CS 760: Machine Learning Spring 2022

Topic 11: Support Vector Machines

Instructor: Daniel L. Pimentel-Alarcón

DO NOT POLLUTE! AVOID PRINTING, OR PRINT 2-SIDED MULTIPAGE.

11.1 Introduction

Support vector machines (SVMs) are considered one of the best "out of the box" classifiers. Similar to logistic regression and random forests, their main purpose is prediction/classification, and are used for similar applications (e.g. distinguish between healthy and Alzheimers). However, SVMs are a more geometric approach. The main idea is to think of each sample as a point in a high-dimensional space, and classify it according to the region where it is located with respect to a *boundary* (see Figure 11.1). The challenge is to find such separating/classifying boundary.

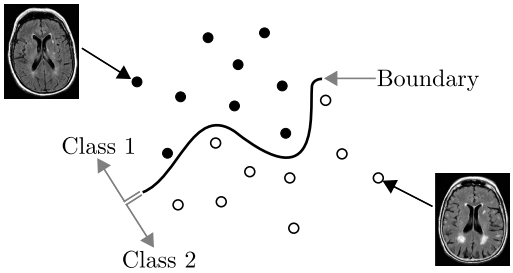


Figure 11.1: An SVM classifies each point according to the region where it is located with respect to a boundary.

11.2 Hyperplanes

Hyperplanes lie at the heart of SVMs. Intuitively, a hyperplane is the generalization of a line and a plane (in 2 and 3 dimensions) to higher dimensions. In words, hyperplanes are subspaces almost as big as the whole space. More formally, a hyperplane $\mathbb{H} \subset \mathbb{R}^{\mathrm{D}}$ is a linear subspace of dimension $\mathrm{D}-1$ (see Figure 11.2 to build some intuition).

Recall from linear algebra that the dimensions of a subspace $\mathbb{U} \subset \mathbb{R}^{\mathrm{D}}$ and its orthogonal complement $\mathbb{U}^{\perp} \subset \mathbb{R}^{\mathrm{D}}$ add to the ambient dimension, i.e.,

$$\mathrm{D} \;=\; \dim(\mathbb{U}) + \dim(\mathbb{U}^{\perp}),$$

which means that $\dim(\mathbb{H}^{\perp}) = 1$, i.e., $\mathbb{H}^{\perp}$ is a line. As such, it is characterized by a single vector. Let $\boldsymbol{\theta} \in \mathbb{R}^{\mathrm{D}}$ be the vector spanning $\mathbb{H}^{\perp}$. Also recall that $\mathbb{U}$ is the collection of all points orthogonal to $\mathbb{U}^{\perp}$. Consequently,

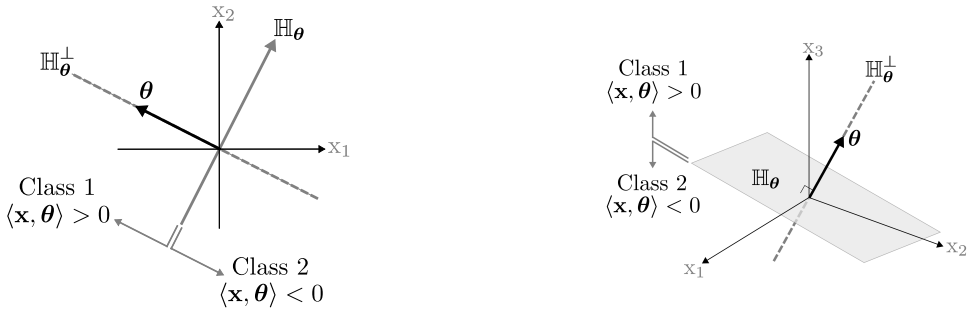


Figure 11.2: Hyperplanes are subspaces of dimension equal to the whole space minus 1. **Left:** A line in $\mathbb{R}^2$ is a hyperplane. **Right:** A plane in $\mathbb{R}^3$ is a hyperplane. Hyperplanes divide the whole space in two halves, providing a natural classifying border. Each hyperplane $\mathbb{H}_{\boldsymbol{\theta}}$ is characterized by the vector $\boldsymbol{\theta}$ spanning its orthogonal complement $\mathbb{H}^{\perp}$. **Question:** Is a line in $\mathbb{R}^3$ a hyperplane?

we can characterize $\mathbb{H}$ as the collection of all points in $\mathbb{R}^{\mathrm{D}}$ that are orthogonal to $\boldsymbol{\theta}$. To make this clear we use the following notation to denote the hyperplane orthogonal to $\boldsymbol{\theta}$:

$$\mathbb{H}_{\boldsymbol{\theta}} \;:=\; \Big\{\, \mathbf{x} \in \mathbb{R}^{\mathrm{D}} \;:\; \langle \mathbf{x}, \boldsymbol{\theta} \rangle = 0 \,\Big\},$$

where $\langle \mathbf{x}, \boldsymbol{\theta} \rangle \;=\; \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}$ denotes the inner product (see Figure 11.2 to build some intuition). Hyperplanes are powerful classification tools because they divide the space in two halves: the points $\mathbf{x}$ *above* the hyperplane, meaning $\langle \mathbf{x}, \boldsymbol{\theta} \rangle > 0$, and the points $\mathbf{x}$ *below* the hyperplane, meaning $\langle \mathbf{x}, \boldsymbol{\theta} \rangle < 0$ (see Figure 11.2).

Like all subspaces, hyperplanes must cross the origin. A more general model is an *affine* hyperplane, that is, a translated hyperplane. More precisely, we define the affine hyperplane $\mathbb{H}_{\boldsymbol{\theta}\theta_0}$ as the hyperplane with $\boldsymbol{\theta}$ as orthogonal direction, translated a distance θ_0 from the origin in the opposite direction of $\boldsymbol{\theta}$, such that if $\mathbf{x}$ is a point in the centered hyperplane, then the point $\mathbf{x} - \theta_0 \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}$ will be in the affine hyperplane. That is:

$$\mathbb{H}_{\boldsymbol{\theta}\theta_0} \;=\; \left\{\, \mathbf{x} - \theta_0 \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|} \in \mathbb{R}^{\mathrm{D}} \;:\; \langle \mathbf{x}, \boldsymbol{\theta} \rangle = 0 \,\right\}.$$

Equivalently,

$$\mathbb{H}_{\boldsymbol{\theta}\theta_0} \;:=\; \left\{\, \mathbf{x} \in \mathbb{R}^{\mathrm{D}} \;:\; \langle \mathbf{x} + \theta_0 \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}, \boldsymbol{\theta} \rangle = 0 \,\right\}.$$

