Structured Data

Learning, prediction, dependency, testing
Session 5 (Lecture) - Part I: structured output prediction

Florence d'Alché-Buc

Contact: `florence.dalche@telecom-paristech.fr`,
M2 Data Science, Télécom ParisTech & Ecole Polytechnique, Université of Paris-Saclay, France

Table of contents

Reminder

Previously, we saw:
End of scoring methods : - with the work about joint kernel maps (Blaschko and Lampert)
- with the use of MLP and deep learning instead of linear models (Chen et al.)
Regression tree methods extended to kernelized outputs
We showed an application to link prediction

Outline

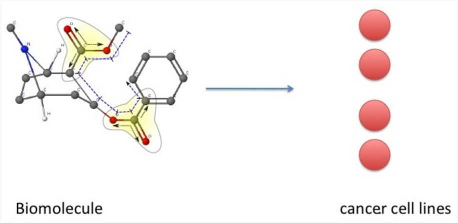
Multi-task regression 1

A very important task in real world machine learning applications:

- **Marketing applications of preference modeling**:where the same choice panel questions (the same x's) are given to many individual consumers, each individual provides his/her own preferences (the y's)
- **Multiple Drug activity prediction**: each drug is described by the same features , each coordinate of the target vector is the activity score of a drug on a specific diseases, some diseases are close, and we want to take into account the relationship between diseases

Multi-task regression

- *INPUT:* an object
- *OUTPUT:* multiple interdependent outputs



Multi-task regression 2

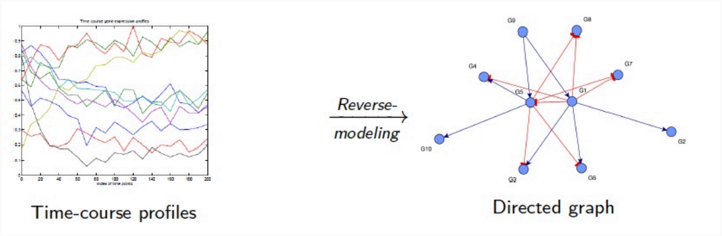
- **Structured multiple class classification**: a multiple class classification problem with relationship between classes like a hierarchy etc...
- **Multi-centered multiple class classification in medecine**: a multiple class classification problem with relationship between classes like a hierarchy AND different datasets coming from different hospitals

Reverse-engineering of a biological dynamical system

Network Inference

Infer the (causal) relationships between state variables from the observation of a noisy multivariate time series

Example: from time course of gene expression, extract a gene regulatory network



Outline

Multi-task classification and regression

Multi-task prediction: formulation 1

Same attributes/ same input space, **same target variable** but observed data are not the same.

- T prediction tasks to solve jointly from T datasets
- fixed unknown probability distribution $\mathbb{P}_t(X, Y)$
- $(\mathbf{x}_{ti}, y_{ti})$ i.i.d. from $\mathbb{P}_t$.
- Let $\mathcal{S}_{(t)} = \{(\mathbf{x}_{ti}, y_{ti}), i = 1, \ldots, n; t = 1, \ldots, T\}$
- How to learn simultaneously T functions $h_t, t = 1, \ldots T$ to respectively predict y_t ?

In the loss function, we incorporate a penalty that encourages the functions h_t to be close to some .

Multi-task classification: formulation 1

Assume

$$\mathbf{w}_t = \mathbf{v}_t + \mathbf{w}_0 \tag{1}$$

and each $\mathbf{v}_t$ is close to $\mathbf{w}_O$

Multitask SVM (linear case)

$\min_{\mathbf{w}_0, \mathbf{v}_t, \xi}$
$\sum_{t=1}^{T} \sum_{i=1}^{n} \xi_{it} + \frac{\lambda_1}{T} \sum_{t=1}^{T} ||\mathbf{v}_t||^2 + \lambda_2 ||\mathbf{w}_0||^2$
s.c. $\forall i = 1, \dots, n, \forall t = 1, \dots, T$
$y_{it}(\mathbf{w}_0 + \mathbf{v}_t)^T \mathbf{x}_{it} \geq 1 - \xi_{it}$
$\xi_{it} \geq 0$

Multi-task classification: towards a new formulation

Optimal solution

$$\mathsf{w}_0^* = \frac{\lambda_1}{\lambda_1 + \lambda_2} \sum_t \mathsf{w}_t^*$$

Proof:

$\mathcal{L}(\mathsf{w}_0, \mathsf{v}_t, \xi_t, \alpha_{it}, \gamma_{it}) =$
$J(\mathsf{w}_0, \mathsf{v}_t, \xi_t) - \sum_{i,t} \alpha_{it} \left(y_{it}(\mathsf{w}_0 + \mathsf{v}_t)^T x_{it} - 1 + \xi_{it}\right) - \sum_{it} \gamma_{it}\xi_{it}$

Setting the derivative of $\mathcal{L}$ to zero

$$\begin{aligned} \mathbf{w}_0^* &= \frac{1}{2\lambda_2} \sum_{it} \alpha_{it} y_{it} \mathbf{x}_{it} \\ \mathbf{v}_t^* &= \frac{T}{2\lambda_1} \sum_{i} \alpha_{it} y_{it} \mathbf{x}_{it} \end{aligned}$$

Multi-task classification: formulation 1'

Multitask SVM (linear), new formulation

$\min_{\mathbf{w}_t,\xi}$
$\left\{\sum_{t=1}^T \sum_{i=1}^n \xi_{it} + \rho_1 \sum_{t=1}^T ||\mathbf{w}_t||^2 + \rho_2 \sum_{t=1}^T ||\mathbf{w}_t - \frac{1}{T}\sum_{s=1}^T \mathbf{w}_s||^2\right\}$
s.c. $\forall i = 1, \dots, n, \forall t = 1, \dots, T$
$y_{it}\mathbf{w}_t^T\mathbf{x}_{it} \geq 1 - \xi_{it}$
$\xi_{it} \geq 0$
where we have :

$$\rho_1 = \frac{1}{T}\frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2}$$
$$\rho_2 = \frac{1}{T}\frac{\lambda_2^2}{\lambda_1+\lambda_2}$$

Therefore, $\mathbf{w}_0^* = \frac{\lambda_1}{T\lambda_2}\sum_t \mathbf{w}_t^*$. This suggests a new formulation...

Ref: Evgeniou and Pontil, 2005.

Multi-task classification: formulation 1'

Multitask SVM (linear), new formulation

$\min_{\mathbf{w}_t,\xi}$
$\left\{\sum_{t=1}^{T}\sum_{i=1}^{n}\xi_{it} + \rho_1\sum_{t=1}^{T}||\mathbf{w}_t||^2 + \rho_2\sum_{t=1}^{T}||\mathbf{w}_t - \frac{1}{T}\sum_{s=1}^{T}\mathbf{w}_s||^2\right\}$
s.c. $\forall i = 1,\dots,n, \forall t = 1,\dots,T$
$y_{it}\mathbf{w}_t^T\mathbf{x}_{it} \geq 1 - \xi_{it}$
$\xi_{it} \geq 0$
where we have :

$$\rho_1 = \frac{1}{T}\frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2}$$
$$\rho_2 = \frac{1}{T}\frac{\lambda_2^2}{\lambda_1+\lambda_2}$$

Solving the new problem

- Apply classic theory of duality and find T vectors $\mathbf{w}_t$
- Or, use the following trick:
    - Define a function $H(x, t) = h_t(x)$
    - $H(t, x) = \mathbf{w}^T \phi(x, t)$
    - $\phi((x, t)) = (\frac{x}{\sqrt{\mu}}, \mathbf{0}, \dots, \mathbf{0}, x, \mathbf{0} \dots, \mathbf{0})$
    - each $\mathbf{0}$ is of dimension d: repeated $t - 1$ times, first and then, repeated, $T - t$ times.
    - μ reflects how much all the tasks are similar
    - $\mathbf{w} = (\sqrt{\mu}\mathbf{w}_0, \mathbf{v}_1, \dots, \mathbf{v}_T)$

