Aprenentatge Automàtic 2

GCED

Lluís A. Belanche

belanche@cs.upc.edu

Soft Computing Research Group
Dept. de Ciències de la Computació (Computer Science)
Universitat Politècnica de Catalunya

2021-2022

LECTURE 5: Kernels redux: definitions, properties, examples in $\mathbb{R}^p$. Kernel design for data objects not in $\mathbb{R}^p$.

Kernel design: practical issues

Euclidean space $\mathbb{R}^p$, but not only ...

- Kernels on real vectors (whole families)
- Kernels on binary vectors (bitstrings = sets)
- General structured kernels:
  - All-subsets kernel
  - Convolution kernels
- Kernels on discrete structures:
  - Tree kernels
  - Graph kernels
- Kernels on distributions (generative kernels):
  - P-kernels
  - Marginalized kernels
- String kernels (text)

... and many others (functional data, categorical data, permutations, ...)

Kernel design: practical issues

All-subsets kernel

Consider a feature space with one feature for every subset $A \subseteq \{1, \ldots, d\}$ of the input variables:

For $\boldsymbol{x} \in \mathbb{R}^p$, feature A is given by $\phi_A(\boldsymbol{x}) := \prod_{i \in A} x_i$ (note $\phi_\emptyset(\boldsymbol{x}) = 1$)

The kernel is defined by the mapping $\phi : \boldsymbol{x} \to (\phi_A(\boldsymbol{x}))_{A \subseteq \{1,\ldots,d\}}$

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \left\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x'}) \right\rangle = \sum_{A \subseteq \{1,\ldots,d\}} \phi_A(\boldsymbol{x}) \phi_A(\boldsymbol{x'})$$

$$= \sum_{A \subseteq \{1,\ldots,d\}} \prod_{i \in A} x_i x'_i = \prod_{i=1}^{d} (1 + x_i x'_i)$$

The last step is obtained by expanding $(1 + x_1 x'_1)(1 + x_2 x'_2) \ldots (1 + x_d x'_d)$

Kernel design: practical issues

All-subsets kernel

We have the freedom to downplay some features (and thus emphasize others) by introducing weighting factors $w_i \geq 0$ for each feature i:

$$\phi_A(\boldsymbol{x}) := \prod_{i \in A} \sqrt{w_i} x_i$$

therefore

$$k_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{x'}) = \prod_{i=1}^{d} (1 + w_i x_i x'_i)$$

Kernel design: practical issues

Bitstring/Binary variables/Sets

Let $\boldsymbol{x}, \boldsymbol{x'} \in \{0,1\}^p$, representing absence/presence of a binary trait:

1. The Simple Matching Coefficient (SMC) is the fraction of $1-1$ matches, and it is a kernel on $\{0,1\}^p$. ¶

2. The Jaccard Coefficient is the fraction of $1-1$ matches among the traits present in either data vector, and it is a kernel on $\{0,1\}^p$. ¶

Kernel design: practical issues

Bitstring/Binary variables/Sets

- Given two sets $A, B \subset U$, where U is finite, consider

$$k(A,B) := \frac{1}{|U|} \sum_{a \in A} \sum_{b \in B} k_{\text{base}}(a,b)$$

- If k_{base} is the overlap kernel $k(a,b) = \begin{cases} 1 & \text{if } a = b; \\ 0 & \text{otherwise.} \end{cases}$

  we get $k(A,B) = \frac{|A \cap B|}{|U|}$, the equivalent to the SMC.

- The equivalent to the Jaccard kernel would be $k(A,B) = \frac{|A \cap B|}{|A \cup B|}$.

Kernel design: practical issues

Generative kernels (I)

Given a conditional probability distribution on $\mathcal{X}|_{\mathcal{Z}}$, we can compare data points by assigning a high value if both have high (conditional) probability:

$$k(\boldsymbol{x}, \boldsymbol{x'}) := \sum_{z \in \mathcal{Z}} p(\boldsymbol{x}|z) p(\boldsymbol{x'}|z) P(z) \qquad \text{discrete case}$$

$$k(\boldsymbol{x}, \boldsymbol{x'}) := \int_{\mathcal{Z}} p(\boldsymbol{x}|z) p(\boldsymbol{x'}|z) p(z)\, dz \qquad \text{continuous case}$$

The feature maps are $(\phi(\boldsymbol{x}))_z = p(\boldsymbol{x}|z)\sqrt{p(z)}$

Hint: $p(\boldsymbol{x}, \boldsymbol{x'}, z) = p(\boldsymbol{x}, \boldsymbol{x'}|z)p(z) = p(\boldsymbol{x}|z)p(\boldsymbol{x'}|z)p(z)$

Kernel design: practical issues

Generative kernels (II)

Given a probability distribution on $\mathcal{X} \times \mathcal{Z}$, and a kernel on $\mathcal{X} \times \mathcal{Z}$ pairs, we can define:

$$k(\boldsymbol{x}, \boldsymbol{x'}) := \sum_{z} \sum_{z'} k\Big((\boldsymbol{x}, z), (\boldsymbol{x'}, z')\Big) p(\boldsymbol{x}|z) p(\boldsymbol{x'}|z')$$

