Chapter 3:
Kernel SVM

In the last chapters we studied how to separate data. We mostly ignored the case where the data was not linearly separable but we will come back to it now.

3.1 Kernel Methods

The Kernel Method is a paradigm to efficiently convert linear learning machines into non-linear ones. At a high level, it works the following way:

1. Define, in a clever way, a non-linear map

$$\phi : \mathbb{R}^d \to \mathbb{R}^D,$$

   where $\mathbb{R}^D$ is a very high dimensional space ($D >> d$).

2. Map the inputs into that space,

$$\mathbf{x_i} \mapsto \phi(\mathbf{x_i}), \quad i = 1, \dots n$$

3. Seperate the data linearly in that space

$$f(x) = \text{sign}(\langle \mathbf{w}, \phi(x) \rangle + b)$$

4. This linear seperation in the high dimensional space, corresponds to a non-linear seperation in the input space.

Example: Consider the map :

$$\begin{aligned} \phi :& \mathbb{R}^2 \to \mathbb{R}^3 \\ & (x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2) \end{aligned}$$

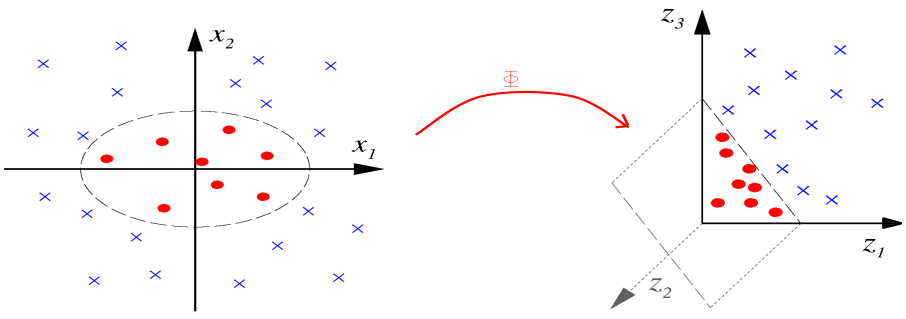

Figure 1: Visualization of the mapping ϕ.

Observation: Notice how in Figure 1 a linear separation in $\mathbb{R}^3$ corresponds to a nonlinear separation in $\mathbb{R}^2$: For instance for the hyperplane $\mathbf{w}^T\mathbf{x}+b$ defined by:

$$\mathbf{w} := \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \text{and} \quad b := -1$$

the corresponding separation for $\mathbf{x} = (x_1, x_2)$ in $\mathbb{R}^2$ would be

$$\mathbf{w}^T\phi(\mathbf{x}) + b = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^T \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} + -1 = x_1^2 + x_2^2 - 1$$

which would be the unit circle. The image space is usually too high to perform operations such as scalar product in it. In our example we can do the following. Let $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x_1}, \tilde{x_2})$. Let us try to compute their inner product in the image space:

$$\phi(\mathbf{x})^T\phi(\tilde{\mathbf{x}}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}^T \begin{pmatrix} \tilde{x_1}^2 \\ \sqrt{2}\tilde{x_1}\tilde{x_2} \\ \tilde{x_2}^2 \end{pmatrix} =$$

$$x_1^2\tilde{x_1}^2 + \sqrt{2}\tilde{x_1}\tilde{x_2}\sqrt{2}x_1x_2 + x_2^2\tilde{x_2}^2 =$$

$$(x_1\tilde{x_1})^2 + 2(x_1\tilde{x_1}x_2\tilde{x_2}) + (x_2\tilde{x_2})^2 =$$

$$(\mathbf{x}^T\tilde{\mathbf{x}})^2.$$

Apparently we can compute the scalar product in the image space by only computing scalar products in $\mathbb{R}^2$. This is an example of the **kernel trick** and $k(\mathbf{x}, \tilde{\mathbf{x}}) := \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^2$ is an example of a kernel.

