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 1: Introduction to Kernel-based Machine Learning

Kernel-based learning

Desiderata for satisfactory learning methods

Robustness to outliers, errors and/or wrong model assumptions

Efficiency in the computational sense (necessary to handle large data-sets)

Flexibility to perform different tasks

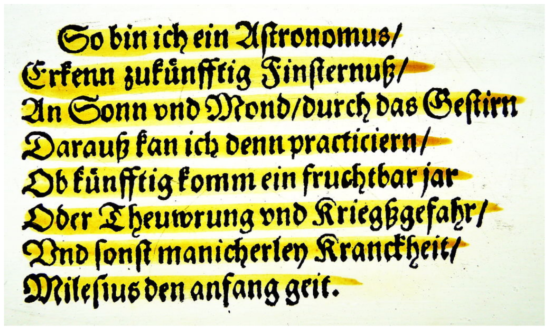
Controlable non-linearity to deliver complexity surplus and accept explicit complexity control

Versatility to accept different data types and incorporate prior knowledge

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

Kernel-Based Learning

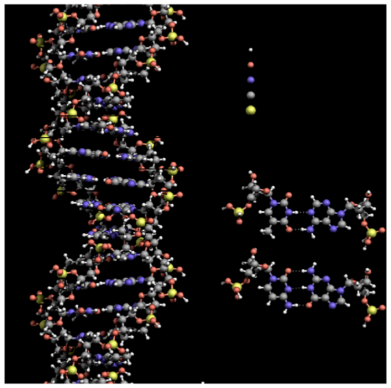
Modern data types



So bin ich ein Aſtronomus/
Erkenn zukünfftig Finſternuß/
An Sonn vnd Mond/durch das Geſtirn
Darauß kan ich denn practiciern/
Ob künfftig komm ein fruchtbar jar
Oder Theuwrung vnd Kriegßgefahr/
Vnd ſonſt manicherley Kranckheit/
Mileſius den anfang geit.

DOCUMENTS

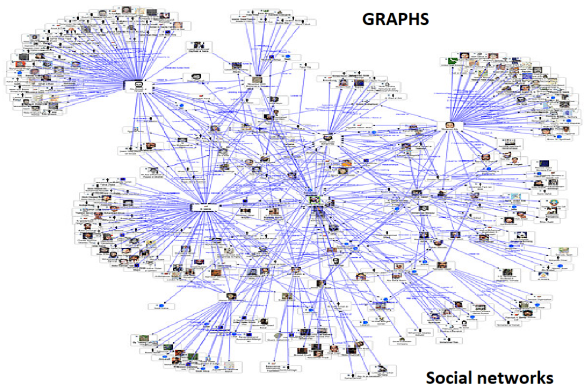
Textual data



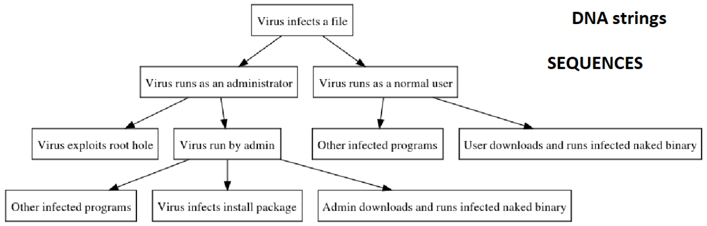
DNA strings

SEQUENCES

GRAPHS

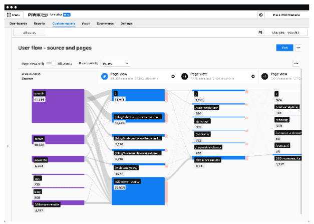


Social networks

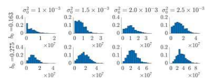

Virus infects a file
Virus runs as an administrator
Virus runs as a normal user
Virus exploits root hole
Virus run by admin
Other infected programs
User downloads and runs infected naked binary
Other infected programs
Virus infects install package
Admin downloads and runs infected naked binary

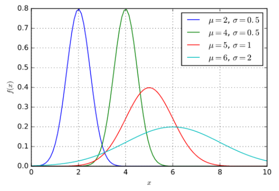
Tree data

HIERARCHIES



Web traffic (click chains)





What else ???