A single multi-task classifier

$$\Phi((\mathbf{x},t)) = (\frac{\mathbf{x}}{\sqrt{\mu}}, \underbrace{\mathbf{0},\ldots,\mathbf{0}}_{t-1}, \mathbf{x}, \underbrace{\mathbf{0},\ldots,\mathbf{0}}_{T-t})$$

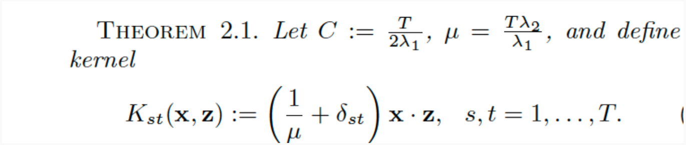
We have then:

$$\mathbf{w}^T \phi((x,t)) = (\mu \mathbf{w}_0 + \mathbf{v}_t)^T x$$
$$\|\mathbf{w}\|^2 = \mu \|\mathbf{w}_0\|^2 + \sum_t \|\mathbf{v}^t\|^2$$

Now the problem boils down to learning a classic SVM with its classic dual formulation

A single multi-task classifier and its dual formulation

A convenient notation:


THEOREM 2.1. *Let* $C := \frac{T}{2\lambda_1}$, $\mu = \frac{T\lambda_2}{\lambda_1}$, *and define* *kernel*

$$K_{st}(\mathbf{x}, \mathbf{z}) := \left(\frac{1}{\mu} + \delta_{st}\right) \mathbf{x} \cdot \mathbf{z}, \quad s, t = 1, \ldots, T.$$

Dual formulation

$\max_\alpha -\frac{1}{2} \sum_{ij,s,t} \alpha_{is} y_{is} \alpha_{jt} y_{jt} K_{st}(\mathrm{x}_i, \mathrm{x}_j) + \sum_{it} \alpha_{it}$
s.c. $\forall i, t,\ 0 \leq \alpha_{it} \leq C$.

Nonlinear multi-task classification

$\mathcal{S} = \{(x_i, t_i; y_i), i = 1, \ldots, n\}$

Dual formulation

$\max_\alpha -\frac{1}{2} \sum_{ij} \beta_i y_i \beta_j y_j G((\mathrm{x}_i, t_i), (\mathrm{x}_j, t_j))) + \sum_i \alpha_i$
s.c. $\forall i,\ 0 \leq \alpha_i \leq C.$

Link with pb in page 17

Equivalent if: $n = mT$ and

$$\begin{aligned} \mathrm{x}_i &= x_{(i \mod T)(i \mod m)} \\ y_j &= y_{(i \mod T)(i \mod m)} \end{aligned}$$

and $G(\mathrm{x}_i, t_i), (\mathrm{x}_j, t_j)) = K_{t_i t_j}(\mathrm{x}_i, \mathrm{x}_j)$

Multi-task prediction with RKHS: formulation 1'

Let us define k, a PDS kernel and $\mathcal{H}_k$, the RKHS associated to that kernel k. $||\cdot||_k$ is the associated nom. NB: here all the functions work in the same input space

Multi-task regression in RKHS

Minimize for $h_1, \ldots, h_T \in \mathcal{H}_k$
$\sum_{t=1}^T \sum_{i=1}^n \ell(\mathrm{x}_{it}, y_{it}, h_t(\mathrm{x}_{it})) + \rho_1 \sum_{t=1}^T ||h_t||_k^2 + \rho_2 \sum_{t=1}^T ||h_t - \frac{1}{T} \sum_{s=1}^T h_s||_k^2$

Structured Multiple class classification

Again : let us look a the linear case. Now M is a $T \times T$ matrix encoding the relationship between the T tasks.

- Training set: $\mathcal{S} = \{(x_i, y_i = (y_{i1}, \dots, y_{iT}), i = 1, \dots n\}$

$\min_{\mathbf{w}_t, \xi_{it}}$
$\left\{ \sum_{t=1}^{T} \sum_{i=1}^{n} \xi_{it} + \rho_1 \sum_{t=1}^{T} ||\mathbf{w}_t||^2 + \rho_2 \sum_{i,j=1}^{T} m_{uv} ||\mathbf{w}_i - \mathbf{w}_j||^2 \right\}$ s.c.
$\forall i = 1, \dots, n, \forall t = 1, \dots, T, y_{it} \mathbf{w}_{it}^T \mathbf{x}_i \geq 1 - \xi_{it}$ and $\xi_{it} \geq 0$

Structured Multi-task Regression in RKHS

Let us define k a PDS kernel and $\mathcal{H}_k$ the RKHS associated to that kernel k. NB: here all the functions work in the same input space

Multi-task regression in RKHS

Minimize for $h_1, \ldots, h_T \in \mathcal{H}_k^T$
$\sum_{t=1}^{T} \left\{ \sum_{i=1}^{n} \ell(\mathrm{x}_i, y_{it}, h_t(\mathrm{x}_i)) + \lambda_1 ||h_t||_k^2 \right\} + \sum_{i,j=1}^{T} m_{ij} ||h_i - h_j||^2$
where M is a $T \times T$ matrix encoding the relationship between the T tasks.

Outline

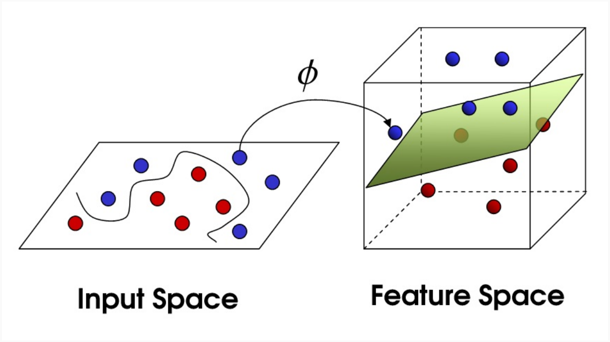
Outline

Multi-task with vector-valued functions

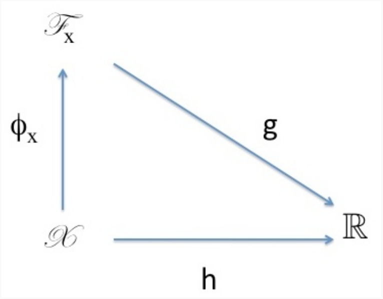
- Similarly to MLP or vector-valued regression trees, we would like to solve the problem 2 with a unique function h with vectorial values.
- The RKHS theory extends to vector-valued functions: it uses operator-valued kernels instead of scalar-valued kernels
- Micchelli and Pontil introduced this theory to the Machine Learning Community

Outline

Reminder: Kernel trick and feature map 1/2



Reminder: Kernel trick and feature map 2/2



$$\begin{tikzcd}
\mathscr{F}_x \arrow[rd, "g"] & \\
\mathscr{X} \arrow[u, "\phi_x"] \arrow[r, "h"'] & \mathbb{R}
\end{tikzcd}$$

- $f(x) = \sum_{i=1}^{n} \alpha_i y_i < \phi(x), \phi(x_i) >_{\mathcal{F}} = \sum_{i=1}^{n} \alpha_i y_i k(x, x_i)$,
- $g(z) = (\sum_i \alpha_i \phi(x_i))^T z$
- $f(x) = g \circ \phi(x)$
- SVM : $h(x) = \text{sign}(f(x) + b)$

Definition

Definition (Reproducing Kernel Hilbert space - RKHS)

Let $\mathcal{H}$ be a Hilbert space of $\mathbb{R}$-valued functions on non-empty set $\mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **reproducing kernel** of $\mathcal{H}$, and $\mathcal{H}$ is a reproducing kernel Hilbert space if:

- $\forall x \in \mathcal{X}, k(\cdot, x) \in \mathcal{X},$
- $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \langle f(\cdot), k(\cdot, x)\rangle_{\mathcal{H}} = f(x)$ (**reproducing property**).

