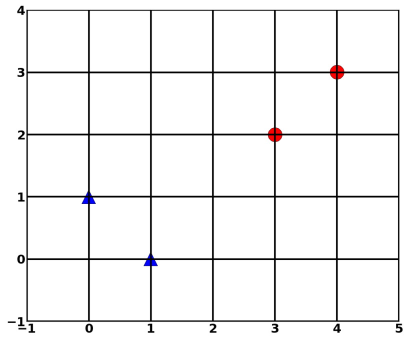
1 Support Vector Machines

Assume we are given dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$.

In SVM the goal is to find some hyperplane which separates the positive from the negative examples, such that the margin (the minimum distance from the decision boundary to the training points) is maximized. Let the equation for the hyperplane be $\mathbf{w}^\top \mathbf{x} + b = 0$.

(a) You are presented with the following set of data (triangle = +1, circle = -1):



Find the equation (by hand) of the hyperplane $\mathbf{w}^\top \mathbf{x} + b = 0$ that would be used by an SVM classifier. Which points are support vectors?

Solution:

Solving linearly separable classification problem is equivalent to finding the points of the two convex hulls which are closest to each other and the maximum margin hyperplane bisects and is normal to the line segment joining these two closest points.

In our case the convex hulls are the line segment joining the negative points and the line segment joining positive points, so the two points that are closest are $(3,2)$ and $(1,0)$. From the above we get that the equation of the hyperplane will pass through point $(2,1)$, with a slope of -1. The equation of this line is $x_1 + x_2 = 3$.

From the fact that $\mathbf{w}^\top \mathbf{x} + b = 0$ we know that $w_1 = w_2$. So we only need two more equations to solve for w_1, w_2 and b.

We also know that for points on the margins, the hyperlanes are $\mathbf{w}^\top \mathbf{x} + b = \pm 1$. We know that $(1,0)$ is on the "positive" hyperplane (as well as $(0,1)$ in this case) and $(3,2)$ is on the "negative" hyperplane, we get the following system of equations:

$$\begin{cases} 1w_1 + 0w_2 + b = 1 \\ 3w_1 + 2w_2 + b = -1 \\ w_1 = w_2 \end{cases}$$

Solving this system of equations, we get $\mathbf{w} = [-\frac{1}{2}, -\frac{1}{2}]^\top$ and $b = \frac{3}{2}$.

Now, we will show that that the support vectors are $(1,0)$ and $(3,2)$ by computing their αs (Lagrange multipliers of the primal optimization problem).

$$\begin{cases} \sum_{i=1}^4 \alpha_i y_i = 0 \\ \sum_{i=1}^4 \alpha_i y_i \mathbf{x}_i = \mathbf{w} \end{cases}$$

Let's denote αs for $\mathbf{x}_1 = (0,1), \mathbf{x}_2 = (1,0)$ and $\mathbf{x}_3 = (3,2)$, as α_1, α_2 and α_3 respectively. We know that $x_4 = (4,3)$ is not a support vector, hence $\alpha_4 = 0$, so we get:

$$\begin{cases} \alpha_1 + \alpha_2 - \alpha_3 = 0 \\ 0\alpha_1 + 1\alpha_2 - 3\alpha_3 = -\frac{1}{2} \\ 1\alpha_1 + 0\alpha_2 - 2\alpha_3 = -\frac{1}{2} \end{cases} \qquad \Rightarrow \alpha_1 = 0, \alpha_2 = \frac{1}{4}, \alpha_3 = \frac{1}{4}$$

(b) We typically frame an SVM problem as trying to *maximize* the margin. Explain intuitively why a bigger margin will result in a model that will generalize better, or perform better in practice.

Solution: One intuition is that if points are closer to the border, we are less certain about their class. Thus, it would make sense to create a boundary where our "certainty" is highest about all the training set points.

Another intuition involves thinking about the process that generated the data we are working with. Since it's a noisy process, if we drew a boundary close to one of our training points of some class, it's very possible that a point of the same class will be generated across the boundary, resulting in an incorrect classification. Therefore it makes sense to make the boundary as far away from our training points as possible.

(c) Show that the width of an SVM slab with linearly separable data is $\frac{2}{\|\mathbf{w}\|}$.