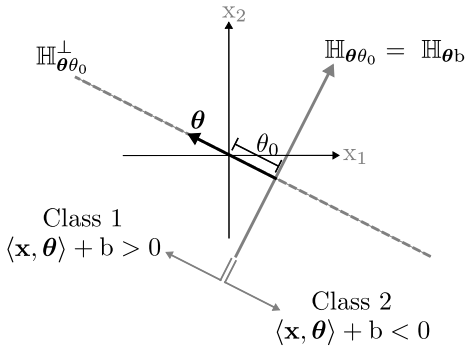
Notice that

$$\left\langle \mathbf{x} + \theta_0 \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}, \boldsymbol{\theta} \right\rangle \;=\; \langle \mathbf{x}, \boldsymbol{\theta} \rangle + \frac{\theta_0}{\|\boldsymbol{\theta}\|} \langle \boldsymbol{\theta}, \boldsymbol{\theta} \rangle \;=\; \langle \mathbf{x}, \boldsymbol{\theta} \rangle + \theta_0 \|\boldsymbol{\theta}\|.$$

Defining the so-called *bias* as $\mathrm{b} := \theta_0 \|\boldsymbol{\theta}\|$, we can equivalently characterize $\mathbb{H}_{\boldsymbol{\theta},\theta_0}$ in terms of $\boldsymbol{\theta}$ and b as the set

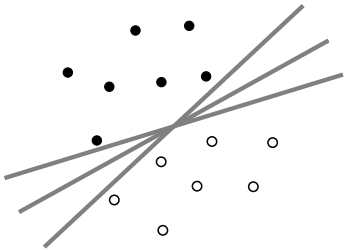
$$\mathbb{H}_{\boldsymbol{\theta}\mathrm{b}} \;:=\; \Big\{\, \mathbf{x} \in \mathbb{R}^{\mathrm{D}} \;:\; \langle \mathbf{x}, \boldsymbol{\theta} \rangle + \mathrm{b} = 0 \,\Big\}.$$

In what follows we use $\mathbb{H}_{\boldsymbol{\theta}\theta_0}$ and $\mathbb{H}_{\boldsymbol{\theta}\mathrm{b}}$ interchangeably.

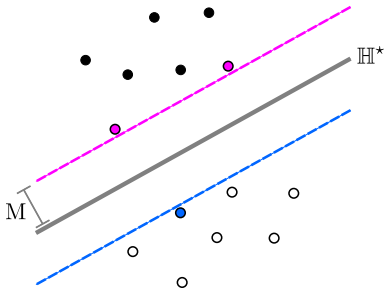


11.3 Maximal Margin Classifier

SVMs happen to be a generalization of the *maximal margin classifier* (MMC), which uses an *affine hyperplane* as the separating/classifying border. Notice, however, that given some data, it is possible that there are infinitely many affine hyperplanes that could separate it, for example:

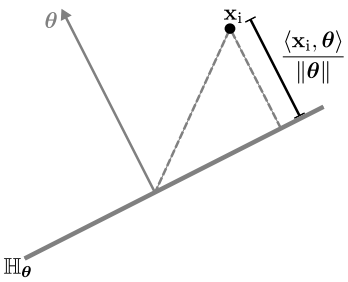


As the name suggests, the MMC finds the affine hyperplane $\mathbb{H}^{\star}$ that separates data $\mathbf{x}_1, \ldots, \mathbf{x}_{\mathrm{N}} \in \mathbb{R}^{\mathrm{D}}$ into classes $\mathrm{y}_1, \ldots, \mathrm{y}_{\mathrm{N}} \in \{-1, 1\}$ with the largest possible margin M:

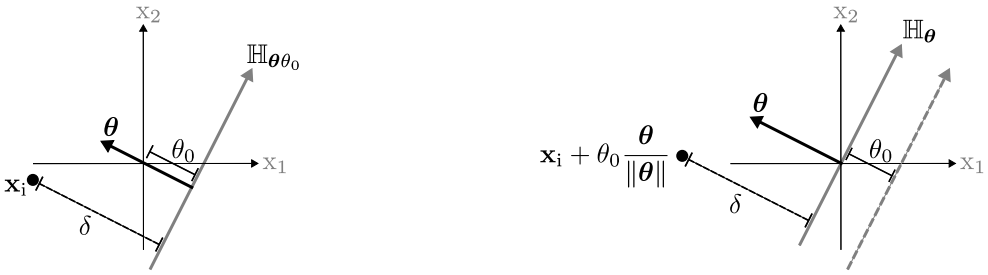


All vectors that are exactly within M distance of the affine hyperplane are called *support vectors*. To determine this distance, recall from linear algebra and projections that the distance between a point $\mathbf{x}_{\mathrm{i}}$ and the hyperplane $\mathbb{H}_{\boldsymbol{\theta}}$ is

$$\delta(\mathbf{x}_{\mathrm{i}}, \mathbb{H}_{\boldsymbol{\theta}}) \;=\; \frac{\langle \mathbf{x}_{\mathrm{i}}, \boldsymbol{\theta} \rangle}{\|\boldsymbol{\theta}\|}$$



Then notice that $\delta(\mathbf{x_i}, \mathbb{H}_{\boldsymbol{\theta}\theta_0})$ is the same as $\delta(\mathbf{x_i} + \theta_0 \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}, \mathbb{H}_{\boldsymbol{\theta}})$:



It follows that

$$\delta(\mathbf{x_i}, \mathbb{H}_{\boldsymbol{\theta}\theta_0}) \;=\; \delta(\mathbf{x_i} + \theta_0 \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}, \mathbb{H}_{\boldsymbol{\theta}}) \;=\; \frac{\langle \mathbf{x_i} + \theta_0 \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}, \boldsymbol{\theta} \rangle}{\|\boldsymbol{\theta}\|} \;=\; \frac{\langle \mathbf{x_i}, \boldsymbol{\theta} \rangle + \theta_0 \|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|} \;=\; \frac{\boldsymbol{\theta}^\mathsf{T} \mathbf{x_i} + \mathrm{b}}{\|\boldsymbol{\theta}\|}.$$

To ensure that each $\mathbf{x_i}$ is always on *the right* side of the affine hyperplane (determined by the label $\mathrm{y_i}$) we can define the *margin* between point $\mathrm{x_i}$ and the affine hyperplane as:

