Kernel-Based Learning & Multivariate Modeling

MIRI Master

Lluís A. Belanche
belanche@cs.upc.edu

Soft Computing Research Group

Universitat Politècnica de Catalunya

2019-2020

Kernel-Based Learning & Multivariate Modeling

Syllabus

Sep 10 Introduction to kernel-based learning
Sep 17 The SVM for classification, regression & novelty detection (I)
Oct 01 The SVM for classification, regression & novelty detection (II)
Oct 08 Kernel design (I): theoretical issues
Oct 15 Kernel design (II): practical issues
Oct 22 Kernelizing ML & stats algorithms
Oct 29 Advanced topics

Kernel-Based Learning

Introduction

Desiderata for satisfactory learning methods (my particular view):

Robustness to outliers, errors and/or wrong model assumptions

Stability against variations of the training data samples

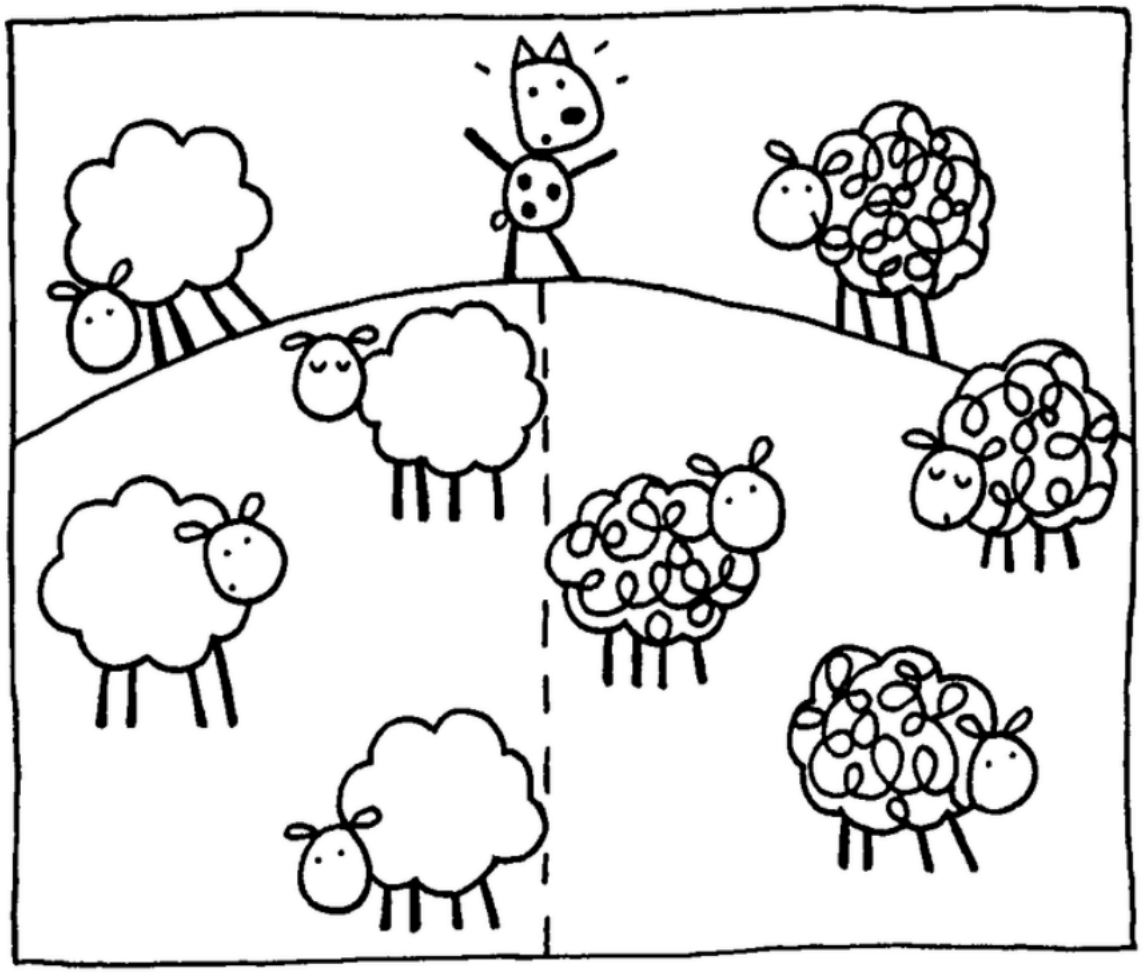
Efficiency in the computational sense (necessary to handle large datasets)