Definition: Kernel A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called kernel function (or simply "kernel") if all of the following holds

1. it is symmetric
2. there exists a map $\phi : \mathbb{R}^d \to \mathcal{H}$ (called kernel feature map into some high-dimensional kernel feature space $\mathcal{H}$ (e.g., $\mathcal{H} = \mathbb{R}^l$ or $\mathcal{H} = \mathbb{R}^\infty$) such that:
$$\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d : \quad k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle.$$

We can now formalize the Kernel Trick.

Kernel Trick

1. Formulate the (linear) learning machine (training and prediction) solely in terms of inner products.
2. Replace the inner products $\langle \mathbf{x_i}, \mathbf{x_j} \rangle$ by the kernel $k(\mathbf{x_i}, \mathbf{x_j})$

Example: Linear Kernel The linear kernel defined as

$$k(\mathbf{x}, \tilde{\mathbf{x}}) := \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle.$$

is a kernel.
Proof:
Symmetry $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle = \langle \tilde{\mathbf{x}}, \mathbf{x} \rangle = k(\tilde{\mathbf{x}}, \mathbf{x})$.
Mapping Choose the identity map $\phi = id$ Then for all $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d$ we have:

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle = \langle id(\mathbf{x}), id(\tilde{\mathbf{x}}) \rangle = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle. \quad \square$$

Theorem 3.1.1: Closure Properties

1. If k is a kernel and $c \in \mathbb{R}_+$, then ck is a kernel.
2. If k_1 and k_2 are kernels, then $k_1 + k_2$ is a kernel.
3. If k_1 and k_2 are kernels, then $k_1 \cdot k_2$ is a kernel

Definition: Polynomial Kernel
The kernel called polynomial kernel of degree $m \in \mathbb{N}$ is defined as

$$k(\mathbf{x}, \tilde{\mathbf{x}}) := (\langle \mathbf{x}, \tilde{\mathbf{x}} \rangle + c)^m$$

where $c \geq 0$ is a parameter.
We finally define one of the most important kernels in practice.

Definition: Gaussian RBF Kernel
The Gaussian RBF kernel is a kernel defined as

$$k(\mathbf{x}, \tilde{\mathbf{x}}) := \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \tilde{\mathbf{x}}\|^2\right)$$

We call the parameter $\sigma^2 > 0$ is called kernel width (or bandwith).

Definition: Kernel Matrix
Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ be the input data, and let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel function. Then the matrix

$$K := \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \in \mathbb{R}^{n \times n}$$

is called kernel matrix.

Theorem 3.1.2
A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathcal{H}$ is a kernel if and only if for any $n \in \mathbb{N}$ and any input points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ the matrix K is positive semi-definite (meaning $\forall \mathbf{v} \in \mathbb{R}^n : \mathbf{v}^\top K \mathbf{v} \geq 0$)

Theorem 3.1.2 is quite useful for proving functions to be kernels, as seen in the following example.

Example: Proof of Addition Closure Property
Let $\mathbf{x_1}, \ldots, \mathbf{x_n} \in \mathbb{R}^d$ be arbitrary datapoints, k_1, k_2 be two kernels and K_1, K_2 the corresponding kernel matrices. Define $k := k_1 + k_2$. The kernel matrix K of k will look like the following.

$$\begin{aligned} K &= \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \\ &= \begin{pmatrix} k_1(\mathbf{x}_1, \mathbf{x}_1) + k_2(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k_1(\mathbf{x}_1, \mathbf{x}_n) + k_2(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k_1(\mathbf{x}_n, \mathbf{x}_1) + k_2(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k_1(\mathbf{x}_n, \mathbf{x}_n) + k_2(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \\ &= K_1 + K_2 \end{aligned}$$

We know K_1, K_2 are positive semi-definite. We now prove K to be positive semi-definite thus by the theorem also we prove k to be a kernel. Let $\mathbf{v} \in \mathbb{R}^d$ we have:

$$\mathbf{v}^T K \mathbf{v} = \mathbf{v}^T (K_1 + K_2)\mathbf{v} = \mathbf{v}^T (K_1 \mathbf{v} + K_2 \mathbf{v}) = \mathbf{v}^T K_1 \mathbf{v} + \mathbf{v}^T K_2 \mathbf{v} \geq 0. \quad \square$$

3.2 Applying the Kernel Trick to the SVM

Let us recall the unconstrained version of our soft margin svm.