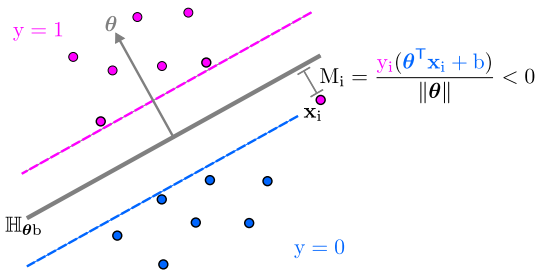
$$\mathrm{M_i} \;:=\; \frac{\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T} \mathbf{x_i} + \mathrm{b})}{\|\boldsymbol{\theta}\|},$$

to ensure that

(a) if $\mathrm{y_i} = 1$, then $\boldsymbol{\theta}^\mathsf{T} \mathbf{x_i} + \mathrm{b} > 0$, indicating that $\mathbf{x_i}$ is above the affine hyperplane,

(b) if $\mathrm{y_i} = -1$, then $\boldsymbol{\theta}^\mathsf{T} \mathbf{x_i} + \mathrm{b} < 0$, indicating that $\mathbf{x_i}$ is below the affine hyperplane,

so that if the signs of $\mathrm{y_i}$ and if $\boldsymbol{\theta}^\mathsf{T} \mathbf{x_i} + \mathrm{b}$ do not match, then the margin is negative, indicating that $\mathbf{x_i}$ was misclassified.



Since we want that the margin is large for *all* points, we need to guarantee that the point with the minimum margin is as large as possible. In other words, we want to find

$$(\boldsymbol{\theta}^\star, \mathrm{b}^\star) \;:=\; \arg\max_{\boldsymbol{\theta},\mathrm{b}} \;\; \min_{\mathrm{i}} \;\; \frac{\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b})}{\|\boldsymbol{\theta}\|}. \tag{11.1}$$

This problem looks quite complicated. However, notice that $\mathrm{M_i}$ is invariant to scalings of $\boldsymbol{\theta}$. That is,

$$\frac{\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b})}{\|\boldsymbol{\theta}\|} = \frac{\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \theta_0\|\boldsymbol{\theta}\|)}{\|\boldsymbol{\theta}\|} = \frac{\mathrm{y_i}(\mathrm{c}\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \theta_0\|\mathrm{c}\boldsymbol{\theta}\|)}{\|\mathrm{c}\boldsymbol{\theta}\|} \qquad \forall\, \mathrm{c} \neq 0.$$

So we can scale $\boldsymbol{\theta}$ so that $\min_\mathrm{i} \mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b}) = 1$, and rewrite (11.1) as

$$(\boldsymbol{\theta}^\star, \mathrm{b}^\star) \;:=\; \arg\max_{\boldsymbol{\theta},\mathrm{b}} \;\; \frac{1}{\|\boldsymbol{\theta}\|} \qquad \text{subject to} \qquad \min_{\mathrm{i}} \;\; \mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b}) = 1.$$

The intuition here is that since $\min_\mathrm{i} \mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b}) = 1$, the minimum margin over all samples is $\frac{\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b})}{\|\boldsymbol{\theta}\|} = \frac{1}{\|\boldsymbol{\theta}\|}$. Moreover, since the condition that $\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b}) \geq 1$ for every i implies $\min_\mathrm{i} \mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b}) = 1$, we can further rewrite (11.1) as

$$(\boldsymbol{\theta}^\star, \mathrm{b}^\star) \;:=\; \arg\max_{\boldsymbol{\theta},\mathrm{b}} \;\; \frac{1}{\|\boldsymbol{\theta}\|} \qquad \text{subject to} \qquad \mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b}) \geq 1 \quad \forall\, \mathrm{i} = 1, \ldots, \mathrm{N}. \tag{11.2}$$

In words, (11.2) is maximizing the general margin $\mathrm{M} := \frac{1}{\|\boldsymbol{\theta}\|}$ such that the margin of each data point $\mathrm{M_i}$ is at least as large as M (guaranteed by the constraint, which is equivalent to: $\mathrm{M_i} := \frac{\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b})}{\|\boldsymbol{\theta}\|} \geq \frac{1}{\|\boldsymbol{\theta}\|} =: \mathrm{M}$).

11.4 Lagrange Multipliers and the Dual Problem

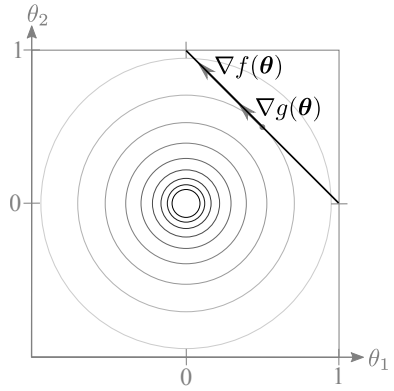
In practice, to solve a constrained problem like (11.2) we use *Lagrange Multipliers* and the *dual problem*. To build some intuition about this method, consider a simpler optimization problem:

$$\boldsymbol{\theta}^\star \;:=\; \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^2} \;\; \|\boldsymbol{\theta}\|^2 \qquad \text{subject to} \qquad \theta_1 + \theta_2 \geq 1, \tag{11.3}$$

Notice that the constraint can be written as $\theta_1 + \theta_2 - 1 \geq 0$. If we plot these two functions $f(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|^2 = \theta_1^2 + \theta_2^2$ and $g(\boldsymbol{\theta}) = \theta_1 + \theta_2 - 1$, we can see that the minimum of $f(\boldsymbol{\theta})$ under the constraint $g(\boldsymbol{\theta}) \geq 0$ is attained when the gradients of these two functions point in the same direction, i.e., when

$$\nabla f(\boldsymbol{\theta}) \;=\; \lambda \nabla g(\boldsymbol{\theta}), \tag{11.4}$$

where the so-called *lagrange multiplier* λ is a non-negative scalar, because we want the gradients to point in the same direction, but they may be of different magnitudes.