In particular, for any x,y $\in \mathcal{X}$,

$$k(x, y) = \langle k(\cdot, x), k(\cdot, y)\rangle_{\mathcal{H}}$$

Building a RKHS from a PDS kernel k

Theorem (Reproducing Kernel Hilbert space induced by a kernel (Aronszajn, 1950)

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite symmetric kernel. Then, there exists a Hilbert space $\mathcal{H}$ and a function $\phi : \mathcal{X} \to \mathcal{H}$ such that:

$$\forall (x, x') \in \mathcal{X} \times \mathcal{X}, k(x, x') =< \phi(x), \phi(x') >_{\mathcal{H}}$$

Furthermore, there exists a unique $\mathcal{H}$ that has the reproducing property:

$$\forall f \in \mathcal{H}, \forall x \in \mathcal{X}, f(x) =< f(\cdot), k(\cdot, x) >$$

Constructive Proof 1/4

Let us define $\mathcal{H}_0 = \text{span}\{\sum_{i\in I} \alpha_i k(\cdot, x_i), x_i \in \mathcal{X}, |I| < \infty\}$.
$\mathcal{H}_0$ is the set of finite linear combinations of functions $x \to k(\cdot, x_i)$.
Introduce the operation $<, \cdot, \cdot >_{\mathcal{H}_0}$:

$$\begin{aligned} \forall f, g, \in \mathcal{H}_0^2, f(\cdot) &= \sum_{i\in I} \alpha_i k(\cdot, x_i) \\ g(\cdot) &= \sum_{j\in J} \beta_j k(\cdot, z_j) \end{aligned}$$

by

$$< f, g >_{\mathcal{H}_0} = \sum_{i\in I, j\in J} \alpha_i \beta_j k(x_i, z_j)$$

We notice that:

$$< f, g >= \sum_{j\in J} \beta_j f(z_j) = \sum_{i\in I} \alpha_i g(x_i)$$

meaning that this product between f and g does not depend on the expansions of f or g. This last equation also shows that this product is bilinear. It is also trivially symmetric. $< \cdot, \cdot >_{\mathcal{H}_0}$ is a dot product on functions of $\mathcal{H}_0$

Constructive Proof 2/4

We define a norm from this dot product:

$$||f||^2 = < f, f > = \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) = \alpha^T K \alpha$$

where K is the Gram matrix associated to k.
Remark: we have a Cauchy-Schwartz inequality for PDS kernels (that we will use).
Proposition: Cauchy-Schwartz inequality
Let k be a PDS kernel then $\forall (x, z) \in \mathcal{X}^2$, we have:

$$k(x, z)^2 \leq k(x, x) k(z, z)$$

Constructive Proof 3/4

We need to prove that we have the reproducing property:

$$\begin{aligned} < f, k(\cdot, x) >_{\mathcal{H}_0} &= < \sum_i \alpha_i k(\cdot, x_i), k(\cdot, x_i) > \\ &= \sum_i \alpha_i k(x, x_i) \\ &= f(x) \end{aligned}$$

Now $\mathcal{H}_0$ is named a pre-Hilbert space and we need to complete it with the limits of Cauchy sequences to get a Hilbert space.
Let $(f_n)_n$, a Cauchy sequence of functions of $\mathcal{H}_0$.

$$\forall \epsilon > 0, \exists N \in \mathbb{N}, \forall p, q > N, ||f_p - f_q||^2 < \epsilon$$

Let us consider $\mathcal{H} = \mathcal{H}_0 \cup \{\text{lim of Cauchy sequences from} \mathcal{H}_0\}$.
Let us call $f = \lim_{n \to \infty} f_n$.
To ensure the reproducing property for these new functions, we need to have the pointwise convergence of $(f_n(x))_n$ for $x \in \mathcal{X}$.

Constructive Proof 4/4

Proof of pointwise convergence of $(f_n(x))_n$ for $x \in \mathcal{X}$
$\forall x \in \mathcal{X}, \forall (p,q) \in \mathbb{N}^2$,

$$\begin{aligned} |fp(x) - f_q(x)| &= | < f_p, k(\cdot,x) - < f_q, k(\cdot,x) > | \\ &= | < f_p - f_q, k(\cdot,x) > | \\ &\leq \sqrt{< f_p - f_q, f_p - f_q >}\sqrt{k(x,x)} \\ &\leq \|fp - f_q\| \sqrt{k(x,x)} \end{aligned}$$

Then it comes that $(f_n(x))_n$ is a Cauchy Sequence in $\mathbb{R}$ and thus has a limit.
now $f(x) = \lim_{n\to\infty} f_n(x)$.
We want to compute $< \lim f_n, k(\cdot,x) >$. Let us first compute:
$\lim_{n\to\infty} < f_n, k(\cdot,x) >= \lim f_n(x) = f(x)$.
We now define the dot product between a limit of Cauchy Sequence and the function $k(\cdot,x)$ from $\mathcal{H}_0$ as: $< lim f_n, k(\cdot,x) >:= \lim f_n(x) = f(x)$. The dot product can be also defined between two limits of Cauchy sequences and also benefit from the reproducing property.

Unicity theorem

Theorem
Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite symmetric kernel and $\mathcal{H}_k$ be a Hilbert space built from k and $\mathcal{X}$, then $\mathcal{H}_k$ is unique.

Feature Space and feature map

Any Hilbert space $\mathcal{H}$ such that there exists $\phi : \mathcal{X} \rightarrow \mathcal{H}$ with:
$\forall (x, x') \in \mathcal{X} \times \mathcal{X}, k(x, x') =< \phi(x), \phi(x') >_{\mathcal{H}}$
is called a feature space associated with k and ϕ is called a feature map.

Representer theorem

Theorem

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite symmetric kernel and $\mathcal{H}_k$, its corresponding RKHS, then, for any non-decreasing function $\Omega : \mathbb{R} \to \mathbb{R}$ and any loss function L: $\mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, any minimizer of :

$$J(f) = L(f(x_1), \dots, f(x_n)) + \lambda\Omega(\|f\|^2_{\mathcal{H}}) \tag{2}$$

admits an expansion of the form:

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot).$$

Moreover if Ω is strictly increasing, then any minimizer of 2 has exactly this form.

Proof of the Representer theorem

Let us define: $\mathcal{H}_1$ = span $\{k(x_i, \cdot), i = 1, \dots, n\}$
Any $f \in \mathcal{H}$ writes as: $f = f_1 + f^\perp$, with $f_1 \in \mathcal{H}_1$ and $f^\perp \in \mathcal{H}_1^\perp$
where $\mathcal{H}$= direct sum of$\mathcal{H}_1$ and $\mathcal{H}_1^+$.
By orthogonality, $\|f\|^2 = \|f_1\|^2 + \left\|f_1^\perp\right\|^2$
Hence, by property of Ω, $\Omega(\|f\|^2) = \Omega(\|f_1\|^2) + \Omega(\left\|f_1^\perp\right\|^2) \geq \Omega(\|f_1\|^2)$
By the reproducing property, we get:
$f(x_i) =< f_1(\cdot) + f_1^\perp(\cdot), k(x_i, \cdot) >=< f_1(\cdot), k(x_i, \cdot) >= f_1(x_i)$
Hence, $L(f(x_1), \dots, f(x_n)) = L(f_1(x_1), \dots, f_1(x_n))$ and $J(f_1) \leq J(f)$
To recap, if f is a minimizer of $J(f)$, then f_1 is also a minimizer of J.
Moreover if Ω is stricly increasing, $J(f_1) < J(f)$, then any $f = f_1 + f_1^\perp$
exactly equals to f_1.