Flexibility to deliver complexity surplus and accept explicit complexity control

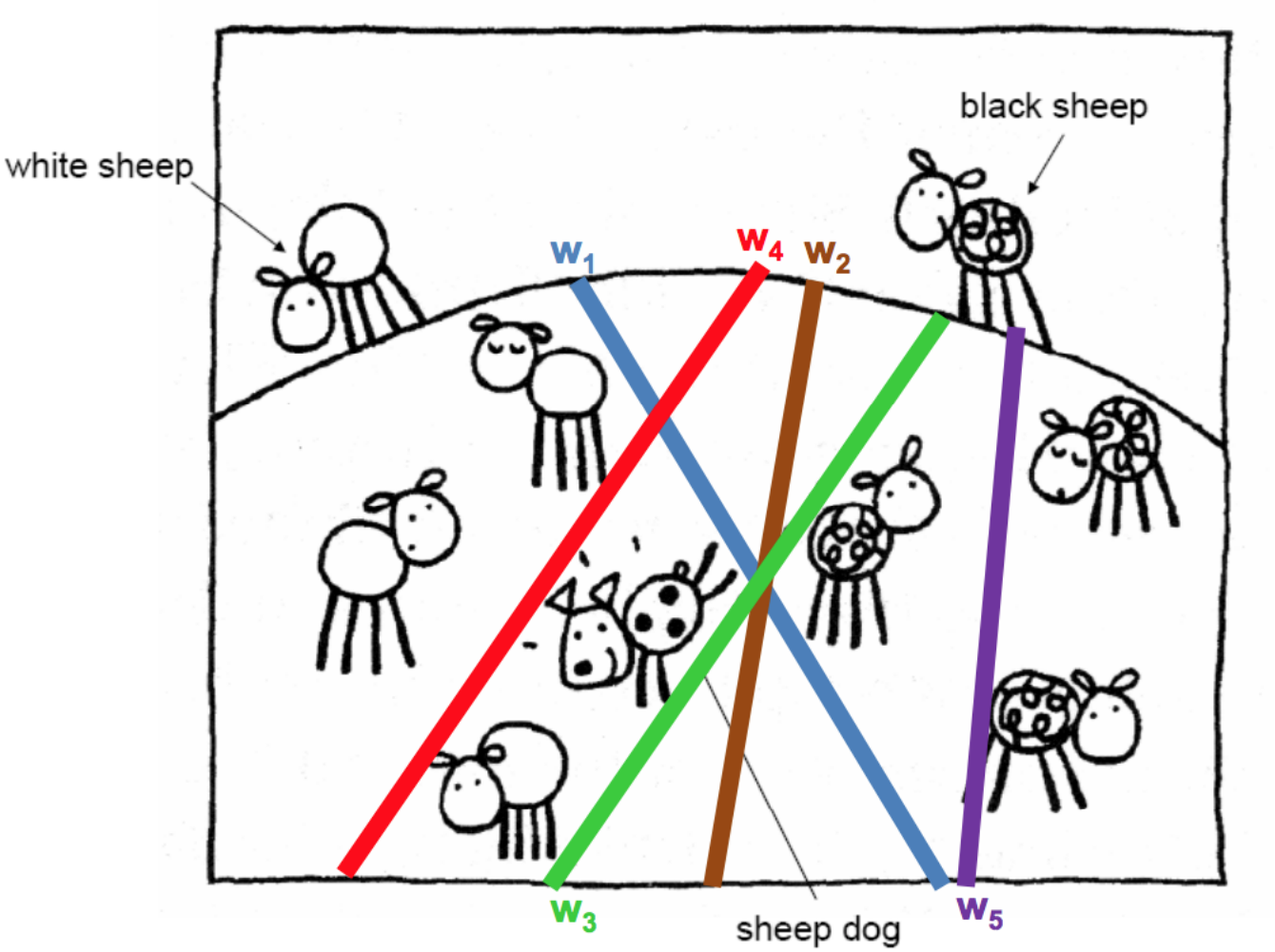
Versatility to accept different data types, perform different tasks and encode prior knowledge

