

# Kernel Methods for Strings and Graphs

Jean-Philippe Vert

Jean-Philippe.Vert@ensmp.fr

Centre for Computational Biology  
Ecole des Mines de Paris, ParisTech

Kyoto University, Bioinformatics Center, February 8th, 2007

## 1 Kernels and kernel methods

## 2 Kernels for biological sequences

- Motivations
- Feature space approach
- Using generative models
- Derive from a similarity measure
- Application: remote homology detection

## 3 Kernels on graphs

- Motivations
- Construction by regularization
- The diffusion kernel
- Harmonic analysis on graphs
- Applications

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
  - Motivations
  - Feature space approach
  - Using generative models
  - Derive from a similarity measure
  - Application: remote homology detection
- 3 Kernels on graphs
  - Motivations
  - Construction by regularization
  - The diffusion kernel
  - Harmonic analysis on graphs
  - Applications

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
  - Motivations
  - Feature space approach
  - Using generative models
  - Derive from a similarity measure
  - Application: remote homology detection
- 3 Kernels on graphs
  - Motivations
  - Construction by regularization
  - The diffusion kernel
  - Harmonic analysis on graphs
  - Applications

# Kernels and Kernels Methods

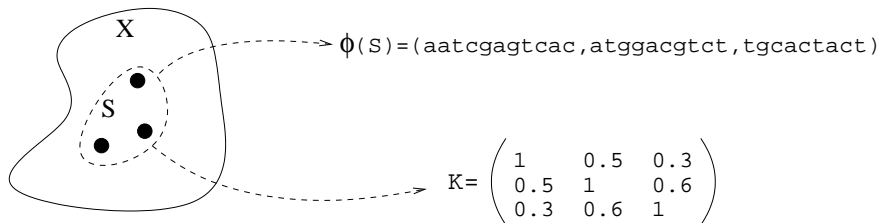
## Motivations

- Develop **versatile** algorithms to process and learn from data
- No hypothesis made regarding the **type of data** (vectors, strings, graphs, images, ...)

## The approach

- Develop methods based on **pairwise comparisons**.
- By imposing **constraints** on the pairwise comparison function (positive definite kernels), we obtain a nice **general framework for learning from data**.

# Representation by pairwise comparisons



## Idea

- Define a “comparison function”:  $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ .
- Represent a set of  $n$  data points  $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  by the  $n \times n$  matrix:

$$[K]_{ij} := K(\mathbf{x}_i, \mathbf{x}_j) .$$

# Positive Definite (p.d.) Kernels

## Definition

A **positive definite (p.d.) kernel** on the set  $\mathcal{X}$  is a function  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  **symmetric**:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}),$$

and which satisfies, for all  $N \in \mathbb{N}$ ,  $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \in \mathcal{X}^N$  et  $(a_1, a_2, \dots, a_N) \in \mathbb{R}^N$ :

$$\sum_{i=1}^N \sum_{j=1}^N a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$



## Remark

- Equivalently, a kernel  $K$  is p.d. if and only if, for any  $N \in \mathbb{N}$  and any set of points  $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \in \mathcal{X}^N$ , the **similarity matrix**  $[K]_{ij} := K(\mathbf{x}_i, \mathbf{x}_j)$  is **positive semidefinite**.
- Complete **modularity** between the **kernel** (mapping a set of points to a matrix) and the **algorithm** (processing the matrix)
- **Poor scalability** w.r.t to the dataset size ( $n^2$ ?)

## Kernels for vectors

Classical kernels for vectors ( $\mathcal{X} = \mathbb{R}^p$ ) include:

- The **linear kernel**

$$K_{lin}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}' .$$

- The **polynomial kernel**

$$K_{poly}(\mathbf{x}, \mathbf{x}') = \left( \mathbf{x}^\top \mathbf{x}' + a \right)^d .$$

- The **Gaussian RBF kernel**:

$$K_{Gaussian}(\mathbf{x}, \mathbf{x}') = \exp \left( -\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} \right) .$$

# Geometric interpretation: Kernels are inner products

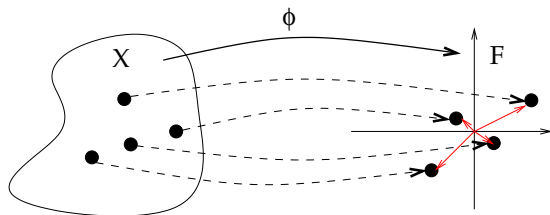
## Theorem (Aronszajn, 1950)

$K$  is a p.d. kernel on the set  $\mathcal{X}$  *if and only if* there exists a *Hilbert space*  $\mathcal{H}$  and a mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H} ,$$

such that, for any  $\mathbf{x}, \mathbf{x}'$  in  $\mathcal{X}$ :