A to-do do list

1. Define a PDS kernel: $k(\cdot, \cdot)$
2. Define a RKHS, $\mathcal{H}$ from k with an appropriate norm $||\cdot||_{\mathcal{H}}$
3. Define a loss functional with two terms: a local loss function ℓ and a penalty function Ω
4. Prove/use a representer theorem to get the form of the minimizer of this functional: $\sum_i \alpha_i k(\cdot, x_i)$
5. Solve the optimization problem with this minimizer

Outline

Outline

Regression with operator-valued kernel

We now consider the following case:

- We search a function h: $\mathcal{X} \to \mathcal{F}_y$
- $\mathcal{F}_y$ is an Hilbert space (NB: can be $\mathbb{R}^p$) : this theory does not need to assume a kernel k_y (it is more general)
- We will define functions of the following form:

$$h(\mathsf{x}) = \sum_i K(\mathsf{x}, \mathsf{x}_i)\mathsf{c}_i$$

Operator-valued kernels

Development of new learning tasks:

- Multi-task learning [Micchelli & Pontil, 2005, Evgeniou *et al.*, 2005, Caponnetto et al. 2008]
- Functional regression [Kadri *et al.*, 2010]
- Structured output prediction, link prediction [Brouard *et al.*, Dinuzzo et al. 2011]
- Semi-supervised learning [Brouard *et al.* 2011, Quang 2011,2014]

Definition of an operator-valued kernel

Let $\mathcal{X}$ be a non-empty set and $\mathcal{F}_y$, a Hilbert space $\mathcal{F}_y$; $\mathcal{L}(\mathcal{F}_y)$ is the set of all bounded linear operators from $\mathcal{F}_y$ to itself.

Operator-valued kernel :

(Senkene & Tempel'man, 1973 ; Caponnetto et al., 2008)

$K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{F}_y)$ is an operator-valued kernel if:

- $\forall (\mathrm{x}, \mathrm{x}') \in \mathcal{X} \times \mathcal{X}$, $K(\mathrm{x}, \mathrm{x}') = K(\mathrm{x}', \mathrm{x})^*$
- $\forall m \in \mathbb{N}$, such that $\forall i \in \{1, \dots, m\}, (\mathrm{x}_i, \mathrm{y}_i) \in \mathcal{X} \times \mathcal{F}_y$,

$$\sum_{i,j=1}^{m} \langle K(\mathrm{x}_i, \mathrm{x}_j)\mathrm{y}_j, \mathrm{y}_i \rangle_{\mathcal{F}_y} \geq 0 \ .$$

Examples of operator-valued kernel ($\mathcal{F}_y = \mathbb{R}^p$)

Refs: Caponnetto et al. 2008; Carmeli et al. 2012; Lim et al. 2013

Trivial kernel (diagonal kernels):

$$K(\mathrm{x}, \mathrm{z}) = k(\mathrm{x}, \mathrm{z}) I$$

with I: identity matrix, and k a scalar kernel).

Let $k_1, \ldots, k_p$ be p scalar kernels

$$K(x, x') = \begin{pmatrix} k_1(x, x') & 0 & \ldots & 0 \\ 0 & k_2(x, x') & \ldots & 0 \\ 0 & 0 & \ldots & k_p(x, x') \end{pmatrix}$$

Examples of operator-valued kernel

Other examples

- Decomposable kernel :
  - $\forall(\mathrm{x}, \mathrm{z}) \in \mathbb{R}^p \times \mathbb{R}^p, K(\mathrm{x}, \mathrm{z}) = k(\mathrm{x}, \mathrm{z})B$ and $B \in S_p^+$
  - This kernel is used for multi-task regression when the matrix coefficient B_{ij} codes for the dependency relationship between task i and task j
- Transformable kernel
  - $\forall(i, j) \in \{1, \ldots, d\}^2, K(\mathrm{x}, \mathrm{z})_{ij} = k(T_i(\mathrm{x}), T_j(\mathrm{z}))$
  - where T_i is a transformation from $\mathcal{X}$ to some set $\mathcal{Z}$, and k:$\mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is a scalar kernel
- Hadamard product of two kernels :
  - $K(x, z) = K_1(x, z) \; o \; K_2(x, z)$

Building a RKHS from an operator-valued kernel

Theorem (Senkene & Tempel'man, 1973 ; Micchelli & Pontil, 2005)

Let $\mathcal{X}$ be a set and $\mathcal{Y}$ be an Hilbert space.
If $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{F}_y)$ is an operator-valued kernel, then there exists a unique RKHS $\mathcal{H}_K$ which admits K as the reproducing kernel, that is

$$\forall x \in \mathcal{X}, \forall \mathbf{y} \in \mathcal{F}_y,\ \langle K(\cdot, x)\mathbf{y}, h\rangle_{\mathcal{H}} = \langle \mathbf{y}, h(x)\rangle_{\mathcal{F}_y}\,. \tag{3}$$

- $\mathcal{H}_K$ is built from functions of the form: $f(\cdot) = \sum_i K(\cdot, x_i)\mathbf{a}_i$, defining a inner product of functions f and $g(\cdot) = \sum_j K(\cdot, z_j)\mathbf{b}_\mathbf{j}$ as $< f, g >_{\mathcal{H}} = \sum_{i,j} \langle \mathbf{a}_i, K(x_i, z_j)\mathbf{b}_j\rangle_{\mathcal{F}_y}$ and completing this space by limits of Cauchy sequences.
- For sake of simplicity we omit K and use $\mathcal{H} = \mathcal{H}_K$

Representer Theorem (least square, supervised case)

Theorem (Micchelli & Pontil, 2005)
Let $\mathcal{X}$ be a set and $\mathcal{Y}$ be an Hilbert space. Given the RKHS $\mathcal{H}$ with reproducing kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{F}_y)$, a set of labeled examples $S_n = \{(x_i, \mathbf{y}_i)\}_{i=1}^{n} \subseteq \mathcal{X} \times \mathcal{F}_y$, a positive $\lambda \in \mathbb{R}^+$, let $J(h) = \sum_{i=1}^{n} \|\mathbf{y}_i - h(x_i)\|_{\mathcal{Y}}^2 + \lambda \|h\|_{\mathcal{H}}^2$ be the functional loss, then the minimizer $\hat{h}$ of $J(h)$

$$\arg\min_{h \in \mathcal{H}} J(h) = \sum_{i=1}^{n} \|\mathbf{y}_i - h(x_i)\|_{\mathcal{F}_y}^2 + \lambda \|h\|_{\mathcal{H}}^2 \ , \tag{4}$$

is unique and admits an expansion: $\quad \hat{h}(\cdot) = \sum_{i=1}^{n} K(\cdot, x_i)\mathbf{c}_i \ ,$

Learning Ridge OVKR

As the loss function $J(h)$ is strictly convex, there exists a unique minimizer.

Closed-form solution

The minimizer of $J(h) = \sum_{i=1}^{n} \|\mathbf{y}_i - h(x_i)\|^2_{\mathcal{F}_y} + \lambda \|h\|^2_{\mathcal{H}}$, with h : $h(\cdot) = \sum_{i=1}^{n} K(\cdot, x_i)\mathbf{c}_i$, has the following form:

$$\hat{h}(\cdot) = \phi_x(\cdot)(K_x + \lambda I_n)^{-1} Y_n$$

K_x is the $n \times n$ block matrix , with each block of the form $K(x_i, x_j)$. Y_n is the vector of all stacked vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$, and ϕ_x is the matrix composed of $[K(\cdot, x_1) \ldots K(\cdot, x_n)]$.

Outline

Multi-task regression: first formulation using Operator-Valued Kernels

Mixed Effect Regularizer

We set:

$$A_\omega = \frac{1}{2(1-\omega)(1-\omega+\omega D)}$$
$$C_\omega = (2 - 2\omega + \omega D)$$

Regularization:

$$\Omega(h) = A_\omega \left(C_\omega \sum_{t=1}^{T} \|h_t\|_k^2 + \omega D \sum_t \|h_t - \bar{h}\|^2 \right)$$

Choose a kernel $K_\omega(x, x) = k(x, x)(\omega \mathbb{1} + (1-\omega) I_T)$

Multi-task regression: second formulation with OVK

Let $k_1, \ldots, k_p$ be p scalar-valued kernels,
$\mathcal{H}_1, \ldots, \mathcal{H}_p$ be the p RKHS defined from these kernels.
Let M be a $p \times p$ positive weight matrix encoding the similarity between regression tasks.
Let us define the following regularizer:

$$\Omega(f) = \frac{1}{2} \sum_{ij}^{p} ||f_i - f_j||_k^2 m_{ij} + \sum_{i=1}^{p} ||f_i||_k^2 m_{ii}$$

This regularizer enforces the similarity between component functions that are close according matrix M.