$\Longrightarrow$ **Generalize** well to unseen data (as well as possible)

Kernel-Based Learning



Kernel-Based Learning



Kernel-Based Learning

Introduction

Kernel methods constitute a very powerful paradigm:

- provide a principled way to perform non linear, nonparametric learning;
- rely on solid functional analytic foundations and enjoy optimal statistical properties;
- they have been successfully applied to a wide range of data analysis problems;
- are specially well suited for high-dimensional data, noisy environments or heterogenous information (unlike other approaches like neural networks)

Kernel-Based Learning

Introduction

- the number of hyper-parameters is usually very small compared to other approaches (again unlike neural networks);

- their main limitations are:

  – their difficult applicability in large data scenarios because of stringent computational requirements in terms of time and (especially) memory;

  – the unability to extract increasingly abstract features from the data ("single hidden layer", unlike, yes, neural networks!)

$\longrightarrow$ *Kernel methods are a central tool in machine learning*

Kernel-Based Learning

Key aspects of kernel methods

1. Input data is embedded (mapped) into a **vector space**

2. **Linear relations** are sought among the elements of this vector space

3. The coordinates of the images (mapped data) are *not* needed: only their **pairwise inner products**

4. Often these inner products can be computed (efficiently and implicitly) in the input space (via a **kernel function**) $\rightarrow$ the `PROMISE`

Kernel-Based Learning

Introduction

Kernel-based methods consist of two ingredients:

$$K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix}$$

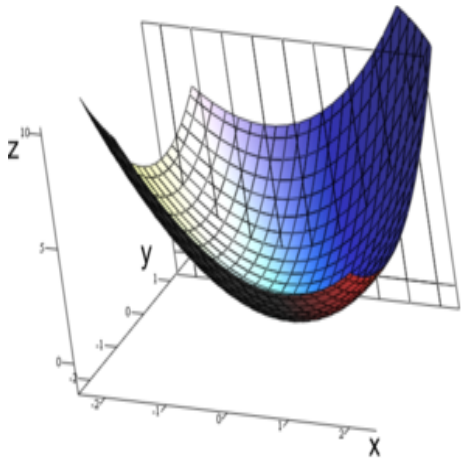
1. The (right) **kernel function** to convert input data into a similarity matrix

2. The **learning algorithm** that uses the kernel matrix and the data to produce a model that performs a prescribed task

Kernel-Based Learning

Introduction

A kernel function is:



1. a special case of similarity measure;

2. a positive semi-definite function;

3. a general inner product in a (special) Hilbert space

It must be psd so that optimization problems can be convex and, therefore, *uniquely* solvable by standard algorithms (moreover, there is a *global* optimum)

Kernel-Based Learning

Introduction

- kernel functions can be used to analyze virtually any kind of data (and to express many forms of previous knowledge) without preprocessing the data

- kernel functions allow the obtention of more general (non-linear) versions of learning algorithms as long as they are base on inner products or Euclidean distances

Kernel-Based Learning

Introduction

Examples of **kernel functions**:

- RBF kernels
- Polynomial kernels
- String kernels
- Anova kernels
- Fisher kernels

Examples of (kernel) **learning algorithms**:

