

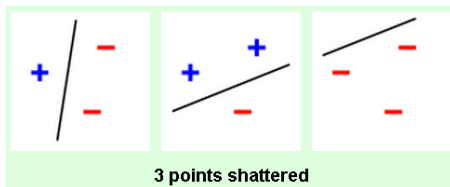
4.1 Kernel Methods

Machine Learning 1: Foundations

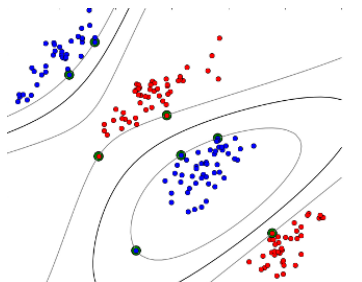
Marius Kloft (TUK)

Recap

In previous lectures: **Linear classification methods**

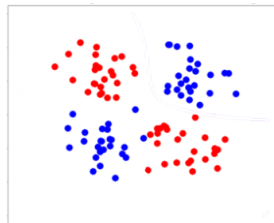


Will not work for non-linear data:



Limitations of Linear Classifiers

For instance, we cannot solve the XOR problem with a linear classifier:



Non-linear algorithm we know so far:
k-nearest neighbor algorithm

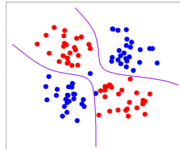
- too simplistic and inaccurate

Other non-linear algorithms we will learn in this course:

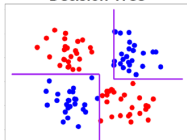
SVM (Gaussian kernel)



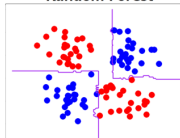
Neural Network



Decision Tree



Random Forest



Contents of this Class

1 Kernel Methods

2 Kernel SVM

1 Kernel Methods

2 Kernel SVM

Kernel Methods

Kernel methods is a paradigm to

- ▶ convert linear learning machines (e.g., linear SVM)
- ▶ into non-linear ones (e.g., kernel SVM)

How does that work?

Core Idea

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

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

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

- 2 Map the inputs into that space,

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

- 3 Separate the data linearly in that space

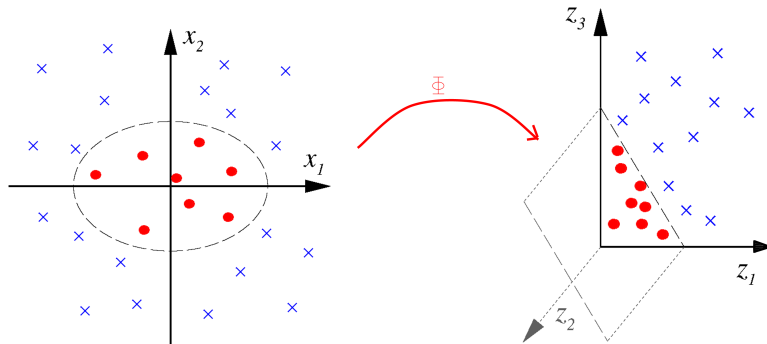
$$f(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b)$$

- 4 Corresponds to non-linear separation in the input space

Do all this very efficiently!

Example

Consider the map $\phi :$

$$\begin{array}{ccc} \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ (x_1, x_2) & \mapsto & (x_1^2, \sqrt{2}x_1x_2, x_2^2) \end{array}$$


Key idea: linear separation in the image space \mathbb{R}^3 corresponds to non-linear separation in the input space \mathbb{R}^2 .

Problem: the image space is usually too high-dimensional to perform any operations in it \Rightarrow **kernel trick**.

Kernel Trick: Example

Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in \mathbb{R}^2 .

- We first map the two points into the higher-dimensional space, using the map $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$.

Now let's try computing their inner product in that space:

$$\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle$$

$$\stackrel{\text{def.}}{=} \left\langle (x_1^2, \sqrt{2}x_1x_2, x_2^2), (\tilde{x}_1^2, \sqrt{2}\tilde{x}_1\tilde{x}_2, \tilde{x}_2^2) \right\rangle$$

$$= (x_1\tilde{x}_1)^2 + 2(x_1\tilde{x}_1x_2\tilde{x}_2) + (x_2\tilde{x}_2)^2$$

$$= \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^2$$

Result: we computed a higher-dimensional inner product via a lower-dimensional one!

$k(\mathbf{x}, \tilde{\mathbf{x}}) := \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^2$ is an example of a **kernel**.

Kernel Trick: Formal Definition

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 $k(\mathbf{x}_i, \mathbf{x}_j)$

Remarks:

- ▶ The kernel trick can be applied only to **linear** learning machines, and not to all of them:
 - ▶ only to those that can be formulated in a way that they access the training data only through inner products between pairs of data points.