Multi-task regression: a second formulation

$$\Omega(f) = \frac{1}{2}\sum_{ij}^{p} ||f_i - f_j||_k^2 m_{ij} + \sum_{i=1}^{p} ||f_i||_k^2 m_{ii}$$

This regularizer enforces the similarity between component functions that are close according matrix M.

$$\begin{aligned}\Omega(f) &= \sum_{i,j} ||f_i||_k^2 m_{ij} - \langle f_i, f_j\rangle_k m_{ij}) + \sum_{i=1}^{p} ||f_i||_k^2 m_{ii} \\ &= \sum_{i=1}^{p} ||f_i||_k^2 \sum_j (1+\delta_{ij}) m_{ij} - \sum_{ij} \langle f_i, f_j\rangle_k m_{ij} \\ &= \sum_{ij} \langle f_i, f_j\rangle_k L_{ij}\end{aligned}$$

with $L = D - M$, $D = (d_{ij})_{ij}$, with the convention: $d_{ij} = \delta_{ij}(\sum_k m_{ik} + m_{ij})$

Useful result

Let us define a decomposable kernel:

$$K(x, x') = Bk(x, x'),$$

with B a sdp matrix and k, a scalar-valued kernel.
Then the ℓ_2 norm in the RKHS $\mathcal{H}_K$ of $f(\cdot) = \sum_\ell K(\cdot, \mathbf{x}_\ell)\mathbf{c}_\ell$ can be written as:

$$||f||_K^2 = \sum_{i,j=1}^{p} B_{ij}^{+} \langle f_i, f_j \rangle_k,$$

where B^+ is the pseudo-inverse of B and $f = (f_1, \dots, f_p)$.
Then the resulting kernel for Multi-task regression is the decomposable kernel: $K(x, x') = L^+ k(x, x')$, with L^+, the pseudo inverse of L.

Take home message

If you want to solve a multi-task regression problem, you define the proper decomposable kernel to get a graph regularizer term $\Omega(f) = \frac{1}{2}\sum_{ij}^{p} ||f_i - f_j||_k^2 m_{ij} + \sum_{i=1}^{p} ||f_i||_k^2 m_{ii}$

Working in RKHS of vector-valued functions

As in the scalar-valued case, to define a regularizer of your choice, it is very often the case that you just need to define the proper kernel !

You have now a new tool to take into account structure in the output space: define the proper operator-valued kernel and thus the proper norm $\|f\|_K^2$. NB: $\|\cdot\|_K := \|\cdot\|_{\mathcal{H}_K}$.

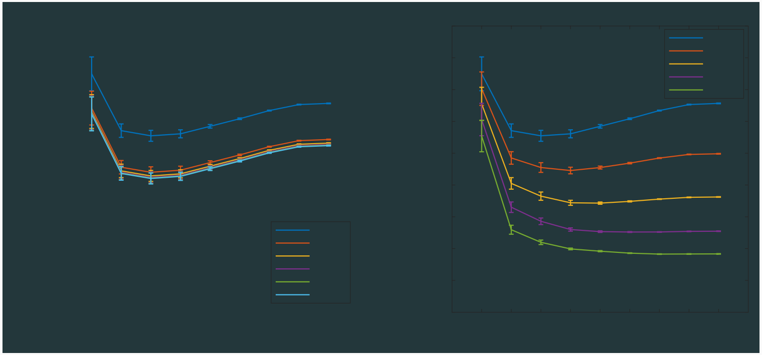
Results on NCI-dataset: Drug activity prediction

Su et al. 2010 Dataset: contains the 2303 molecules that are all active against at least one cell line. Each molecule is represented by a graph, where nodes correspond to atoms and edges to bonds between atoms. The Tanimoto kernel (Ralaivola et al. 2005) is used for the scalar input kernel:

$$K(x, x') = \frac{k_m(x, x')}{k_m(x, x) + k_m(x', x') - k_m(x, x')}.$$

k_m is chosen as a linear path kernel. Input feature vectors $\varphi_{x_m}(x)$ are binary vectors that indicate the presences and absences in the molecules of all existing paths containing a maximum of m bonds. $m = 6$.

Results on NCI-dataset: Drug activity prediction



Outline

Other applications

- Vector Field learning
- Mismatch Removal
- Image Colorization
- Structured Classification

Mismatch Vector Field Learning

Establishing correspondences between two images of the same scene

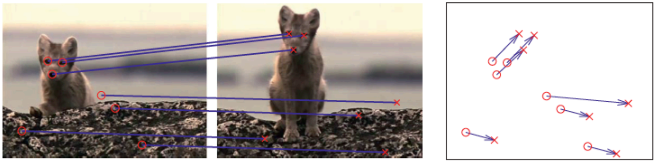


Fig. 1. Schematic illustration of motion field introduced by image pairs. Left: an image pair and its putative matches; right: motion field samples introduced b matches in the left figure. ° and × indicate feature points in the first and second images, respectively.

Ref: Ma et al. Pattern Recognition, 2013.

Vector Autoregressive models

Model and observation assumptions

The temporal evolution of the system is ruled by a **first-order autoregressive** model $h : \mathbb{R}^p \to \mathbb{R}^p$:

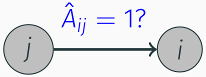
$$\mathrm{x}_{t+1} = h(\mathrm{x}_t) + \boldsymbol{\epsilon}_{t+1} \tag{5}$$

where

- $\mathrm{x}_1, \ldots, \mathrm{x}_{N+1} \in \mathbb{R}^p$: observed time series of a dynamical system comprising of p variables at time $t = 1, \ldots, N+1$
- $\boldsymbol{\epsilon}_t$: a noise term (chosen Gaussian) with zero-mean

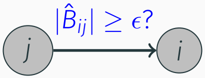
Network inference by thresholding the Jacobian

	Linear	Nonlinear
1. Model	$x_{t+1} = Bx_t + \epsilon_{t+1}$	
2. Learn	B	
4. Adjacency matrix		



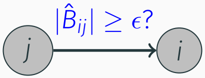
Network inference by thresholding the Jacobian

	Linear	Nonlinear
1. Model	$x_{t+1} = Bx_t + \epsilon_{t+1}$	
2. Learn	B	
4. Adjacency matrix	Threshold $\hat{B}$	



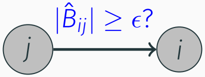
Network inference by thresholding the Jacobian

	Linear	Nonlinear
1. Model	$x_{t+1} = Bx_t + \epsilon_{t+1}$	$h : \mathbb{R}^p \to \mathbb{R}^p$ $x_{t+1} = h(x_t) + \epsilon_{t+1}$
2. Learn	B	h
4. Adjacency matrix	Threshold $\hat{B}$	



Network inference by thresholding the Jacobian

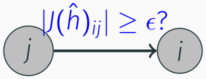
	Linear	Nonlinear
1. Model	$x_{t+1} = Bx_t + \epsilon_{t+1}$	$h : \mathbb{R}^p \to \mathbb{R}^p$ $x_{t+1} = h(x_t) + \epsilon_{t+1}$
2. Learn	B	h
	$\hat{B}_{ij} = \frac{\partial (\hat{B}x_t)^i}{\partial x_t^j}$	
4. Adjacency matrix	Threshold $\hat{B}$	