Defining the *Lagrangian* as

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) \;:=\; f(\boldsymbol{\theta}) - \lambda g(\boldsymbol{\theta}),$$

we can write (11.4) as

$$\nabla \mathcal{L}(\boldsymbol{\theta}, \lambda) \;=\; \mathbf{0}. \tag{11.5}$$

We thus conclude that the minimizer that we are looking for must satisfy (11.5) for some λ. In our example, $\nabla \mathcal{L}(\boldsymbol{\theta}, \lambda)$ has entries

$$\frac{\mathrm{d}}{\mathrm{d}\theta_{\mathrm{i}}} \mathcal{L}(\boldsymbol{\theta}, \lambda) \;=\; \frac{\mathrm{d}}{\mathrm{d}\theta_{\mathrm{i}}} \Big[\theta_1^2 + \theta_2^2 - \lambda(\theta_1 + \theta_2 - 1) \Big] \;=\; 2\theta_{\mathrm{i}} - \lambda. \tag{11.6}$$

Setting these to zero according to (11.5) and solving for $\boldsymbol{\theta}$ we obtain the minimizer we were looking for:

$$\boldsymbol{\theta} \;=\; \lambda \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}. \tag{11.7}$$

The only problem is that it is in terms of λ, which we do not know. To discover what λ should be, let

$$\boldsymbol{\theta}_\lambda \;:=\; \arg\min_{\boldsymbol{\theta}} \; \mathcal{L}(\boldsymbol{\theta}, \lambda) \qquad \text{subject to} \qquad \nabla \mathcal{L}(\boldsymbol{\theta}, \lambda) \;=\; \mathbf{0},$$

and notice that $\mathcal{L}(\boldsymbol{\theta}_\lambda, \lambda)$ is a lower bound on $f(\boldsymbol{\theta}^\star)$. To see this write

$$\mathcal{L}(\boldsymbol{\theta}_\lambda, \lambda) \;=\; \min_{\boldsymbol{\theta}: \nabla \mathcal{L} = \mathbf{0}} \; \mathcal{L}(\boldsymbol{\theta}, \lambda) \;\leq\; \mathcal{L}(\boldsymbol{\theta}^\star, \lambda) \;=\; f(\boldsymbol{\theta}^\star) - \lambda g(\boldsymbol{\theta}^\star) \;\leq\; f(\boldsymbol{\theta}^\star),$$

where the last inequality follows because $\lambda \geq 0$, and since $\boldsymbol{\theta}^\star$ satisfies the constraint by definition, $g(\boldsymbol{\theta}^\star) \geq 0$. Since we want the closest lower bound on $f(\boldsymbol{\theta}^\star)$, it makes sense that we try to find

$$\hat{\lambda} \;:=\; \arg\max_{\lambda \geq 0} \; \min_{\boldsymbol{\theta}: \nabla \mathcal{L} = \mathbf{0}} \; \mathcal{L}(\boldsymbol{\theta}, \lambda) \;=\; \arg\max_{\lambda \geq 0 : \nabla \mathcal{L} = \mathbf{0}} \; \mathcal{L}(\boldsymbol{\theta}_\lambda, \lambda),$$

which is often called the *dual problem*. In our example, $\boldsymbol{\theta}_\lambda$ can be determined using our standard Optimization 101 approach: take the derivative (which we already did in (11.6)), set to zero, and solve for $\boldsymbol{\theta}$ to get

$$\boldsymbol{\theta}_\lambda \;=\; \lambda \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, \tag{11.8}$$

same as in (11.7). This solution already satisfies $\nabla \mathcal{L}(\boldsymbol{\theta}_\lambda, \lambda) = \mathbf{0}$ by construction, so we don't need to worry about this constraint (as we will see below when studying the MMC case, in general the constraint is not always automatically satisfied, and needs to be *carried*). It follows that

$$\mathcal{L}(\boldsymbol{\theta}_\lambda, \lambda) \;=\; \lambda - \frac{\lambda^2}{2}.$$

To maximize this we follow our standard Optimization 101 approach once more: first take the derivative:

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} \mathcal{L}(\boldsymbol{\theta}_\lambda, \lambda) \;=\; 1 - \lambda,$$

then set it to zero, and solve for λ to obtain $\hat{\lambda} = 1$. Plugging this back into (11.8) we get

$$\hat{\boldsymbol{\theta}} \;:=\; \boldsymbol{\theta}_{\hat{\lambda}} \;=\; \hat{\lambda} \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \;=\; \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}.$$

As shown before, in general $f(\hat{\boldsymbol{\theta}})$ is a lower bound on $f(\boldsymbol{\theta}^\star)$, and their difference is known as the *dual gap*. Whenever this difference is zero, i.e., $f(\hat{\boldsymbol{\theta}}) = f(\boldsymbol{\theta}^\star)$, we say that there exists *strong duality*. There are many results that establish conditions for strong duality, often called *constraint qualifications*. One important example are *Slater's conditions*:

Theorem 11.1 (Slater's Theorem)**.** There exists strong duality for the problem

$$\min_{\boldsymbol{\theta}} \; f(\boldsymbol{\theta}) \qquad \text{subject to} \qquad g_{\mathrm{i}}(\boldsymbol{\theta}) \geq 0 \quad \forall\, \mathrm{i} = 1, \ldots, \mathrm{N} \tag{11.9}$$

if f and every g_{i} are convex, and there exists some $\boldsymbol{\theta}$ that satisfies all constraints with strict inequalities.

Proof. See Section 5.3.2 of *Convex Optimization*, by Stephen Boyd and Lieven Vandenberghe, 7th Edition, 2009. □

Since the Slater's conditions in Theorem 11.1 are satisfied for problem (11.3), we conclude that its solution is

$$\boldsymbol{\theta}^\star \;=\; \hat{\boldsymbol{\theta}} \;=\; \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}.$$

11.5 Finding the Maximal Margin Classifier

Recall that our goal is to solve (11.2) to find the affine hyperplane (defined by $\boldsymbol{\theta}^\star$ and θ_0) that best separates our data, i.e., the affine hyperplane with the largest margin. Since maximizing the margin $\frac{1}{\|\boldsymbol{\theta}\|}$ is equivalent to minimizing $\|\boldsymbol{\theta}\|$ or $\|\boldsymbol{\theta}\|^2$, we can rewrite (11.2) as:

$$(\boldsymbol{\theta}^\star, \theta_0) \;:=\; \operatorname*{arg\,min}_{\boldsymbol{\theta}, \theta_0} \; \|\boldsymbol{\theta}\|^2 \qquad \text{subject to} \qquad \mathrm{y}_{\mathrm{i}}(\boldsymbol{\theta}^{\mathsf{T}} \mathbf{x}_{\mathrm{i}} + \theta_0) \geq 1 \quad \forall\, \mathrm{i} = 1, \ldots, \mathrm{N}. \tag{11.10}$$