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right)\right)$$

Let $\mathbf{w}^*$ be the optimal solution. We use the fact that $\mathbf{w}^* \in \text{span}(\{\mathbf{x_1}, \ldots \mathbf{x_n}\})$, which is a special case of the general representer theorem which we will prove later on. So there exists $\alpha \in \mathbb{R}^n$ such that

$$\mathbf{w}^* = \sum_{j=1}^{n} \alpha_j \mathbf{x_j}.$$

So we can we also find the optimal solution by optimizing over the possible α. By plugging in we get

$$\min_{b\in\mathbb{R},\alpha\in\mathbb{R}^n} \frac{1}{2}\|\sum_{j=1}^{n}\alpha_j\mathbf{x_j}\|^2 + C\sum_{i=1}^{n}\max\left(0, 1-y_i\left(\left(\sum_{j=1}^{n}\alpha_j\mathbf{x_j}\right)^T\mathbf{x}_i+b\right)\right) = \tag{1}$$

by multiplying out we equivalently get

$$\min_{b\in\mathbb{R},\alpha\in\mathbb{R}^n} \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i\mathbf{x_i}\right)^T\left(\sum_{j=1}^{n}\alpha_j\mathbf{x_j}\right) + C\sum_{i=1}^{n}\max\left(0, 1-y_i\left(\sum_{j=1}^{n}\alpha_j{\mathbf{x_j}}^T\mathbf{x}_i+b\right)\right) = \tag{2}$$

which by multiplying out and reordering in the last sum is equivalent to

$$\min_{b\in\mathbb{R},\alpha\in\mathbb{R}^n} \frac{1}{2}\left(\sum_{i,j=1}^{n}\alpha_i\alpha_j{\mathbf{x_i}}^T\mathbf{x_j}\right) + C\sum_{i=1}^{n}\max\left(0, 1-y_i\left(\sum_{j=1}^{n}\alpha_j{\mathbf{x_i}}^T\mathbf{x}_j+b\right)\right) = \tag{3}$$

As the SVM is used in the higher dimensional space we want to make use of our kernel function we then replace all occurrences of ${\mathbf{x_i}}^\top\mathbf{x}_j$ by $k(\mathbf{x}_i,\mathbf{x}_j)$ to finally get.

$$\min_{b\in\mathbb{R},\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) + C\sum_{i=1}^{n}\max\left(0, 1-y_i\left(\sum_{j=1}^{n}\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)+b\right)\right) \tag{4}$$

We finally get the prediction function:

$$\begin{aligned} f(\mathbf{x}) =& \text{sign}\left({\mathbf{w}^*}^T\phi(\mathbf{x})+b\right) \\ =& \text{sign}\left(\left(\sum_{j=1}^{n}\alpha_j\phi(\mathbf{x_j})\right)^T\phi(\mathbf{x})+b\right) \\ =& \text{sign}\left(\sum_{j=1}^{n}\alpha_j k(\mathbf{x},\mathbf{x}_j)+b\right) \end{aligned}$$

Notice that we actually don't need the feature map ϕ. Now that we have the training and prediction algorithm, it remains to solve the optimization problem (4). For this we use a similiar approach to the stochastic gradient method from Chapter 2.