- Support Vector Machine (SVM) and Relevance Vector Machine (RVM)
- kernel Fisher Discriminant Analysis (FDA)
- kernel Principal Components analysis (PCA)
- kernel Canonical Correspondence analysis (CCA)
- kernel (regularized) regression (logistic, linear)
- kernel k-means

Kernel-Based Learning

Introduction

Designing/choosing an appropriate kernel function is not easy in general:

- Consider the **RBF kernel**:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{x} - \boldsymbol{x'}\|^2}{\sigma^2}\right),\ \sigma^2 \in \mathbb{R}$$

- All information of a problem (besides the target values) is tunneled through the kernel matrix $\mathbf{K} = (k_{ij})$, where $k_{ij} := k(\boldsymbol{x}_i, \boldsymbol{x}_j)$

- if σ^2 is very small, then $\mathbf{K} \approx \mathbf{I}$ (all data are dissimilar): over-fitting

- if σ^2 is very large, then $\mathbf{K} \approx \mathbf{J}$ (all data are similar): under-fitting

Kernel-Based Learning

Notation

We have a data sample $D = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_n, t_n)\}$

- $(\boldsymbol{x}_i, t_i)$ are drawn i.i.d. from some unknown (joint) prob. distribution
- $\boldsymbol{x}_i \in \mathcal{X}, t_i \in \mathcal{T}$ (input space, output space)

Kernel-Based Learning

Linear regression

Problem: We wish to find a function $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$ which best models a data set D for the case $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{T} = \mathbb{R}$

- If $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0$, add one dimension $\boldsymbol{x} := (1, \boldsymbol{x}^\top)^\top$; $\boldsymbol{w} := (w_0, \boldsymbol{w}^\top)^\top$
- Call $\mathbf{X}_{n \times d}$ the matrix of the $\boldsymbol{x}_i^\top$ and $\boldsymbol{t} = (t_1, \ldots, t_n)^\top$
- If D is generated as $(\boldsymbol{x}, f(\boldsymbol{x}))$ for some f, the $\boldsymbol{x}_i$ vectors are linearly independent and $n = d$, then there is a unique solution for $\mathbf{X}\boldsymbol{w} = \boldsymbol{t}$ given by $\widehat{\boldsymbol{w}} = \mathbf{X}^{-1}\boldsymbol{t}$; in any other case, the problem is ill-posed

Kernel-Based Learning

Ill-posed & well-posedness

A problem is **ill-posed** if the solution may not always exist, is not uniquely determined or is unstable (small variations in the initial conditions of the problem cause the solution to change quite a lot).

Jacques Hadamard (1865-1963)

- Learning problems are in general ill-posed.
- The solution is to use an **inductive principle**; one of the most popular is the **regularization** principle.

Kernel-Based Learning

Ridge regression

Under the standard assumptions $t_i = f(\boldsymbol{x}_i) + \varepsilon_i$ and $\varepsilon_i \sim N(0, \sigma^2)$, a maximum likelihood argument leads to the minimization of the regularized empirical error (a.k.a penalized likelihood):

$$E_\lambda(\boldsymbol{w}) := \sum_{i=1}^{n} (t_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2 + \lambda \sum_{j=0}^{d} w_j^2 = (\boldsymbol{t} - \mathbf{X}\boldsymbol{w})^\top (\boldsymbol{t} - \mathbf{X}\boldsymbol{w}) + \lambda \boldsymbol{w}^\top \boldsymbol{w}$$

1. Note $E_\lambda(\boldsymbol{w}) = \|\boldsymbol{t} - \mathbf{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$

2. The parameter $\lambda > 0$ defines a trade-off between the fit to the data and the complexity of the model (length of $\boldsymbol{w}$ in this case)

Kernel-Based Learning

Ridge regression: primal representation

Setting $\nabla_{\boldsymbol{w}} E_\lambda = 0$, we obtain the (regularized) normal equations

$$-2\mathbf{X}^\top(\boldsymbol{t} - \mathbf{X}\boldsymbol{w}) + 2\lambda\boldsymbol{w} = 0$$

with solution $\widehat{\boldsymbol{w}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^\top\boldsymbol{t}$

and therefore $f(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^\top\boldsymbol{x} = \boldsymbol{t}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\boldsymbol{x}$.