Network inference by thresholding the Jacobian

	Linear	Nonlinear
1. Model	$x_{t+1} = Bx_t + \epsilon_{t+1}$	$h : \mathbb{R}^p \to \mathbb{R}^p$ $x_{t+1} = h(x_t) + \epsilon_{t+1}$
2. Learn	B	h
3. Jacobian $J(\hat{h})_{ij}$	$\hat{B}_{ij} = \frac{\partial (\hat{B}x_t)^i}{\partial x_t^j}$	$T\left(\frac{\partial \hat{h}(x_1)^i}{\partial x_1^j}, \dots, \frac{\partial \hat{h}(x_N)^i}{\partial x_N^j}\right)$
4. Adjacency matrix	Threshold $\hat{B}$	Threshold $J(\hat{h})$

Examples for $T(z_1, \dots, z_N)$:
$\frac{1}{N}\sum_{t=1}^{N} z_t$, median$(z_1, \dots, z_N)$, max$(|z_1|, \dots, |z_N|), \dots$



Learning a sparse OVKR

Case 1: Kernel K is given

Learning h boilds down to the following optimization problem :

$$\underset{C}{\text{minimize}} \;\; \mathcal{L}(C) = \sum_{i=1}^{n} ||\mathbf{y}_i - h_C(\mathbf{x}_i)||_2^2 \tag{6}$$

Learning a sparse OVKR

Case 1: Kernel K is given

Learning h boilds down to the following optimization problem :

$$\underset{C}{\text{minimize}} \ \mathcal{L}(C) = \sum_{i=1}^{n} ||\mathbf{y}_i - h_C(\mathbf{x}_i)||_2^2 + \lambda_h ||h_C||_{\mathcal{H}_K}^2 \tag{6}$$

Learning a sparse OVKR

Case 1: Kernel K is given

Learning h boilds down to the following optimization problem :

$$\underset{C}{\text{minimize}} \;\; \mathcal{L}(C) = \sum_{i=1}^{n} ||\mathbf{y}_i - h_C(\mathbf{x}_i)||_2^2 + \lambda_h ||h_C||_{\mathcal{H}_K}^2 + \Omega(C) \tag{6}$$

- Penalty on $\mathbf{c}_\ell$'s :
  - $\Omega_1(C) = \lambda_C ||C||_{\ell_1} = \lambda_C \sum_{\ell=1}^{n} \sum_{p=1}^{d} |c_\ell^p|$
  - $\Omega_{struct}(C) = \lambda_C ||C||_{\ell_1/\ell_2} = \lambda_C \sum_{\ell=1}^{n} ||\mathbf{c}_\ell||_2$

Learning a sparse OVKR

Case 2: $K(x, x') = Bk(x, x')$ Parameter B of Kernel K has to be learned

Learning h boilds down to the following optimization problem :

$$\underset{B,C}{\text{minimize}} \ \ \mathcal{L}(B, C) = \sum_{i=1}^{n} ||\mathbf{y}_i - h_{B,C}(\mathbf{x}_i)||_2^2 + \lambda_h ||h_{B,C}||^2_{\mathcal{H}_K} + \Omega(C) + \Omega(B) \qquad (6)$$

- Penalty on $\mathbf{c}_\ell$'s :
  - $\Omega_1(C) = \lambda_C ||C||_{\ell_1} = \lambda_C \sum_{\ell=1}^{n} \sum_{p=1}^{d} |c_\ell^p|$
  - $\Omega_{struct}(C) = \lambda_C ||C||_{\ell_1/\ell_2} = \lambda_C \sum_{\ell=1}^{n} ||\mathbf{c}_\ell||_2$
- Penalty on B :
  - $\Omega_1(B) = \lambda_B ||B||_{\ell_1} + 1_{\mathcal{S}_d^+}(B) = \lambda_B \sum_{p,q=1}^{d} |B_{pq}| + 1_{\mathcal{S}_d^+}(B)$

Learning OVKR

- For fixed $\hat{B}$, the loss function to be minimized becomes:

$$\mathcal{L}(\hat{B}, C) = \sum_{i=1}^{n} ||\mathbf{y}_i - h_{\hat{B},C}(\mathbf{x}_i)||_2^2 + \lambda_h ||h_{\hat{B},C}||^2_{\mathcal{H}_K} + \Omega(C) \qquad (7)$$

- For given $\hat{C}$, the loss function to be minimized is the following:

$$\mathcal{L}(B, \hat{C}) = \sum_{i=1}^{n} ||\mathbf{y}_i - h_{B,\hat{C}}(\mathbf{x}_i)||^2 + \lambda_h ||h_{B,\hat{C}}||^2_{\mathcal{H}_K} + \Omega(B) \qquad (8)$$

Learning OVKR

- For fixed $\hat{B}$, the loss function to be minimized becomes:

$$\mathcal{L}(\hat{B}, C) = \sum_{i=1}^{n} ||\mathbf{y}_i - h_{\hat{B},C}(\mathbf{x}_i)||_2^2 + \lambda_h ||h_{\hat{B},C}||^2_{\mathcal{H}_K} + \Omega(C) \qquad (7)$$

- For given $\hat{C}$, the loss function to be minimized is the following:

$$\mathcal{L}(B, \hat{C}) = \sum_{i=1}^{n} ||\mathbf{y}_i - h_{B,\hat{C}}(\mathbf{x}_i)||^2 + \lambda_h ||h_{B,\hat{C}}||^2_{\mathcal{H}_K} + \Omega(B) \qquad (8)$$

We employ proximal gradient algorithms to minimize (7) and (8)
Ref: introduction to proximal algorithms, see Vandenberghe 'slides `http://www.seas.ucla.edu/~vandenbe/236C/lectures/proxgrad.pdf` + Ref : Lim et al. 2015.

What are proximal algorithms about?

- Class of optimization algorithms
- Tools for convex, **nonsmooth**, constrained and **large-scale** problems
- Recent interest in machine learning, image and signal processing communities
- Proximal operators are **not new** [Moreau, 1962]

Definition

Proximal operator

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a closed proper convex function, the proximal operator $\textbf{prox}_{\lambda f} : \mathbb{R}^n \to \mathbb{R}^n$ of the scaled function λf, where $\lambda > 0$ is defined by

$$\textbf{prox}_{\lambda f}(v) = \arg\min_{x} \left\{ f(x) + \frac{1}{2\lambda} \|x - v\|_2^2 \right\} \tag{9}$$

Proximal gradient algorithms

- Type 1 problem

$$\text{minimize } f(x) + g(x) \tag{10}$$

  - $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are closed proper convex functions
  - f is **differentiable** and g possibly **nonsmooth**
  - Forward-Backward Splitting [Bruck 1975],[Lions & Mercier, 1979],[Beck & Teboulle, 2009,2010]

Proximal gradient algorithms

- Type 1 problem

$$\text{minimize } f(x) + g(x) \tag{10}$$

  - $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are closed proper convex functions
  - f is **differentiable** and g possibly **nonsmooth**
  - Forward-Backward Splitting [Bruck 1975],[Lions & Mercier, 1979],[Beck & Teboulle, 2009,2010]
- Type 2 problem

$$\text{minimize } f(x) + \sum_{i=1}^{N} g_i(x) \tag{11}$$

  - $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ are closed proper convex functions for all $i = 1 \dots N$
  - f is **differentiable** and g_i's are possibly **nonsmooth**
  - Generalized Forward-Backward Splitting [Raguet et al., 2011]

Alternate scheme to learn C and B

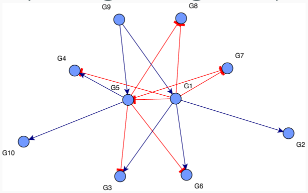
```
Inputs : $B_0 \in \mathcal{S}_p^+$; $M$; $\epsilon_B$; $\epsilon_C$
Initialize : $m = 0$; STOP=false
while $m < M$ and STOP=false do
    Step 1: Given $B_m$, minimize the loss function (7) and obtain $C_m$
    Step 2: Given $C_m$, minimize the loss function (8) and obtain $B_{m+1}$
    if $m > 0$ then
        STOP:=$||B_m - B_{m-1}|| \leq \epsilon_B$ and $||C_m - C_{m-1}|| \leq \epsilon_C$
    end if
    Step 3: $m \leftarrow m + 1$
end while
```

Other procedure to learn C and B

- Dinuzzo et al. 2011, Sylvester equations for C and block-coordinate descent for B but only smooth penalties
- Ciliberto et al. 2015, variable change and recent algorithm (barrier method)

DREAM3 data set

- DREAM stands for Dialogue for Reverse Engineering Assessments and Methods
- Why DREAM3 ? Two data sets including time-series without perturbation data
- An example of gene regulatory network : *E. coli* subnetwork



$\rightarrow$ activation $\rightarrow$ inhibition

Results on 10-size networks (DREAM3 challenge)

Simulated data to have ground truth to build AUROC and AUPR.