Solution: The width of the margin is defined by the points that lie on it, also called support vectors. The equation of the plane is $\mathbf{w}^\top \mathbf{x} + b = 0$ and since $c(\mathbf{w}^\top \mathbf{x} + b) = 0$ defines the same plane, we have the freedom to choose the normalization of $\mathbf{w}$. Let us choose normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$, for the positive and negative support vectors respectively.

Let's say we have a point, $\mathbf{x}_+$, which is a support vector. The distance between $\mathbf{x}_+$ and the separating hyperplane can be calculated by projecting the vector starting at the plane $\mathbf{x}$ and ending at $\mathbf{x}_+$ onto the plane's unit normal vector.

Since $\mathbf{w}$ by definition is orthogonal to the hyperplane, we want to project $\mathbf{x}_+ - \mathbf{x}$ onto the unit vector normal to the hyperplane, $\frac{\mathbf{w}}{\|\mathbf{w}\|}$.

$$\frac{\mathbf{w}^\top}{\|\mathbf{w}\|}(\mathbf{x}_+ - \mathbf{x}) = \frac{1}{\|\mathbf{w}\|}(\mathbf{w}^\top\mathbf{x}_+ - \mathbf{w}^\top\mathbf{x}) = \frac{1}{\|\mathbf{w}\|}(\mathbf{w}^\top\mathbf{x}_+ + b - \mathbf{w}^\top\mathbf{x} - b)$$

Since we set $\mathbf{w}^\top\mathbf{x}_+ + b = +1$ and by definition, $\mathbf{w}^\top\mathbf{x} + b = 0$, this quantity just turns into $\frac{1}{\|\mathbf{w}\|}$, or $\mathbf{1}\|\mathbf{w}\|$, so the distance is the absolute value, $\frac{1}{\|\mathbf{w}\|}$.

Since the margin is half of the slab, we double it to get the full width of $\frac{2}{\|\mathbf{w}\|}$.

(d) Write SVM as an optimization problem. Conclude that maximizing the margin is equivalent to minimizing $\|\mathbf{w}\|$.

Solution: In the previous part we saw that the width of the margin is $\frac{1}{\|\mathbf{w}\|}$. Since the margin hyperplane are given by $\mathbf{w}^\top\mathbf{x}+b = +1$ and $\mathbf{w}^\top\mathbf{x}+b = -1$, we have the following constraints for all i:

$$\begin{aligned} \mathbf{w}^\top\mathbf{x}_i + b &\geq +1, \quad \text{if } y_i = +1 \quad \text{or} \\ \mathbf{w}^\top\mathbf{x}_i + b &\geq -1, \quad \text{if } y_i = -1 \end{aligned}$$

We can rewrite the constraints above as $y_i(\mathbf{w}^\top\mathbf{x}_i - b) \geq 1 \quad \forall i$.

Since our gola is to maximize the margin, formally we can write SVM as the following optimization problem:

$$\begin{aligned} &\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|} \\ &\text{s.t.} \quad y_i(\mathbf{w}^\top\mathbf{x}_i - b) \geq 1 \quad \forall i \end{aligned}$$

Notice that maximizing $\|\mathbf{w}\|^{-1}$ is equivalent to minimizing $\|\mathbf{w}\|$ and since $f(x) = x^2$ is an increasing funciton on $[0,\infty)$ it also equivalent to minimizing $f(\|\mathbf{w}\|) = \|\mathbf{w}\|^2$. So we get,

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2} \\ &\text{s.t.} \quad y_i(\mathbf{w}^\top\mathbf{x}_i - b) \geq 1 \quad \forall i \end{aligned}$$

The above is an optimization problem with a convex quadratic objective and only linear constraints. This optimization problem is quadratic programming (QP) problem, for which many efficient solvers are available.

(e) Will moving points which are not support vectors further away from the decision boundary effect the SVM's hinge loss?

Solution: No, the hinge loss is defined as $\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b))$. For nonsupport vectors, the right hand side of the max function is already negative and moving the point further away from the boundary will make it only more negative. The max will return a zero regardless. This means that the loss function and the consequent decision boundary is entirely determined by the orientation of the support vectors and nothing else.