Notice the striking similarities between (11.10) and the Example problem in (11.3). Defining

$$\begin{aligned} f(\boldsymbol{\theta}) \;&=\; \|\boldsymbol{\theta}\|^2 \;=\; \sum_{\mathrm{j}=1}^{\mathrm{D}} \theta_{\mathrm{j}}^2, \\ g_{\mathrm{i}}(\boldsymbol{\theta}, \mathrm{b}) \;&=\; \mathrm{y}_{\mathrm{i}}(\boldsymbol{\theta}^{\mathsf{T}} \mathbf{x}_{\mathrm{i}} + \mathrm{b}) - 1, \end{aligned}$$

we can see that (11.10) is one instance of the general optimization problem with N constraints in (11.9). So we will solve it by the method of Lagrange multipliers, which instead of (11.10), aims to solve the *dual* problem

$$\max_{\boldsymbol{\lambda} \geq \mathbf{0}} \; \min_{\boldsymbol{\theta}, \theta_0 : \nabla \mathcal{L} = \mathbf{0}} \; \mathcal{L}(\boldsymbol{\theta}, \mathrm{b}, \boldsymbol{\lambda}) \tag{11.11}$$

where $\boldsymbol{\lambda} := [\lambda_1 \;\; \lambda_2 \;\; \cdots \;\; \lambda_{\mathrm{N}} \,]^{\mathsf{T}}$ is the *Lagrange multipliers vector*, also known as *dual variables*, one for each constraint. In this case the Lagrangian is

$$\mathcal{L}(\boldsymbol{\theta}, \mathrm{b}, \boldsymbol{\lambda}) \;=\; \|\boldsymbol{\theta}\|^2 \;-\; \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}} \Big[\mathrm{y}_{\mathrm{i}}(\boldsymbol{\theta}^{\mathsf{T}} \mathbf{x}_{\mathrm{i}} + \mathrm{b}) - 1 \Big],$$

The first step is to find

$$(\boldsymbol{\theta}_{\boldsymbol{\lambda}}, \mathrm{b}_{\boldsymbol{\lambda}}) \;:=\; \operatorname*{arg\,min}_{\boldsymbol{\theta}, \mathrm{b} : \nabla \mathcal{L} = \mathbf{0}} \; \mathcal{L}(\boldsymbol{\theta}, \mathrm{b}, \boldsymbol{\lambda}),$$

Using our standard Optimization 101 approach, we first compute

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta},\mathrm{b},\boldsymbol{\lambda}) &= 2\boldsymbol{\theta} - \sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}\mathbf{x}_{\mathrm{i}},\\ \frac{\mathrm{d}}{\mathrm{db}}\mathcal{L}(\boldsymbol{\theta},\mathrm{b},\boldsymbol{\lambda}) &= -\sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}.\end{aligned}$$

It follows that the constraint $\nabla\mathcal{L}(\boldsymbol{\theta},\mathrm{b},\boldsymbol{\lambda})=\mathbf{0}$ will be met as long as

$$\boldsymbol{\theta}_{\boldsymbol{\lambda}} = \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}\mathbf{x}_{\mathrm{i}} \tag{11.12}$$

$$\boldsymbol{\lambda}^{\mathsf{T}}\mathbf{y} = \sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\mathrm{y}_{\mathrm{i}} = 0, \tag{11.13}$$

where $\mathbf{y}=[\mathrm{y}_1 \;\; \mathrm{y}_2 \;\; \cdots \;\; \mathrm{y}_{\mathrm{N}}]^{\mathsf{T}}$. Using this information, we see that

$$\mathcal{L}(\boldsymbol{\theta}_{\boldsymbol{\lambda}},\mathrm{b}_{\boldsymbol{\lambda}},\boldsymbol{\lambda}) = \sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}} - \frac{1}{4}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\lambda_{\mathrm{j}}\mathrm{y}_{\mathrm{i}}\mathrm{y}_{\mathrm{j}}\mathbf{x}_{\mathrm{i}}^{\mathsf{T}}\mathbf{x}_{\mathrm{j}}.$$

Hence, it all boils down to solving the dual:

$$\hat{\boldsymbol{\lambda}} := \underset{\boldsymbol{\lambda}\geq\mathbf{0}:\nabla\mathcal{L}=\mathbf{0}}{\arg\max}\;\mathcal{L}(\boldsymbol{\theta}_{\boldsymbol{\lambda}},\mathrm{b}_{\boldsymbol{\lambda}}\boldsymbol{\lambda}) = \underset{\boldsymbol{\lambda}\geq\mathbf{0}:\boldsymbol{\lambda}^{\mathsf{T}}\mathbf{y}=0}{\arg\max}\;\sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}} - \frac{1}{4}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\lambda_{\mathrm{j}}\mathrm{y}_{\mathrm{i}}\mathrm{y}_{\mathrm{j}}\langle\mathbf{x}_{\mathrm{i}},\mathbf{x}_{\mathrm{j}}\rangle, \tag{11.14}$$

which we can solve using standard optimization techniques, like gradient descent. Notice that the constraint $\nabla\mathcal{L}(\boldsymbol{\theta},\mathrm{b},\boldsymbol{\lambda})$ is equivalent to (11.12) and (11.13). While the condition (11.12) is no longer relevant in the right hand side of (11.14), the condition (11.13) needs to be *carried*. In general $\hat{\boldsymbol{\lambda}}$ will be a sparse vector. Its non-zero entries correspond to points that lie in the margin, i.e., the support vectors. The solution will not change if the samples corresponding to zero entries in $\hat{\boldsymbol{\lambda}}$ are removed.

Once we have $\hat{\boldsymbol{\lambda}}$, we can plug it back into (11.12) to obtain $\hat{\boldsymbol{\theta}} := \boldsymbol{\theta}_{\hat{\boldsymbol{\lambda}}}$. To recover b, observe that for any support vector $\mathbf{x}_{\mathrm{i}}$ (whose corresponding value $\lambda_{\mathrm{i}}>0$), we have $\mathrm{y}_{\mathrm{i}}(\hat{\boldsymbol{\theta}}^{\mathsf{T}}\mathbf{x}_{\mathrm{i}}+\mathrm{b})=1$, because $\mathbf{x}_{\mathrm{i}}$ lies exactly on the margin. Multiplying both sides by y_{i} we have $\mathrm{y}_{\mathrm{i}}^2(\hat{\boldsymbol{\theta}}^{\mathsf{T}}\mathbf{x}_{\mathrm{i}}+\mathrm{b})=\mathrm{y}_{\mathrm{i}}$, or equivalently, $\hat{\boldsymbol{\theta}}^{\mathsf{T}}\mathbf{x}_{\mathrm{i}}+\mathrm{b}=\mathrm{y}_{\mathrm{i}}$ We thus conclude that

$$\hat{\mathrm{b}} = \mathrm{y}_{\mathrm{i}} - \hat{\boldsymbol{\theta}}^{\mathsf{T}}\mathbf{x}_{\mathrm{i}}.$$

Finally, since problem (11.10) satisfies the Slater's conditions in Theorem 11.1, we conclude that its minimizer $(\boldsymbol{\theta}^{\star},\mathrm{b}^{\star})$ is the pair $(\hat{\boldsymbol{\theta}},\hat{\mathrm{b}})$.