- 5 networks whose structure is taken from E. coli or S. Cerevisiae: simulation of Michaelis-Menten equations
- Datasets: 4 time series of 21 time-points (4 initial conditions)
- Comparison with the best team in DREAM3 challenge that only uses time-series and other methods
- Assessment of performance : area under the ROC curve (TP rate vs FP rate), area under the Precision Recall curve (Pr = TP/P, Rec = TP)

Comparison with state-of-the-art methods 1

	AUROC				
	E1	E2	Y1	Y2	Y3
OKVAR Prox* + *True B*	96.2	97.1	95.8	90.6	89.7
OKVAR Prox	**81.5**	**78.7**	**76.5**	**70.3**	**75.1**
LASSO	69.5	57.2	46.6	62.0	54.5
GPODE	60.7	51.6	49.4	61.3	57.1
G1DBN	63.4	77.4	60.9	50.3	62.4
Team 236	62.1	65.0	64.6	43.8	48.8
Team 190	57.3	51.5	63.1	57.7	60.3

LASSO : sparse linear models, GPODE (Aijo et al. 2009): structure inference method based on non-parametric Gaussian process modeling and parameter estimation of nonlinear ordinary differential equations, G1DBN (Lebre et al. 2009) : Dynamic Bayesian Network inference, Teams 236 : best team on DREAM3 using only time-series (Bayesian method), no perturbation data

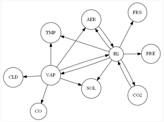
Comparison with state-of-the-art methods 2

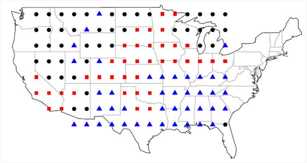
	AUPR				
	E1	E2	Y1	Y2	Y3
OKVAR Prox* + *True B*	43.2	51.6	27.9	40.7	36.4
OKVAR Prox	**32.1**	**50.1**	**35.4**	37.4	**39.7**
LASSO	17.0	16.9	8.5	32.9	23.2
GPODE	18.0	14.6	8.9	**37.7**	34.1
G1DBN	16.5	36.4	11.6	23.2	26.3
Team 236	19.7	37.8	19.4	23.6	23.9
Team 190	15.2	18.1	16.7	37.1	37.3

When no ground truth is provided: climate data

- 125 equally spaced meteorological stations in USA
- Up to 12 climate variables
- Model selection using one station with BIC, learning on the others

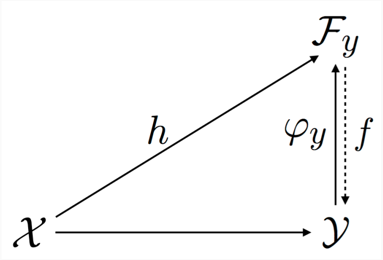
We used OKVAR to infer a consensus network from mutliple runs of OKVAR on climate time series of each of the remaining stations. We clustered the inferred networks and visualized the clustering on the US map.





Outline

Learning with complex outputs with output kernel



$$\begin{tikzcd} & & \mathcal{F}_y \arrow[d, dashed, shift left, "f"] \\ \mathcal{X} \arrow[urr, "h"] \arrow[rr] & & \mathcal{Y} \arrow[u, shift left, "\varphi_y"] \end{tikzcd}$$

1. Define k_y for the output space, ϕ_y is a feature map associated to k and $\mathcal{F}_y$ the feature space corresponding to ϕ_y
2. learn $h : \mathcal{X} \to \mathcal{F}_y$ from training data S
3. Once h is learned, to make a prediction : solve a pre-image problem

Reminder: Link prediction with Output Kernel Regression

To solve the link prediction task:

1. We learn $h_{tree} : \mathcal{X} \to \mathcal{F}_y$
2. For prediction :
    - we compute for a new pair:
      $f(x, x') = \text{sign}(< h_{tree}(x), h_{tree}(x') >_{\mathcal{F}_y} -\theta)$

Limitations of tree-based methods

- Not appropriate when huge input dimension (although Random Forest is a possible key)
- Not appropriate for structured data
- Not appropriate for semi-supervised learning

Idea: learn with functions in vector-valued RKHS with kernelized outptus

Learning with complex outputs with input and output kernel

Motivation: deal with structured outputs as well as structured inputs

1. Define k_y for the output space, ϕ_y is a feature map associated to k and $\mathcal{F}_y$ the feature space corresponding to ϕ_y
2. NB : special case : $\mathcal{F}_y = \mathbb{R}^p$
3. Define $K_x : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{F}_y)$ for the input space (φ_x: feature map associated with K)
4. Learn $h : \mathcal{X} \to \mathcal{F}_y$ from training data S
5. Once h is learned, to make a prediction : solve a pre-image problem

$$\begin{tikzcd}
\mathcal{B}(\mathcal{F}_y,\mathcal{H}) \arrow[r, "g"] & \mathcal{F}_y \arrow[d, shift left, dashed, "f"] \\
\mathcal{X} \arrow[u, "\phi_x"] \arrow[ur, "h"] \arrow[r] & \mathcal{Y} \arrow[u, shift left, "\varphi_y"]
\end{tikzcd}$$

Application to link prediction

Link prediction with output kernel

In this task, we assume that for each object $u \in \mathcal{U}$

- $\phi_y(u) \in \mathcal{F}_y$, using a feature map $\phi_y(\cdot)$ and a feature space $\mathcal{G}_y$ associated to a given kernel $k_y : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$
- k_y codes for the proximity of two nodes in the graph

Training data:

Description of the input data by $K_x = (K_{ij})_{ij}$ the block matrix composed of matrices $K(u_i, u_j)$ and K_y

Application of output kernel regression to link prediction

Input and output kernel regression for link prediction

We define f as follows:

$$f(u, u') = (\langle (h(u), h(u')\rangle - t),$$

where t is a threshold and $h : \mathcal{U} \rightarrow \mathcal{F}_y$ approximates the relationship between u and $\phi_y(u)$.

We define $K : \mathcal{U} \times \mathcal{U} \rightarrow \mathcal{L}(\mathcal{F}_y)$, K has values in the set of linear bounded operators on $\mathcal{F}_y$.

$$h(u) = \sum_{\ell} K(u, u')\mathbf{c}_\ell \in \mathcal{F}_y$$

Revisiting link prediction

Similarly to output kernel tree (OK3), we have: $\forall (u, u'), \langle h(u), h(u') \rangle_{\mathcal{F}_y}$ is an approximation of $k_y(u, u')$
How to choose k_y for link prediction?