1. Since $\mathbf{X}$ is $n \times d$, the matrix $\mathbf{X}^\top\mathbf{X}$ is $d \times d$

2. $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_d$ always has an inverse, for all $\lambda > 0$

3. The "model size" does not grow with data size (a **parametric** model)

Kernel-Based Learning

Dual representation

It turns out that the regularized solution can also be written as:

$$\widehat{\boldsymbol{w}} = \sum_{i=1}^{n} \widehat{\alpha}_i \boldsymbol{x}_i$$

In consequence,

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \widehat{\alpha}_i \, \boldsymbol{x}_i^\top \boldsymbol{x}$$

1. The vector of parameters $\widehat{\boldsymbol{\alpha}} = (\widehat{\alpha}_1, \ldots, \widehat{\alpha}_n)^\top$ is $\widehat{\boldsymbol{\alpha}} = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n)^{-1} \boldsymbol{t}$

2. The **Gram matrix** $\mathbf{X}\mathbf{X}^\top$ ("matrix of inner products") is $n \times n$

Kernel-Based Learning

Primal and dual

So we have the **primal** and the **dual** forms for $f(\boldsymbol{x})$:

$$f(\boldsymbol{x}) = \widehat{\boldsymbol{w}}^\top \boldsymbol{x} = \sum_{j=0}^{d} \widehat{w}_j x_j \quad \text{and} \quad f(\boldsymbol{x}) = \sum_{i=1}^{n} \widehat{\alpha}_i \, \boldsymbol{x}_i^\top \boldsymbol{x}$$

The dual form is more convenient when $d >> n$:

- The primal requires the computation and inversion of $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_d$, requiring $O(nd^2 + d^3)$ operations

- The dual requires the computation and inversion of $\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n$, requiring $O(dn^2 + n^3)$ operations

Kernel-Based Learning

How can we perform non-linear regression?

First create a *feature map*, a function $\phi : \mathbb{R}^d \to \mathbb{R}^D$, with $D \in \mathbb{N} \cup \{\infty\}$

$$\boldsymbol{x} \mapsto \phi(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \cdots, \phi_D(\boldsymbol{x}))^\top$$

- $\phi(\boldsymbol{x})$ is called a *feature vector*
- $\{\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^d\}$ is the *feature space* (possibly part of a larger vector space)
- as a technicality, we could easily add $\phi_0(\boldsymbol{x}) = 1$ as before, if desired

Kernel-Based Learning

Hilbert spaces

The right structure for this vector space $\mathcal{H}$ is that of a **Hilbert space**:

David Hilbert (1862-1943)

- A vector space endowed with an **inner product** $\langle , \rangle$ whose associated norm defines a complete metric
- Completeness means that all Cauchy sequences defined in $\mathcal{H}$ converge to an element of $\mathcal{H}$
- Example: the l_2 space of square-summable sequences

(informally) Generalizes the notion of Euclidean space: an infinite-dimensional space with the structure of $\mathbb{R}^d$ (distances, lenghts and angles are well-defined)

Kernel-Based Learning

The regression function has now the primal representation:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle = \sum_{j=1}^{D} w_j \phi_j(\boldsymbol{x})$$

- This feature space has the structure of $\mathbb{R}^D$ (a vector space)

- In consequence, there is also the dual representation:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \widehat{\alpha}_i \, \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle$$

(BTW, how general are all these results?)

Kernel-Based Learning

Feature maps and kernels (1)

Given a feature map $\phi : \mathbb{R}^d \to \mathcal{H}$, being $\mathcal{H}$ a Hilbert space, we define its associated **kernel function** $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \left\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x'}) \right\rangle, \qquad \boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^d$$

One key point is that, for some feature maps, computing $k(\boldsymbol{x}, \boldsymbol{x'})$ is independent of D (the dimension of $\mathcal{H}$)

$\to$ the `PROMISE`!!!