Typical applications of generative kernels are found in **graphical models**:

- $\mathcal{X}$ are the **observed** variables and $\mathcal{Z}$ are the **hidden** (latent) variables
- A kernel for the observed ones is obtained by taking the expectation w.r.t. the hidden ones (*marginalizing* them away)
- Applications: HMMs for DNA sequences, or stochastic context-free grammars for RNA sequences –see e.g. "Kernel methods in genomics and computational biology" by J.P. Vert

Kernel design: practical issues

The Spectrum (aka n-Gram) kernel

- Let Σ be a finite alphabet: an n-Gram is a block of n adjacent characters in Σ

  Define $k(\boldsymbol{x}, \boldsymbol{x'}) := \sum_{s \in \Sigma^n} |s \in \boldsymbol{x}| \cdot |s \in \boldsymbol{x'}|$

Example: Word `aababc` in alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, $n = 2$:

aa	ab	ac	ba	bb	bc ...
1	2	0	1	0	1 ...

- The feature vectors are sparse and the kernel can be computed in $O(|\boldsymbol{x}| + |\boldsymbol{x'}|)$ time and memory

Kernel design: practical issues

Kernels from graphs

- Consider a graph $G = (V, E)$, where the set of vertices (nodes) V are the data points and E is the set of edges. Call $N = |V|$, the number of nodes
- The idea is to compute a (base) matrix $S_{N \times N}$ whose entries are the weights of the edges and consider $S^2 = SS$ (S is only assumed to be symmetric)
- Typical use: **connectivity matrix** of G: the (i, j) element of S^2 is the number of paths of length exactly 2 between i and j

---

Examples:

1. protein-protein interactions
2. people-to-people interactions

In 2, the (i, j) element of S^2 is the number of common friends between data points i and j (it can be thought of as a measure of their similarity)

Kernel design: practical issues

Kernels from graphs

Notes:

- The entries of S may be real-valued numbers (e.g., symmetric bounded similarities)
- Higher powers of S measure higher-order similarities
- Only the even powers are guaranteed to be PSD

Consider, for a given $\lambda \in (0, 1)$:

$$\sum_{k=0}^{\infty} \frac{1}{k!} \lambda^k S^k =: \exp(\lambda S)$$

1. If S is symmetric, then $S = U \Lambda U^T$ (spectral decomposition), so $S^2 = (U \Lambda U^T)(U \Lambda U^T) = U \Lambda^2 U^T$.
2. In general, we have $S^k = U \Lambda^k U^T$ and therefore:

$$K := \exp(\lambda S) = U \exp(\lambda \Lambda) U^T$$

   is an example of a **diffusion** kernel.

Example in a real application domain

Handling missing values in microbiology

- Modern modelling problems are difficult for a number of reasons, including the challenge of dealing with a significant amount of missing information

- **Missing values** almost always represent a serious problem because they force to preprocess the dataset and a good deal of effort is normally put in this part of the modelling

- In order to process such datasets with kernel methods, an imputation procedure is then deemed a necessary but demanding step

Example in a real application domain

Handling missing values in microbiology

- The study of fecal source pollution in waterbodies is a major problem in ensuring the welfare of human populations

- **Microbial source tracking** (MST) methods attempt to identify the source of contamination, allowing for improved risk analysis and better water management

- The available dataset includes 148 observations about 10 chemical, microbial, and eukaryotic markers of fecal pollution in water

- All variables (except the class variable) are binary, i.e., they signal the presence or absence of a particular marker

Example in a real application domain

Handling missing values in microbiology

Origin	HF183	HF134	CF128	Humito	Pomito	Bomito	ADO	DEN
Human :50	0 :68	0 :81	0 :104	0 :35	0 :83	0 :78	0 :56	0 :80
Cow :26	1 :40	1 :26	1 : 5	1 :79	1 :32	1 :32	1 :59	1 :34
Poultry:31	**?**:31	**?**:32	**?**:30	**?**:25	**?**:24	**?**:29	**?**:24	**?**:25
Pig :32								

Summary (counts) table for the full dataset. The first column is the target class. The symbol **?** denotes a missing value.

The percentage of missing values is around 19.8 %, and all the predictive variables have percentages between 17 % and 23 %

Example in a real application domain

Handling missing values in microbiology

Theorem. Let the symbol **?** denote a missing element, for which only equality is defined, and $\mathcal{X}$ a finite discrete set. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel in $\mathcal{X}$ and P a probability mass function in $\mathcal{X}$. Then the function $k^{\mathbf{?}}(x, y)$ given by

$$k^{\mathbf{?}}(x,y) := \begin{cases} k(x,y), & \text{if } x, y \neq \mathbf{?} \ ; \\ g(x) := \sum\limits_{y' \in X} P(y') k(x,y'), & \text{if } x \neq \mathbf{?} \text{ and } y = \mathbf{?}; \\ g(y) := \sum\limits_{x' \in X} P(x') k(x',y), & \text{if } x = \mathbf{?} \text{ and } y \neq \mathbf{?}; \\ G := \sum\limits_{x' \in X} P(x') \sum\limits_{y' \in X} P(y') k(x',y'), & \text{if } x = y = \mathbf{?} \end{cases}$$

is a kernel in $\mathcal{X} \cup \{\mathbf{?}\}$.