- Use the **diffusion kernel** [Kondor & Lafferty, 2002)] :

$$K_y = \exp(-\beta L),$$

- where the graph Laplacian L is defined by :$L = D - A$
- with D the degree matrix and A the adjacency matrix of the training graph

Training data:
Description of the input data by $K_x = (K_{ij})_{ij}$ the block matrix composed of matrices $K(u_i, u_j)$ and K_y

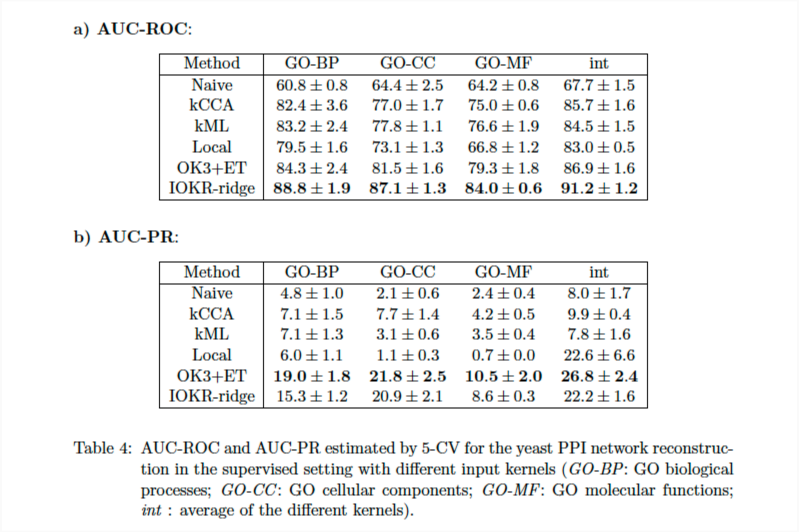
Semi-supervised learning with operator-valued kernel

The link prediction problem can be solved by minimizing:

$$L(f) = \sum_{\ell} ||\phi_y(u_\ell) - f(u_\ell)||_K^2 + \lambda_f ||f||_K^2 + \lambda_s \sum_{\ell,m} w_{\ell m} ||f(u_\ell) - f(u_m)||^2,$$

with W a matrix such that $w_{\ell m}$ encodes proximity between u_ℓ and u_m.
NB: as for kernel ridge regression, there is a closed form solution.(Brouard et al. 2011)

Network inference: AUPR on yeast ppi net (yeast)



a) AUC-ROC:

Method	GO-BP	GO-CC	GO-MF	int
Naive	60.8 ± 0.8	64.4 ± 2.5	64.2 ± 0.8	67.7 ± 1.5
kCCA	82.4 ± 3.6	77.0 ± 1.7	75.0 ± 0.6	85.7 ± 1.6
kML	83.2 ± 2.4	77.8 ± 1.1	76.6 ± 1.9	84.5 ± 1.5
Local	79.5 ± 1.6	73.1 ± 1.3	66.8 ± 1.2	83.0 ± 0.5
OK3+ET	84.3 ± 2.4	81.5 ± 1.6	79.3 ± 1.8	86.9 ± 1.6
IOKR-ridge	$\mathbf{88.8 \pm 1.9}$	$\mathbf{87.1 \pm 1.3}$	$\mathbf{84.0 \pm 0.6}$	$\mathbf{91.2 \pm 1.2}$

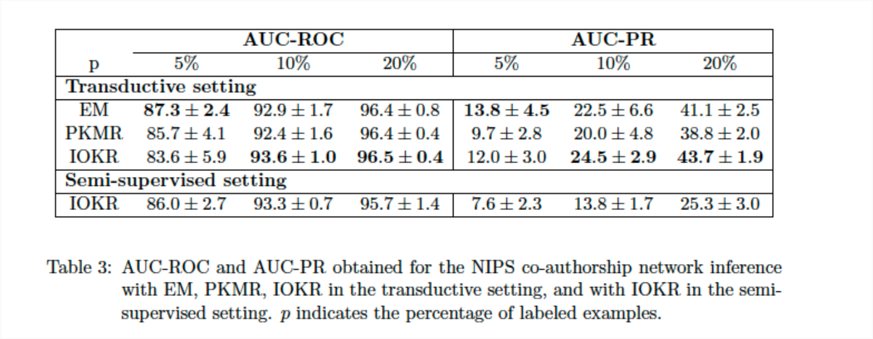
b) AUC-PR:

Method	GO-BP	GO-CC	GO-MF	int
Naive	4.8 ± 1.0	2.1 ± 0.6	2.4 ± 0.4	8.0 ± 1.7
kCCA	7.1 ± 1.5	7.7 ± 1.4	4.2 ± 0.5	9.9 ± 0.4
kML	7.1 ± 1.3	3.1 ± 0.6	3.5 ± 0.4	7.8 ± 1.6
Local	6.0 ± 1.1	1.1 ± 0.3	0.7 ± 0.0	22.6 ± 6.6
OK3+ET	$\mathbf{19.0 \pm 1.8}$	$\mathbf{21.8 \pm 2.5}$	$\mathbf{10.5 \pm 2.0}$	$\mathbf{26.8 \pm 2.4}$
IOKR-ridge	15.3 ± 1.2	20.9 ± 2.1	8.6 ± 0.3	22.2 ± 1.6

Table 4: AUC-ROC and AUC-PR estimated by 5-CV for the yeast PPI network reconstruction in the supervised setting with different input kernels (*GO-BP*: GO biological processes; *GO-CC*: GO cellular components; *GO-MF*: GO molecular functions; *int* : average of the different kernels).

- Results of Céline Brouard: various input kernel, output kernel: diffusion kernel

Network inference:Conference co-authorship (NIPS)



p	AUC-ROC 5%	AUC-ROC 10%	AUC-ROC 20%	AUC-PR 5%	AUC-PR 10%	AUC-PR 20%
Transductive setting						
EM	**87.3 ± 2.4**	92.9 ± 1.7	96.4 ± 0.8	**13.8 ± 4.5**	22.5 ± 6.6	41.1 ± 2.5
PKMR	85.7 ± 4.1	92.4 ± 1.6	96.4 ± 0.4	9.7 ± 2.8	20.0 ± 4.8	38.8 ± 2.0
IOKR	83.6 ± 5.9	**93.6 ± 1.0**	**96.5 ± 0.4**	12.0 ± 3.0	**24.5 ± 2.9**	**43.7 ± 1.9**
Semi-supervised setting						
IOKR	86.0 ± 2.7	93.3 ± 0.7	95.7 ± 1.4	7.6 ± 2.3	13.8 ± 1.7	25.3 ± 3.0

Table 3: AUC-ROC and AUC-PR obtained for the NIPS co-authorship network inference with EM, PKMR, IOKR in the transductive setting, and with IOKR in the semi-supervised setting. p indicates the percentage of labeled examples.

- Results of Céline Brouard, JMLR 2016.

Outline

Part II : structured output prediction

- Large Margin approaches, Logistic Regression model (CRF), Joint Kernel Map
- Output Kernel Regression, Operator-valued Kernel Regression, Input Output Kernel Regression
- Open challenges : large scale approaches

Outline

References

- Álvarez, M. A. and Rosasco, L. and Lawrence, N. D., Kernels for vector-valued functions: a review, Foundations and Trends in Machine Learning, 4:3,2012. [to deepen the first part]
- Caponnetto, A. and Micchelli, C. A. and , M. and Ying, Y., Universal MultiTask Kernels, Journal of Machine Learning Research, 9,2008.
- Evgeniou, Micchelli, Pontil, Regularizing multi-tasks, 2005.
- Francesco Dinuzzo, Cheng Soon Ong, Peter V. Gehler, Gianluigi Pillonetto: Learning Output Kernels with Block Coordinate Descent. ICML 2011: 49-56 (Eligible for the project)
- N. Lim, F. d'Alché-Buc, C. Auliac, G. Michailidis, Operator-valued Kernel based Vector Autoregressive Models for Network Inference, Machine Learning Journal, 2014. (Eligible for the project)
- Micchelli and Pontil, Learning vector-valued functions, JMLR 2005.(seminal paper)
- C. Brouard, F. d'Alché-Buc, M. Szafranski, Semi-supervised link prediction with penalized output kernel regression, ICML 2011
- J. Ma, J. Zhao, J. Tuian, X. Bai, Z. Tu. Regularized Vector Field with sparse approximation, Pattern Recognition (2013) - (Eligible for the project)