Algorithm Doubly SGD algorithm for kernel SVM

Input: A starting point $(b.\alpha)_1$

1: **for** $t \leftarrow 1$ to T **do**
2: Randomly select B datapoints
3: Denote their indexes by $J \subset \{1, \ldots, n\}$
4: $$(b,\alpha)_{t+1} := (b,\alpha)_t - \lambda_t \nabla_{(b,\alpha)} \left(\frac{n^2}{2B^2} \sum_{i,j \in J} \alpha_i \alpha_j k\left(\mathbf{x}_i, \mathbf{x}_j\right) + \frac{Cn}{B} \sum_{i \in J} \max\left(0, 1 - y_i \left(\frac{n}{B} \sum_{j \in J} \alpha_j k\left(\mathbf{x}_i, \mathbf{x}_j\right) + b \right)\right)\right)$$
5: **end for**
6: **return** $(b,\alpha)_T$

Observation This is basically the same algorithm as SGD, notice that we are approximating

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j k\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

by

$$\frac{n^2}{2B^2} \sum_{i,j \in J} \alpha_i \alpha_j k\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

as we only calculated the sum for $2B^2$ items, so we need to scale it up. Similiar approximations are done in the rest of the algorithm. Just like in SGD, we are giving up a bit of preciseness to be faster.

In the construction of the Kernel SVM we used the fact that $\mathbf{w}^* \in \text{span}(\{\mathbf{x_1}, \ldots \mathbf{x_n}\})$. We will now prove this using a more general statement.

Theorem 3.2.1 General Representer Theorem
Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel over a Hilbert space $\mathcal{H}$. Let $L : \mathbb{R}^n \to \mathbb{R}$ be any function. Then any solution

$$\mathbf{w}^* \in \arg\min_{\mathbf{w} \in \mathcal{H}} \frac{1}{2}\|\mathbf{w}\|^2 + L\left(\langle \mathbf{w}, \phi\left(\mathbf{x}_1\right)\rangle, \ldots, \langle \mathbf{w}, \phi\left(\mathbf{x}_n\right)\rangle\right)$$

satisfies

$$\text{there exist } \alpha_1, \ldots, \alpha_n \text{ such that } \mathbf{w}^* = \sum_{i=1}^{n} \alpha_i \phi\left(\mathbf{x}_i\right)$$

Proof: Assume the contrary, that is, there exists an optimal solution with:

$$\mathbf{w}^* = \mathbf{w}_{\|} + \underbrace{\mathbf{w}_{\perp}}_{\neq 0}$$

with $\mathbf{w}_{\|} \in \text{span}\left(\phi\left(x_1\right), \ldots, \phi\left(x_n\right)\right)$ and
$\mathbf{w}_{\perp} \notin \text{span}\left(\phi\left(x_1\right), \ldots, \phi\left(x_n\right)\right)$, i.e., $\forall i : \langle \mathbf{w}_{\perp}, \phi\left(\mathbf{x}_i\right)\rangle = 0$

Then, we have, for all $i = 1, \ldots, n$

$$\begin{aligned}\langle \mathbf{w}^*, \phi(\mathbf{x}_i) \rangle &= \langle \mathbf{w}_\parallel + \mathbf{w}_\perp, \phi(\mathbf{x}_i) \rangle = \langle \mathbf{w}_\parallel, \phi(\mathbf{x}_i) \rangle + \underbrace{\langle \mathbf{w}_\perp, \phi(\mathbf{x}_i) \rangle}_{=0} \\ &= \langle \mathbf{w}_\parallel, \phi(\mathbf{x}_i) \rangle\end{aligned}$$

Furthermore, we have:

$$\|\mathbf{w}^*\|^2 = \|\mathbf{w}_\parallel\|^2 + \underbrace{\|\mathbf{w}_\perp\|^2}_{>0} > \|\mathbf{w}_\parallel\|^2$$

Thus the objective function is smaller in the point $\mathbf{w}_\perp$ than it is in the optimum $\mathbf{w}^*$. This contradicts the optimality of $\mathbf{w}^*$. $\square$