2 Curse of Dimensionality

We have a training set: $(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})$, where $\mathbf{x}^{(i)} \in \mathbb{R}^d$. To classify a new point $\mathbf{x}$, we can use the nearest neighbor classifier:

$$\text{class}(\mathbf{x}) = y^{(i^*)} \quad \text{where } \mathbf{x}^{(i^*)} \text{ is the nearest neighbor of } \mathbf{x}.$$

Assume any data point $\mathbf{x}$ that we may pick to classify is inside the Euclidean ball of radius 1, i.e. $\|\mathbf{x}\|_2 \le 1$. To be confident in our prediction, in addition to choosing the class of the nearest neighbor, we want the distance between $\mathbf{x}$ and its nearest neighbor to be small, within some positive ϵ:

$$\|\mathbf{x} - \mathbf{x}^{(i^*)}\|_2 \le \epsilon \quad \text{for all } \|\mathbf{x}\|_2 \le 1. \tag{1}$$

What is the minimum number of training points we need for inequality (1) to hold (assuming the training points are well spread)? How does this lower bound depend on the dimension d?

Hint: Think about the volumes of the hyperspheres in d dimensions.

Solution: Let B_0 be the ball centered at the origin, having radius 1 (inside which we assume our data lies). Let $B_i(\epsilon)$ be the ball centered at $\mathbf{x}^{(i)}$, having radius ϵ. For inequality (1) to hold, for any point $\mathbf{x} \in B_0$, there must be at least one index i such that $\mathbf{x} \in B_i(\epsilon)$. This is equivalent to saying that the union of $B_1(\epsilon), \dots, B_n(\epsilon)$ covers the ball B_0. Let $\text{vol}(B)$ indicate the volume of object B, then we have

$$\sum_{i=1}^{n} \text{vol}(B_i(\epsilon)) = n\text{vol}(B_1(\epsilon)) \ge \text{vol}(\cup_{i=1}^{n} B_i(\epsilon)) \ge \text{vol}(B_0).$$

where the last inequality holds because we are assuming the union of $B_1(\epsilon), \dots, B_n(\epsilon)$ covers the ball B_0. This implies

$$n \ge \frac{\text{vol}(B_0)}{\text{vol}(B_1(\epsilon))} = \frac{c(1^d)}{c\epsilon^d} = \frac{1}{\epsilon^d}$$

Where the constant c is dependent on the formula for the volume of a hypersphere in d dimensions.

Note that we can pick $\frac{1}{\epsilon^d}$ training points and still satisfy (1) only if all the training points are well spread (the union of $B_1(\epsilon), \dots, B_n(\epsilon)$ covers the ball B_0).

This lower bound suggests that to make an accurate prediction on high-dimensional input, we need exponentially many samples in the training set. This exponential dependence is sometimes called the *curse of dimensionality*. It highlights the difficulty of using non-parametric methods for solving high-dimensional problems.

Extra: If we receive a dataset in a d dimensional ambient space but the data truly only is in k dimensions where $k < d$, the effective lower bound on the number of training points decreases from $\frac{1}{\epsilon^d}$ to $\frac{1}{\epsilon^k}$. For example, in Figure 1, the training data points $\mathbf{x}^{(i)} \in \mathbb{R}^3$, but a simple linear transformation can bring them to $\mathbb{R}^2$. Assuming these training points are well spread, we can use a training set of $\frac{1}{\epsilon^2}$ points to cover the space of the dataset (according to (1)).

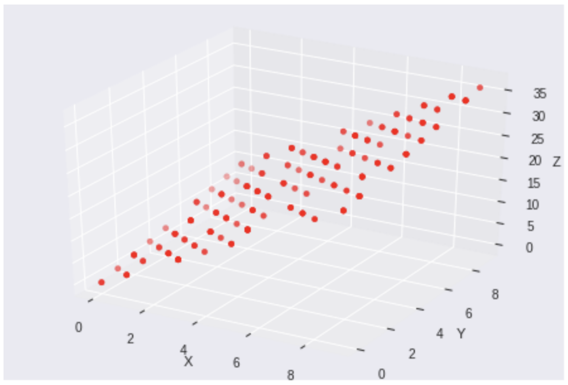


Figure 1: 3D dataset that can be represented in 3D