Formulation:

Let us consider the value of ξ_i for the case of $C \neq 0$. Because the objective will always try to minimize ξ_i as much as possible, the equation must hold as an *equality* and we have:

$$\xi_i = \begin{cases} 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) & \text{if } y_i(\mathbf{w}^T \mathbf{x}_i + b) < 1 \\ 0 & \text{if } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \end{cases}$$

This is equivalent to the following closed form:

$$\xi_i = \max(1 - y_i(\mathbf{w}^T \mathbf{x}_i + b), 0).$$

If we plug this closed form into the objective of our SVM optimization problem, we obtain the following *unconstrained* version as loss function and regularizer:

$$\min_{\mathbf{w},b} \underbrace{\mathbf{w}^T \mathbf{w}}_{l_2-regularizer} + C \sum_{i=1}^n \underbrace{\max\left[1 - y_i(\mathbf{w}^T \mathbf{x} + b), 0\right]}_{hinge-loss}$$

This formulation allows us to optimize the SVM paramters $(\mathbf{w}, b)$ just like logistic regression (e.g. through gradient descent). The only difference is that we have the **hinge-loss** instead of the **logistic loss**.