Kernel-Based Learning & Multivariate Modeling

MIRI Master - DMKM Master

Lluís A. Belanche
belanche@cs.upc.edu

Soft Computing Research Group

Universitat Politècnica de Catalunya

2016-2017

Kernel-Based Learning & Multivariate Modeling

Contents by lecture

Sept 14 Introduction to Kernel-Based Learning
Sept 21 The SVM for classification, regression and novelty detection (I)
Sept 28 The SVM for classification, regression and novelty detection (II)
Oct 05 Kernel design (I): theoretical issues
Oct 19 Kernel design (II): practical issues
Oct 26 Kernelizing ML & stats algorithms
Nov 02 Advanced topics

Kernel-Based Learning

Introduction

Desiderata for learning methods (note this is my particular view):

Robust Insensitive to outliers, errors and/or wrong model assumptions

Stable Against variations of the training data samples

Efficient In the computational sense (necessary to handle large datasets)

Effective Flexibility (complexity surplus) & explicit complexity control

Versatile Accept different data types and/or similarity measures

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

Kernel-Based Learning

Introduction

Kernel-based methods consist of two ingredients:

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

2. The **algorithm** that uses the kernels and data to produce a model

Kernel-Based Learning

Introduction

Examples of kernel functions:

- RBF
- Polynomial
- String
- Wavelet
- Fisher

Examples of learning algorithms:

- Support Vector Machine (SVM) and Relevance Vector Machine (RVM)
- kernel Fisher Discriminant Analysis (FDA)
- kernel PCA, kernel CCA
- kernel regression (logistic, ridge)
- kernel k-means

Kernel-Based Learning

Introduction

Relationship between the two ingredients:

- The real art is to pick an appropriate kernel. Consider the RBF kernel:

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

- 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 $\mathbf{K} \approx \mathbf{1}$ (all data are similar): under-fitting

Kernel-Based Learning

Linear regression

Problem: We wish to find a function $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ which best models a data set $S = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\} \subset \mathbb{R}^d \times \mathbb{R}$

- If $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$, just add one dimension $\boldsymbol{x} \leftarrow (1, \boldsymbol{x})$; $\boldsymbol{w} \leftarrow (w_0, \boldsymbol{w})$

- Call $X_{N\times d}$ the matrix of the $\boldsymbol{x}_n$ and $\boldsymbol{t} = (t_1, \ldots, t_N)^T$

- If S is generated as $(\boldsymbol{x}, f(\boldsymbol{x}))$, the $\boldsymbol{x}_n$ vectors are linearly independent and $N = d$, then there is a unique solution for $X\boldsymbol{w} = \boldsymbol{t}$ given by $\boldsymbol{w}^* = X^{-1}\boldsymbol{t}$. 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 $t = f(\boldsymbol{x}) + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$,
ML leads to the minimization of the regularized (= penalized) empirical error:

$$E_\lambda(\boldsymbol{w}) = \sum_{n=1}^{N} (t_n - \boldsymbol{w}^T \boldsymbol{x}_n)^2 + \lambda \sum_{i=0}^{d} w_i^2 = (\boldsymbol{t} - X\boldsymbol{w})^T (\boldsymbol{t} - X\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

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

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

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

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

with solution $\boldsymbol{w}^* = (X^TX + \lambda I_d)^{-1}X^T\boldsymbol{t}$

and therefore $f(\boldsymbol{x}) = (\boldsymbol{w}^*)^T\boldsymbol{x} = \langle \boldsymbol{w}^*, \boldsymbol{x}\rangle = \boldsymbol{t}^TX(X^TX + \lambda I_d)^{-1}\boldsymbol{x}$.

Since X is $N \times d$, the matrix X^TX is $d \times d$

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

2. $X^TX + \lambda I_d$ always has an inverse, for all $\lambda > 0$

Kernel-Based Learning

Dual representation

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

$$\boldsymbol{w}^* = \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n$$

In consequence,

$$f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle$$

1. The vector of parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^T$ is $\boldsymbol{\alpha} = (XX^T + \lambda I_N)^{-1}\boldsymbol{t}$

2. The **Gram matrix** XX^T ("matrix of inner products") is $N \times N$

Kernel-Based Learning

Primal and dual

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

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}^*, \boldsymbol{x} \rangle = \sum_{i=0}^{d} w_i^* x_i \quad \text{and} \quad f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle$$

The dual form is (apparently) more convenient when $d >> N$:

- The primal requires the computation and inversion of $X^T X + \lambda I_d$, requiring $O(Nd^2 + d^3)$ operations

- The dual requires the computation and inversion of $XX^T + \lambda I_N$, requiring $O(dN^2 + N^3)$ operations

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

Feature maps (1)

How can we perform **non-linear** function regression?

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

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

$\phi(\boldsymbol{x})$ is called the *feature vector* and $\{\phi(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^d\}$ is the *feature space* (possibly part of a larger vector space $\mathcal{H}$), and typically $D >> d$.

(as a technicality, we could easily define $\phi_0(\boldsymbol{x}) = 1$ as before, if need be)

Kernel-Based Learning

Feature maps (2)

The non-linear 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:

$$f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \phi(\boldsymbol{x}_n), \phi(\boldsymbol{x}) \rangle$$

(how general is this result???)

Kernel-Based Learning

Feature maps (3)

We will see that it suffices for $\mathcal{H}$ to have the structure of a **Hilbert space**:

- A vector space endowed with an **inner product** whose associated norm defines a complete metric
- Distances, lenghts and angles are well-defined for the elements of the space
- Completeness means that all Cauchy sequences defined in $\mathcal{H}$ converge to an element of $\mathcal{H}$

David Hilbert (1862-1943)

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:

$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{v}) \rangle , \qquad \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$$

One key point is that, for some feature maps, computing $k(\boldsymbol{u}, \boldsymbol{v})$ is independent of the dimension of $\mathcal{H}$ (it only depends on d).

$\to$ the PROMISE!!!

Kernel-Based Learning

Feature maps and kernels (2)

Our regression function has now the dual representation:

$$f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \phi(\boldsymbol{x}_n), \phi(\boldsymbol{x}) \rangle = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

This dual representation:

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

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

Kernel-Based Learning

Feature maps and kernels (3)

The new vector of parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^T$ is now given by

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

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 X)

Kernel-Based Learning

Kernel ridge regression

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

$$f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle$$

where

$$\boldsymbol{\alpha} = (XX^T + \lambda I_N)^{-1} \boldsymbol{t},$$

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

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

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 (4)

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:

$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \phi(\boldsymbol{u}), \phi(\boldsymbol{v}) \rangle , \qquad \boldsymbol{u}, \boldsymbol{v} \in \mathcal{X}$$

$\mathcal{X}$ can be any space, $\mathcal{H}$ is always a 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 the regularized empirical error for mapped data:

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

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

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

Kernel-Based Learning

Regularization-based learning algorithms

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

$$\boldsymbol{w}^* = \sum_{n=1}^{N} \alpha_n \phi(\boldsymbol{x}_n)$$

and therefore

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}^*, \phi(\boldsymbol{x}) \rangle = \sum_{j=1}^{D} w_j^* \phi_j(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}), \qquad \boldsymbol{\alpha} = (\mathbf{K} + \lambda I_N)^{-1} \boldsymbol{t}$$

This result is usually called the **Representer Theorem**.