Probability distributions

Kernel-Based Learning

Introduction

The **standard way** of representing and processing data by learning machines. $\mathcal{X}$ is the input space; $S(\mathcal{X}) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is a finite iid sample of objects in $\mathcal{X}$ (dataset from reality/Nature)

1. Find a (computer) suitable representation $\phi : \mathcal{X} \to \mathcal{F}$ for the data objects

2. Create the new dataset S, ready for being analyzed by an algorithm, as:

$$S := \{\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)\}$$

Kernel-based learning

Introduction

The **kernel methods** way of representing and processing data by learning machines. $\mathcal{X}$ and $S(\mathcal{X})$ as before

1. Choose a comparison function (a.k.a. **similarity measure**) $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for the data objects (called a **kernel function**)

2. Create the matrix $\mathbf{K}_{n\times n}$ (called a **kernel matrix**), ready for being analyzed by a kernelized algorithm, as:

$$\mathbf{K} := (k(\boldsymbol{x}_i, \boldsymbol{x}_j)), i, j = 1, \ldots, n$$

Kernel-based learning

Introduction

Example: we want to work with **Oligonucleotides** (short DNA molecules, formed as chains of linked units called nucleotides)

$\mathcal{X}$ is the set of all oligonucleotides, $S(\mathcal{X})$ a sample of n oligonucleotides, $\phi : \mathcal{X} \to \mathcal{F}$ maps every oligonucleotide to an element of the set of all finite sequences of nucleotides $\{A, G, C, T\}$.

<u>EXAMPLE</u>: for some $\boldsymbol{x} \in \mathcal{X}, \phi(\mathcal{X}) = AGTCCAT$.

A learning algorithm then could either:

1. Process the dataset $\{AGTCCAT, CCACG, \dots\}$ directly; or

2. Transform (a.k.a preprocess) the dataset into a more suitable representation (needed as a consequence of the choice of learning algorithm), as $\Gamma : \mathcal{F} \to \mathbb{R}^p$ (e.g., most neural networks)

Kernel-based learning

Introduction

In kernel methods we would have $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $(s,t) \to k(s,t)$ is a comparison function between two oligonucleotides.

Then we would create $\mathbf{K}_{n\times n}$:

$$\mathbf{K} = \begin{pmatrix} k(\boldsymbol{x}_1,\boldsymbol{x}_1) & k(\boldsymbol{x}_1,\boldsymbol{x}_2) & \dots \\ \vdots & \ddots & \\ k(\boldsymbol{x}_n,\boldsymbol{x}_1) & & k(\boldsymbol{x}_n,\boldsymbol{x}_n) \end{pmatrix} \tag{1}$$

... a **kernel method** is any learning algorithm that processes **K** (i.e. takes **K** as input)

Kernel-based learning

Introduction

Notes:

1. The representation as a kernel matrix does not depend on the nature of the data: $\mathcal{X}$ could be virtually anything: sheep cartoons, Shakespeare texts, bird sounds, ... This implies a modularity between the learning algorithm and the comparison function

2. The size of $\mathbf{K}_{n\times n}$ scales with n. This is very useful in problems where $n \ll p$ (as in Computational biology); for example: $n = 50$ tissues, $p = 15,000$ genes implies a $\mathbf{K}_{50\times 50}$ matrix

3. Some (many?) times, it is the case that the coding function Γ is difficult to find. Example: there is no obvious way of representing sequences of proteins as vectors in $\mathbb{R}^p$, but there is a lot of work done in *comparing* two such sequences

Kernel-based learning

Introduction

Definition. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a (real) **positive semi-definite** (PSD) kernel function in $\mathcal{X}$ (or simply "kernel") when

1. It is symmetric: $\forall \boldsymbol{x}, \boldsymbol{x'} \in \mathcal{X},\ k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$

2. For every $n \in \mathbb{N}$, and every choice $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathcal{X}$,

   the (Gram) matrix $\mathbf{K} = (k_{ij})$, where $k_{ij} := k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is PSD.

This definition implies that all kernel matrices are symmetric and PSD

Kernel-based learning

Introduction

Definition. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a (real) **positive definite** (PD) kernel function in $\mathcal{X}$ (or simply "PD kernel") when

1. It is symmetric: $\forall \boldsymbol{x}, \boldsymbol{x'} \in \mathcal{X},\ k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$

2. For every $n \in \mathbb{N}$, and every choice $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathcal{X}$,

   the (Gram) matrix $\mathbf{K} = (k_{ij})$, where $k_{ij} := k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is PD.