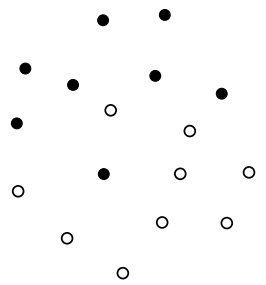
Remark 11.1. Given a new data point $\mathbf{x}$, its class is given by the sign of $\hat{\boldsymbol{\theta}}^{\mathsf{T}}\mathbf{x}+\hat{\mathrm{b}}$, which we can rewrite as

$$\hat{\boldsymbol{\theta}}^{\mathsf{T}}\mathbf{x}+\hat{\mathrm{b}} = \boldsymbol{\theta}_{\boldsymbol{\lambda}}^{\mathsf{T}}\mathbf{x}+\hat{\mathrm{b}} = \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}\mathbf{x}_{\mathrm{i}}^{\mathsf{T}}\mathbf{x}+\hat{\mathrm{b}} = \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}\langle\mathbf{x},\mathbf{x}_{\mathrm{i}}\rangle+\hat{\mathrm{b}}$$

where we can see that the class of $\mathbf{x}$ is determined by its weighted inner product with the support vectors (because for all other vectors $\lambda_{\mathrm{i}}=0$). Hence classification can be done quite efficiently.

11.6 Support Vector Classifier

It is pretty clear that a hyperplane may not classify two classes perfectly:

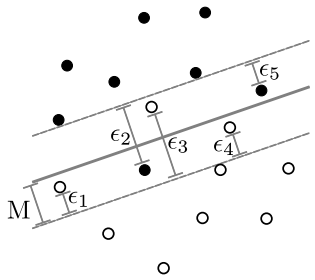


The support vector classifier (SVC), also known as the soft margin classifier, arises as an idea to address this problem. They key idea is to allow some wiggle room by letting some points to be misclassified, using the following optimization instead of (11.10)

$$(\boldsymbol{\theta}^\star, \mathrm{b}^\star) \;:=\; \underset{\boldsymbol{\theta},\mathrm{b},\boldsymbol{\epsilon}\geq\mathbf{0}}{\arg\min}\; \|\boldsymbol{\theta}\|^2 + \mathrm{E}\|\boldsymbol{\epsilon}\|_1 \qquad \text{subject to} \qquad \mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b}) \geq (1-\epsilon_\mathrm{i}) \quad \forall\, \mathrm{i} = 1,\ldots,\mathrm{N}, \tag{11.15}$$

where $\boldsymbol{\epsilon} = [\epsilon_1 \;\; \epsilon_2 \;\; \cdots \;\; \epsilon_\mathrm{N}]^\mathsf{T}$ quantifies the error, a.k.a. *slack*, that will be allowed in each sample, and the tuning parameter $\mathrm{E} \geq 0$ determines the relative importance between maximizing the margin $\frac{1}{\|\boldsymbol{\theta}\|}$ and minimizing the error $\|\boldsymbol{\epsilon}\|_1$. The constraint $\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i} + \mathrm{b}) \geq (1-\epsilon_\mathrm{i})$ ensures that the margin of the i^{th} sample $\mathrm{M_i} = \frac{\mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}_\mathrm{i}+\mathrm{b})}{\|\boldsymbol{\theta}\|}$ is larger than the general margin $\mathrm{M} = \frac{1}{\|\boldsymbol{\theta}\|}$, or maybe *a little* smaller; how *little* is determined by ϵ_i, which should be greater than zero (otherwise the constraint would enforce $\mathrm{M_i}$ to be a little larger than M).

This type of approach is often called a *relaxation*, and will allow some points to be inside the margin, and even on the wrong side of the affine hyperplane:



Using the same techniques as before, we can show that (11.15) can be solved through its dual, given by

$$\hat{\boldsymbol{\lambda}} \;=\; \underset{\mathrm{E}\geq\boldsymbol{\lambda}\geq\mathbf{0}:\boldsymbol{\lambda}^\mathsf{T}\mathbf{y}=0}{\arg\max} \;\; \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_\mathrm{i} - \frac{1}{4}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}} \lambda_\mathrm{i}\lambda_\mathrm{j}\mathrm{y_i}\mathrm{y_j}\langle \mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{j}\rangle. \tag{11.16}$$

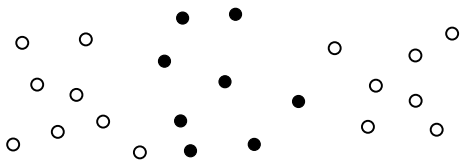
The main difference between MMC and SVC, i.e., between (11.14) and (11.16), is the constraint that $\boldsymbol{\lambda} \leq \mathrm{E}$. The intuition is that without this slack constraint, if $\mathbf{x}_\mathrm{i}$ is misclassified, then $\lambda_\mathrm{i} \to \infty$. The slack constraint limits $\boldsymbol{\lambda}$ so that misclassifications are allowed. In general, a solution to (11.16) will be a sparse vector: its entries equal to E correspond to points on the wrong side of the margin (not necessarily misclassified); its entries smaller than E and larger than zero, i.e., the ones satisfying $\mathrm{E} > \lambda_\mathrm{i} > 0$, correspond to points in the

margin, i.e., the support vectors; its zero entries correspond to all other correctly classified points, whose removal does not change the solution.

Given $\hat{\boldsymbol{\lambda}}$, $\hat{\boldsymbol{\theta}} = \frac{1}{2}\sum_{i=1}^{N} \hat{\lambda}_i y_i \mathbf{x}_i$, as in the MMC case, and $\hat{b} = y_i - \hat{\boldsymbol{\theta}}^\mathsf{T}\mathbf{x}_i$ for any support vector $\mathbf{x}_i$, i.e., one such that $E > \lambda_i > 0$. Given a new data point $\mathbf{x}$, its class is given by the sign of $\hat{\boldsymbol{\theta}}^\mathsf{T}\mathbf{x} + \hat{b}$, same as before.