Example in a real application domain

Handling missing values in microbiology

For the particular case of binary variables $x, y \in \{v_1, v_2\}$, a convenient approach is to define the kernel:

$$k_{0/1}(x, y) := \mathbb{I}_{\{x=y\}}$$

where

$$\mathbb{I}_{\{z\}} = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{if } z \text{ is false} \end{cases}$$

Example in a real application domain

Handling missing values in microbiology

Consider now $\boldsymbol{x}, \boldsymbol{y} \in \{0, 1, \textbf{?}\}^p$. When we apply the Theorem to this kernel, we obtain an extended multivariate kernel:

$$\mathcal{K}_1(\boldsymbol{x}, \boldsymbol{y}) := \frac{1}{p} \sum_{i=1}^{p} \begin{cases} 1 & \text{if } x_i = y_i = 1 \ ; \\ P_i(x_i), & \text{if } x_i \neq \textbf{?} \text{ and } y_i = \textbf{?}; \\ P_i(y_i), & \text{if } x_i = \textbf{?} \text{ and } y_i \neq \textbf{?}; \\ (P_i(0))^2 + (P_i(1))^2, & \text{if } x_i = y_i = \textbf{?}; \\ 0, & \text{otherwise} \end{cases}$$

This kernel is a generalization of the classical *Simple Matching Coefficient*, proposed by Sokal and Michener for numerical taxonomy

Example in a real application domain

Handling missing values in microbiology

Alternatives???

Given $\boldsymbol{x}, \boldsymbol{y} \in$.

Let $c(\boldsymbol{x})$ be the set of completions of $\boldsymbol{x}$. Given two vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0, 1, \textbf{?}\}^p$, the function

$$\mathcal{K}_2(\boldsymbol{x}, \boldsymbol{y}) := \frac{1}{|c(\boldsymbol{x})||c(\boldsymbol{y})|} \sum_{\boldsymbol{x}' \in c(\boldsymbol{x})} \sum_{\boldsymbol{y}' \in c(\boldsymbol{y})} k(\boldsymbol{x}', \boldsymbol{y}')$$

is a kernel in $\{0, 1, \textbf{?}\}^p$.

Example in a real application domain

Handling missing values in microbiology

			10x10cv for each class			
Approach	C	10x10cv	Human	Cow	Poultry	Swine
$\mathcal{K}_1$	2.0	79.3	95.4	64.5	75.2	69.4
$\mathcal{K}_2$	1.6	78.2	92.6	62.8	71.8	74.2
MI-1	1.0	79.9	92.7	66.4	69.4	80.2
MI-2	1.0	79.0	94.5	57.5	70.8	78.8

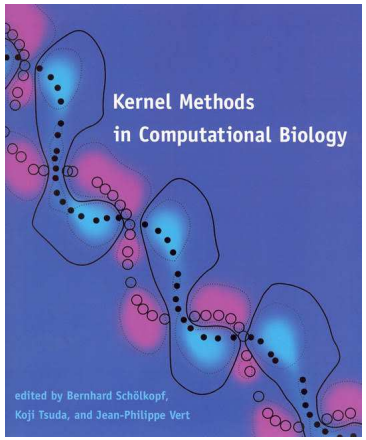
Mean 10x10cv accuracies for the four approaches to handle missing values. Also shown are best cost parameter C and detailed class performance.

(*joint work with G. Nebot, T. Aluja and V. Kobayashi*)

Kernel design: practical issues

More Kernels!

Kernels abound in computational biology and computational chemistry (e.g., phylogenetic profiles, protein 3D structures)



Example: the prediction of **interacting proteins** to reconstruct an interaction network can be posed as a binary classification problem: given a pair of proteins, do they interact or not?

$\longrightarrow$ we need kernel between *pairs* of proteins!

Kernel design: practical issues

More Kernels!

The available data is about each single protein; it is then natural to derive kernels for **pairs** of proteins k_{pair} from any kernel k for **single** proteins:

$$k_{\text{pair}}\Big((A,B),(C,D)\Big) := k(A,C)k(B,D) + k(A,D)k(B,C)$$

(there is usually no order in a protein pair, so we try both matches)

· Using Product Kernels to Predict Protein Interactions. *Advances in Biochemical Engineering/Biotechnology* (110), pp 215-245 (2007)
· Kernel methods for predicting protein-protein interactions. *Bioinformatics*. 2005

Kernel design: practical issues

Conclusions

- The power of kernel methods partly relies in the ability to process virtually any sort of data as soon as a valid kernel is defined
- Importance of designing kernels that do not constitute explicit inner products between objects, and therefore fully exploit the kernel trick
- Possibility of learning the kernel function (or the kernel matrix) from the training data
- Theoretical analyses are needed on the implications of the kernel choice for the success of specific kernel-based methods