Kernel-Based Learning

Feature maps and kernels (2)

Our regression function has now the dual representation:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \widehat{\alpha}_i \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle = \sum_{i=1}^{n} \widehat{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x})$$

This dual representation:

- ... is a **non-linear model** (in the input space)
- ... is a **linear model** (in the feature space)

We then have a **non-parametric model** (complexity grows with data size)

Kernel-Based Learning

Back to ridge regression ...

The new vector of parameters $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^\top$ is now given by

$$\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \boldsymbol{t},$$

where (as introduced earlier) $\mathbf{K} = (k_{ij})$, with $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

So we can do ridge regression based only on $\mathbf{K}$ (and throw away $\mathbf{X}$)

$\rightarrow$ this is called **Kernel ridge regression** (KRR)

Kernel-Based Learning

Kernel ridge regression

What if we take the (simplest) choice $\phi(\boldsymbol{x}) = \boldsymbol{x}$? In this case $d = D$ and $k(\boldsymbol{x}, \boldsymbol{x'}) = \left\langle \boldsymbol{x}, \boldsymbol{x'} \right\rangle = \boldsymbol{x}^\top \boldsymbol{x'}$. The regularized solution reads:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \widehat{\alpha}_i \boldsymbol{x}_i^\top \boldsymbol{x}$$

where

$$\widehat{\boldsymbol{\alpha}} = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n)^{-1} \boldsymbol{t},$$

(so $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ in this particularly simple case)

This means we have generalized (the dual of) standard ridge regression via a kernel function (we have **kernelized** it!)

Kernel-Based Learning

Kernelizing ...

Many (classical and new) learning algorithms can be kernelized:

1. They require solving a problem where the data appear in the form of pairwise inner products (or pairwise Euclidean distances)

2. The solution is expressed as a linear combination of the kernel function centered at the data (ideally we only use *some* of the data: **sparsity**)

3. Examples include SVMs, ridge regression, perceptrons, FDA, PLS [supervised], as well as PCA, k-means, Parzen Windows [unsupervised]

Kernel-Based Learning

General feature maps

A feature map is of the general form $\phi : \mathcal{X} \to \mathcal{H}$

The associated kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \Big\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x'}) \Big\rangle, \qquad \boldsymbol{x}, \boldsymbol{x'} \in \mathcal{X}$$

$\mathcal{X}$ can be any space, $\mathcal{H}$ is always a (special case of) Hilbert space

---

In our starting development, $\mathcal{X} = \mathbb{R}^d$

Kernel-Based Learning

Regularization-based learning algorithms

The key for this is the **regularization framework**; consider again the regularized empirical error for mapped data:

$$E_\lambda(\boldsymbol{w}) = \sum_{i=1}^{n} (t_i - \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle)^2 + \lambda \|\boldsymbol{w}\|^2, \qquad \lambda > 0$$

which we now generalize to arbitrary **loss** functions $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$:

$$E_\lambda(\boldsymbol{w}) = \sum_{i=1}^{n} L(t_i, \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle) + \lambda \|\boldsymbol{w}\|^2, \qquad \lambda > 0$$

Kernel-Based Learning

Regularization-based learning algorithms

Theorem (informal). If L is differentiable w.r.t. its second argument and $\hat{\boldsymbol{w}}$ is a minimizer of $E_\lambda(\boldsymbol{w})$, then we can represent:

$$\hat{\boldsymbol{w}} = \sum_{i=1}^{n} \hat{\alpha}_i \phi(\boldsymbol{x}_i)$$

and therefore

$$f(\boldsymbol{x}) = \langle \hat{\boldsymbol{w}}, \phi(\boldsymbol{x}) \rangle = \sum_{i=1}^{n} \hat{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x})$$

This result is usually called the **Representer Theorem**. In the case of KRR, $\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \boldsymbol{t}$.