11.7 Nonlinear Boundaries

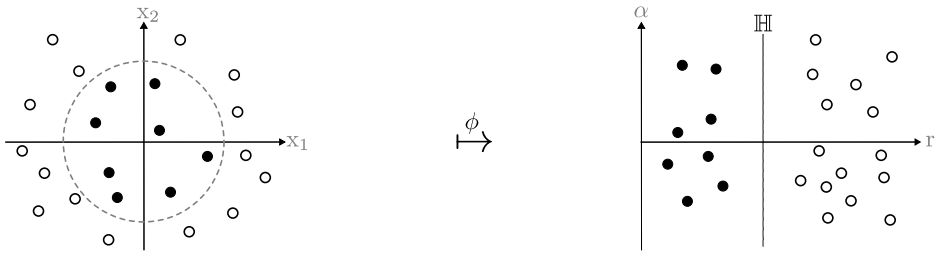
It is pretty clear that not all datasets can be divided by a linear boundary:



One advantage of the MMC and the SVC is that one can easily adapt them to produce nonlinear boundaries using a few (but smart) changes that transform the feature space into a new space where data can be separated with a linear boundary that corresponds to a non-linear boundary in the original space. For example, with the cartesian-to-polar coordinates mapping:

$$\boldsymbol{\phi} \;:\; \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \;\mapsto\; \begin{bmatrix} \sqrt{x_1^2 + x_2^2} \\ \arctan \frac{x_2}{x_1} \end{bmatrix} \;=:\; \begin{bmatrix} r \\ \alpha \end{bmatrix},$$

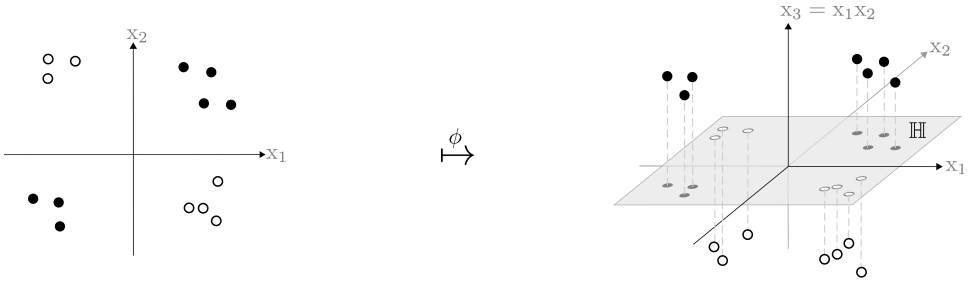
one can use a linear boundary in the transformed space (r, α) to classify data that is non-linearly separable in the original feature space (x_1, x_2):



As another example, consider the following mapping:

$$\boldsymbol{\phi} \;:\; \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \;\mapsto\; \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix} \;=:\; \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

which can be used to *lift* non-linearly separable data to a linearly separable higher-dimensional space:



Given a *feature map* $\boldsymbol{\phi} : \mathbb{R}^{\mathrm{D}} \to \mathbb{R}^{\mathrm{D}'}$, the problem reduces to finding the linear boundary that separates the transformed data, i.e.,

$$(\boldsymbol{\theta}^\star, \mathrm{b}^\star) \;:=\; \underset{\boldsymbol{\theta},\mathrm{b},\boldsymbol{\epsilon}\geq\mathbf{0}}{\arg\min}\; \|\boldsymbol{\theta}\|^2 + \mathrm{E}\|\boldsymbol{\epsilon}\|_1 \qquad \text{subject to} \qquad \mathrm{y_i}(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x_i}) + \mathrm{b}) \geq (1 - \epsilon_\mathrm{i}) \quad \forall\, \mathrm{i} = 1, \ldots, \mathrm{N}. \tag{11.17}$$

Notice that the only difference between (11.17) and (11.15) is that we are replacing $\mathbf{x_i}$ with $\boldsymbol{\phi}(\mathbf{x_i})$, which, depending on $\boldsymbol{\phi}$, may also change the size of $\boldsymbol{\theta}$, but nothing else. Hence the dual of (11.17) is

$$\hat{\boldsymbol{\lambda}} \;=\; \underset{\mathrm{E}\geq\boldsymbol{\lambda}\geq\mathbf{0}:\boldsymbol{\lambda}^\mathsf{T}\mathbf{y}=0}{\arg\max}\; \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_\mathrm{i} - \frac{1}{4}\sum_{\mathrm{i}=1}^{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}} \lambda_\mathrm{i}\lambda_\mathrm{j}\mathrm{y_i}\mathrm{y_j}\langle\boldsymbol{\phi}(\mathbf{x_i}), \boldsymbol{\phi}(\mathbf{x_j})\rangle. \tag{11.18}$$

Given $\hat{\boldsymbol{\lambda}}$, $\hat{\boldsymbol{\theta}} = \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{N}} \hat{\lambda}_\mathrm{i}\mathrm{y_i}\boldsymbol{\phi}(\mathbf{x_i})$, and $\hat{\mathrm{b}} = \mathrm{y_i} - \hat{\boldsymbol{\theta}}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x_i})$ for any support vector $\boldsymbol{\phi}(\mathbf{x_i})$, i.e., one such that $\mathrm{E} > \lambda_\mathrm{i} > 0$. Given a new data point $\mathbf{x}$, its class is given by the sign of $\hat{\boldsymbol{\theta}}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}) + \hat{\mathrm{b}}$.

Now notice that (11.18) only depends on the inner products $\langle\boldsymbol{\phi}(\mathbf{x_i}), \boldsymbol{\phi}(\mathbf{x_j})\rangle$. Hence, to design powerful non-linear boundaries, one does not need to design $\boldsymbol{\phi}$. In fact, one does not even require to explicitly compute $\boldsymbol{\phi}$. It suffices to design and compute its inner product, often called *kernel*:

$$K(\mathbf{x_i}, \mathbf{x_j}) \;:=\; \langle\boldsymbol{\phi}(\mathbf{x_i}), \boldsymbol{\phi}(\mathbf{x_j})\rangle.$$

For example, one may define

$$K(\mathbf{x_i}, \mathbf{x_j}) \;=\; (\mathbf{x_i}^\mathsf{T}\mathbf{x_j} + 1)^\mathrm{d}, \tag{11.19}$$

which would produce a degree-d polynomial. For instance, with $\mathbf{x_i} \in \mathbb{R}^2$ and $\mathrm{d} = 2$, we would get:

$$K(\mathbf{x_i}, \mathbf{x_j}) \;=\; (\mathrm{x_{i1}x_{j1}} + \mathrm{x_{i2}x_{j2}} + 1)^2,$$

from which we can derive its corresponding feature mapping

$$\boldsymbol{\phi}(\mathbf{x}) \;=\; \begin{bmatrix} \mathrm{x}_1^2 & \mathrm{x}_2^2 & \sqrt{2}\mathrm{x}_1\mathrm{x}_2 & \sqrt{2}\mathrm{x}_1 & \sqrt{2}\mathrm{x}_2 & 1 \end{bmatrix}^\mathsf{T}.$$