PD $\Rightarrow$ PSD

PD $\Leftrightarrow$ PSD & non-singular

Kernel-based learning

Introduction

There are many **equivalent characterizations** of the PSD property for real symmetric matrices. Here are some: $\mathbf{A}_{n\times n}$ is PSD if and only if ...

1. all of its eigenvalues are non-negative
2. the determinants of all of its leading principal minors are non-negative
3. there is a PSD matrix $\mathbf{B}$ such that $\mathbf{B}\mathbf{B}^\top = \mathbf{A}$ (this matrix is unique, denoted with $\mathbf{B} = \mathbf{A}^{1/2}$, and called the *principal square root* of $\mathbf{A}$)
4. $\forall \boldsymbol{c} \in \mathbb{R}^n,\ \boldsymbol{c}^\top \mathbf{A} \boldsymbol{c} \geq 0$

Kernel-based learning

kernels as inner products

Example. Suppose $\mathcal{X} = \mathbb{R}^p$. A natural similarity measure is their inner product (known as dot product):

$$k(\boldsymbol{x}, \boldsymbol{x'}) := \boldsymbol{x}^\top \boldsymbol{x'} = \sum_{j=1}^{p} x_j x'_j$$

Proposition. This function is a valid kernel ¶

Notes:

1. it constitutes a *linear* operation between data points

2. it is valid only for data objects in $\mathbb{R}^p$

Kernel-based learning

This (basic) result raises several deeper questions

1. Are there other (perhaps more general) linear vectorial operations (in $\mathbb{R}^p$) which are valid kernels?

2. Can we create more complex (non-linear) kernels in $\mathbb{R}^p$?

3. Can we create kernels in general spaces $\mathcal{X} \neq \mathbb{R}^p$?

4. Are all kernel functions inner products?

5. Are all inner products valid kernel functions?

6. Is this framework useful, flexible, etc [**desiderata**]... for machine learning? [**How?**]

Kernel-based learning

Introduction

Let $\phi : \mathcal{X} \to \mathbb{R}^p$. We define the function $k(\boldsymbol{x}, \boldsymbol{x}') := \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$. We want to prove that, whatever the ϕ function is, k is a kernel function. ¶

$\boxed{\text{Note that nothing is assumed about the (input) space } \mathcal{X}\text{, which could be any set; in particular, } \mathcal{X} \text{ does not need to be a vector space.}}$

The classic theorem by Aronszajn generalizes this result and says that all kernels are of this form:

Theorem (Aronszajn, 1950). Let k be a kernel in some space $\mathcal{X}$. Then there exists a Hilbert space of functions $\mathcal{H}$ (uniquely generated by k) and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \right\rangle_{\mathcal{H}}, \forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. ¶

Kernel-based learning

What is a Hilbert space?

A Hilbert space (HS) is a vector space endowed with an inner product that is complete wrt the induced norm.

Examples:

1. $\mathbb{R}^p$ with the standard inner product is a finite-dimensional HS
2. l_2 (the space of square-summable real sequences) is an infinite-dimensional HS ¶

Completeness means that all Cauchy sequences converge to an element within the space (using the norm induced by the inner product)

Kernel-based learning

kernels as similarity measures

Within ML, kernels are conceptually regarded as some form of similarity measure: given $\boldsymbol{x}, \boldsymbol{x'} \in \mathcal{X}$, $k(\boldsymbol{x}, \boldsymbol{x'})$ grows as $\boldsymbol{x}, \boldsymbol{x'}$ are more similar (although $k(\boldsymbol{x}, \boldsymbol{x'}) \geq 0$ does not necessarily hold).

Examples:

1. two biological sequences are similar when there exists a good alignment between them

2. two graphs are similar when they share many common paths

Kernel-based learning

Summary

- Kernels are symmetric and PSD functions
- Kernels are inner products in some Hilbert space
- Kernels are (interpreted as) similarity measures, and can be often derived from (metric) distances

The **goal** of this (half) course: extend well-understood, linear statistical learning techniques to real-world, complicated, structured, high-dimensional data based on a rigorous mathematical framework leading to practical modelling tools and algorithms [**desiderata**].

(borrowed from J.P. Vert)

Organization of the course

Theory

Problems

Lab practice

Practical work

Partial exam

Collaborate