Kernel

Definition

A function $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is called **kernel function** (or simply “**kernel**”) if all of the following holds

- 1 it is a symmetric
- 2 there exists a map $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ (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:

$$\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d : k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle .$$

In a nutshell: “k computes inner products in some high-dimensional space”

Additional practical requirement:

- ▶ k should be very efficiently computable!

Example 1: Linear Kernel

Definition

The **linear kernel** is defined as

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

Proposition

The **linear kernel** is a kernel.

Proof

- ① By the symmetry of the inner product, we have, for all $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d$:

$$k(\mathbf{x}, \tilde{\mathbf{x}}) \stackrel{(\star)}{=} \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle = \langle \tilde{\mathbf{x}}, \mathbf{x} \rangle \stackrel{(\star)}{=} k(\tilde{\mathbf{x}}, \mathbf{x}),$$

where (\star) is by the definition of the linear kernel. Thus the linear kernel is symmetric.

- ② We choose the identify map $\phi := \text{id}$ and the kernel feature space $\mathcal{H} := \mathbb{R}^d$. Then, for all $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d$, we have:

$$\begin{aligned} k(\mathbf{x}, \tilde{\mathbf{x}}) &\stackrel{(\star)}{=} \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle \\ &\stackrel{(\star)}{=} \langle \text{id}(\mathbf{x}), \text{id}(\tilde{\mathbf{x}}) \rangle \\ &= \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle, \end{aligned}$$

where again (\star) is by the definition of the linear kernel. □

Example 2/3: Polynomial Kernel

Definition

The **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.

Proposition

The **polynomial kernel** is a kernel.

Proof Idea

One can verify that, for the right choice of the coefficients $c_i \in \mathbb{R}$, the following map is a kernel feature map for the polynomial kernel:

$$\phi : \mathbf{x} \mapsto c_i (x_1^{i_1} \cdots x_d^{i_d})_{i=(i_1, \dots, i_d) \in \mathbb{N}_0^d : \sum_{j=1}^n i_j \leq m} . \quad (1)$$

Full proof: exercise sheet. □

Why the name 'polynomial kernel'?

Recall from Slide 6 that, in kernel methods, we use the classifier

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle . \quad (2)$$

Plugging (1) into (2), we obtain

$$f(\mathbf{x}) = \sum_{i=(i_1, \dots, i_d) \in \mathbb{N}_0^d : \sum_{j=1}^n i_j = m} w_i x_1^{i_1} \cdots x_d^{i_d} .$$

Thus f is a polynomial.

Example 3/3: Gaussian RBF Kernel

Definition

The **Gaussian RBF kernel** is defined as

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

The parameter $\sigma^2 > 0$ is called **kernel width** (or bandwidth).

Note: This is the most widely used kernel function in practice!

Proof: exercise sheet

Oftentimes the Gaussian RBF kernel is simply called 'Gaussian kernel' or 'RBF' kernel.

Properties of Kernels

Theorem

- 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 $k_1 * k_2$ is a kernel.

Proof: is an exercise on the current exercise sheet.

Kernel matrix

Definition

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

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

is called **kernel matrix**.

Equivalent characterization of kernels:

Theorem

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

Conclusion: Kernel Methods

Idea:

- ▶ Map the data into a high-dimensional space and use a simple linear separation there, corresponding to a non-linear classifier in the input space:

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

Kernel trick

- ▶ for the sake of efficiency, perform that mapping only implicitly via a kernel function

Kernelization of a learning machine

- 1 formulate learning machine such that it accesses the data only through inner products between data points
- 2 replace all occurrences of these inner products by kernel

Next week:

How to represent learning machines in terms of scalar products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ & Example of kernelization of SVM.

Further Information (non-mandatory class content)

Full Story: More Precise Definition of Kernels. Sorry!!

Problem:

- ▶ In the definition of kernels, we used the notion of a kernel feature space \mathcal{H} , without defining it!

We said it should:

- ▶ cover the case $\mathcal{H} = \mathbb{R}^d$
- ▶ but allow also for infinite-dimensional spaces \mathbb{R}^{∞} .

Definition

A **Hilbert space** \mathcal{H} is a real vector space together with an inner product such that any Cauchy sequence converges in \mathcal{H} .

Remark:

- ▶ For ML purposes it suffices to think of a Hilbert space as a generalization of \mathbb{R}^d to infinitely many dimensions

Examples of Hilbert spaces:

- ▶ $\mathcal{H} := \mathbb{R}^d$, for any $d \in \mathbb{N}$
- ▶ $\mathcal{H} := \{\mathbf{x} = (x_1, x_2, x_3, \dots) \in \mathbb{R}^{\infty} : \sum_{i=1}^{\infty} x_i^2 < \infty\}$

Definition

A symmetric function $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is called **kernel function** (or simply “**kernel**”) if and only if there exists a map $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ into a Hilbert space \mathcal{H} (called **kernel feature space**) such that

$$\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d : k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle .$$

“k computes inner products in some Hilbert space”