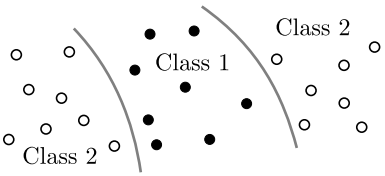
Besides the *polynomial kernel* in (11.19), and the *linear kernel*, which is simply $K(\mathbf{x_i}, \mathbf{x_j}) := \langle\mathbf{x_i}, \mathbf{x_j}\rangle$, another very popular choice is the *radial basis function* (RBF) or *gaussian kernel*:

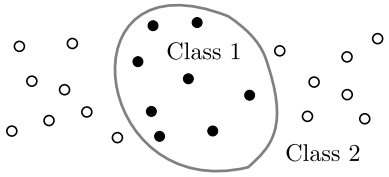
$$K(\mathbf{x_i}, \mathbf{x_j}) \;=\; e^{-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2}, \tag{11.20}$$

where $\gamma \geq 0$ is a parameter, and its corresponding feature mapping is the concatenation of the D infinite-dimensional vectors:

$$\boldsymbol{\phi}(\mathrm{x}) \;=\; e^{-\gamma\mathrm{x}^2}\begin{bmatrix} 1 & \sqrt{\frac{2\gamma}{1!}}\mathrm{x} & \sqrt{\frac{(2\gamma)^2}{2!}}\mathrm{x}^2 & \sqrt{\frac{(2\gamma)^3}{3!}}\mathrm{x}^3 & \cdots \end{bmatrix}^\mathsf{T},$$

which can be derived from the Taylor series expansion of e^x. Depending on the dataset at hand, one kernel may be better than another, and there may be more than one *correct* option.





The *support vector machine* (SVM) is simply a generalization of the SVC that uses a kernel function K and its corresponding feature mapping $\boldsymbol{\phi}$.

11.8 Kernel Algebra

More formally, a kernel $K(\mathbf{x}, \mathbf{x}')$ is a function that quantify the similarity between vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{\mathrm{D}}$, and can be expressed as the inner product $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle$ for some feature mapping $\boldsymbol{\phi} : \mathbb{R}^{\mathrm{D}} \to \mathbb{R}^{\mathrm{D}'}$. The kernels in (11.19) and (11.20) give two examples. In general, one may construct a new kernel from existing ones, because kernels are closed under positive scaling, sum, product, point-wise limit, and composition with a power series.

For example, given two kernel functions $K_1(\mathbf{x}, \mathbf{x}')$, $K_2(\mathbf{x}, \mathbf{x}')$, their linear combination $\mathrm{c}_1 K_1(\mathbf{x}, \mathbf{x}) + \mathrm{c}_2 K_2(\mathbf{x}, \mathbf{x}')$ is also a kernel. Similarly, if $K(\mathbf{x}, \mathbf{x}')$ is a kernel, then so is $e^{K(\mathbf{x}, \mathbf{x}')}$. Given kernels $K_1(\mathbf{x}, \mathbf{x}')$ and $K_2(\mathbf{x}, \mathbf{x}')$ with feature mappings $\boldsymbol{\phi}_1(\mathbf{x})$ and $\boldsymbol{\phi}_2(\mathbf{x})$, the following table outlines some operators that result in valid kernels, and their respective feature mappings.

Kernel composition	Feature mapping
$K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$	$\boldsymbol{\phi}(\mathbf{x}) = [\boldsymbol{\phi}_1^{\mathsf{T}}(\mathbf{x}) \quad \boldsymbol{\phi}_2^{\mathsf{T}}(\mathbf{x})]^{\mathsf{T}}$
$K(\mathbf{x}, \mathbf{x}') = \mathrm{c} K_1(\mathbf{x}, \mathbf{x}')$, $\mathrm{c} > 0$	$\boldsymbol{\phi}(\mathbf{x}) = \sqrt{\mathrm{c}}\boldsymbol{\phi}_1$
$K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') K_2(\mathbf{x}, \mathbf{x}')$	$\phi(\mathbf{x})_\ell = \phi_{1i}(\mathbf{x})\ \phi_{2j}(\mathbf{x})$
$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathsf{T}} \mathbf{A} \mathbf{x}'$, with $\mathbf{A}$ p.s.d.	$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{L}^{\mathsf{T}} \mathbf{x}$, where $\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$
$K(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) f(\mathbf{x}') K_1(\mathbf{x}, \mathbf{x}')$	$\boldsymbol{\phi}(\mathbf{x}) = f(\mathbf{x}) \phi_1(\mathbf{x})$

11.9 Advantages and Disadvantages

- Since the learning task is framed as a convex optimization problem, SVM's can find globally optimal solutions, in contrast to other formulations, like neural networks, which suffer from local minima.
- Classification can be done quite efficiently, using nothing but inner product of the support vectors.
- Kernels provide a powerful option to produce non-linear boundaries.
- Using kernels one may compute distances between high-dimensional feature mappings without explicitly computing the mapping.
- There is a wide range of optimization methods to learn SVM's.
- One SVM can represent only one binary classification task. Multi-class problems require multiple SVM's and additional encoding.

- The empirical performance of SVM's is comparable to the state-of-the-art for many tasks.
- Kernels can be extended for other tasks, like anomaly detection and regression.

11.10 SVM Recipe

To summarize, here is how we would use SVM's in a typical scenario:

- Collect features $\mathbf{x}_\mathrm{i} \in \mathbb{R}^\mathrm{D}$ and labels $\mathrm{y}_\mathrm{i} \in \{-1, 1\}$ from N samples.
- Pick a kernel function $K(\mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{j})$ with its respective feature mapping $\boldsymbol{\phi}(\mathbf{x})$.
- Use your favorite optimization method to solve the dual problem (11.18) to obtain $\hat{\boldsymbol{\lambda}}$.
- Compute $\hat{\boldsymbol{\theta}} = \frac{1}{2} \sum_{\mathrm{i}=1}^{\mathrm{N}} \hat{\lambda}_\mathrm{i} \mathrm{y}_\mathrm{i} \boldsymbol{\phi}(\mathbf{x}_\mathrm{i})$
- Given any i such that $\mathrm{E} > \lambda_\mathrm{i} > 0$, compute $\hat{\mathrm{b}} = \mathrm{y}_\mathrm{i} - \hat{\boldsymbol{\theta}}^\mathsf{T} \boldsymbol{\phi}(\mathbf{x}_\mathrm{i})$
- Given a new sample $\mathbf{x}$, its class is given by the sign of $\hat{\boldsymbol{\theta}}^\mathsf{T} \boldsymbol{\phi}(\mathbf{x}) + \hat{\mathrm{b}}$.