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}} .$$



# Corollary: The kernel trick

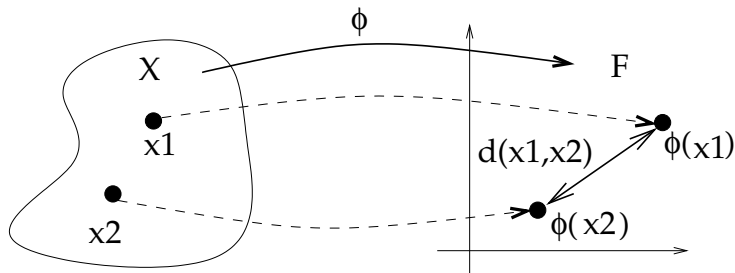
## Kernel trick

Any algorithm to process finite-dimensional vectors that can be expressed **only in terms of pairwise inner products** can be applied to potentially infinite-dimensional vectors in the feature space of a p.d. kernel by **replacing each inner product evaluation by a kernel evaluation**.

## Remark

- The proof of this proposition is trivial, because the kernel is exactly the inner product in the feature space.
- This trick has **huge practical applications**, in particular to **extend linear methods** to **non-linear** settings and **non-vector** data.
- Vectors in the feature space are only manipulated **implicitly**, through pairwise inner products.

# Kernel trick example: computing distances in the feature space



$$\begin{aligned}d_K(\mathbf{x}_1, \mathbf{x}_2)^2 &= \|\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)\|_{\mathcal{H}}^2 \\ &= \langle \Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2), \Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2) \rangle_{\mathcal{H}} \\ &= \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_1) \rangle_{\mathcal{H}} + \langle \Phi(\mathbf{x}_2), \Phi(\mathbf{x}_2) \rangle_{\mathcal{H}} - 2 \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle_{\mathcal{H}}\end{aligned}$$

$$d_K(\mathbf{x}_1, \mathbf{x}_2)^2 = K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)$$

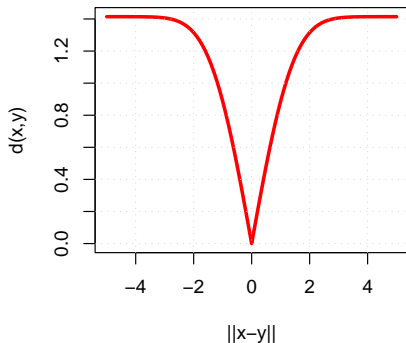
# Distance for the Gaussian kernel

- The Gaussian kernel with bandwidth  $\sigma$  on  $\mathbb{R}$  is:

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}},$$

- $K(\mathbf{x}, \mathbf{x}) = 1 = \|\Phi(\mathbf{x})\|_{\mathcal{H}}^2$ , so all points are on the unit sphere in the feature space.
- The distance between the images of two points  $\mathbf{x}$  and  $\mathbf{y}$  in the feature space is given by:

$$d_K(\mathbf{x}, \mathbf{y}) = \sqrt{2 \left[ 1 - e^{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}} \right]}$$



## RKHS definition

- To each p.d. kernel on  $\mathcal{X}$  is associated a unique **Hilbert space of function**  $\mathcal{X} \rightarrow \mathbb{R}$ , called the reproducing kernel Hilbert space (RKHS)  $\mathcal{H}$ .
- Typical functions are:

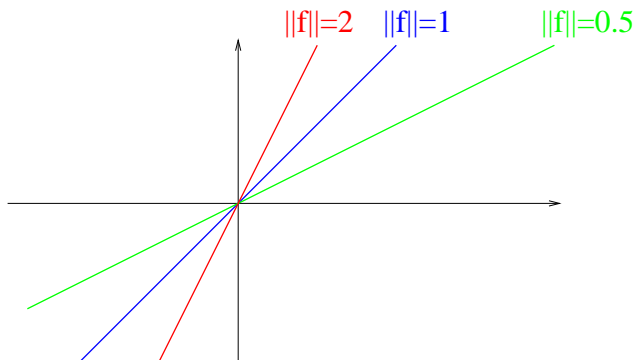
$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}) ,$$

with norm

$$\|f\|_{\mathcal{H}}^2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) .$$

# Example: Linear kernel

$$\begin{cases} K_{lin}(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^\top \mathbf{x}' . \\ f(\mathbf{x}) &= \mathbf{w}^\top \mathbf{x} , \\ \|f\|_{\mathcal{H}} &= \|\mathbf{w}\|_2 . \end{cases}$$





# Examples: Gaussian RBF kernel

$$K_{\text{Gaussian}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right),$$

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right),$$

$$\begin{aligned} \|f\|_{\mathcal{H}}^2 &= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right) \\ &= \int |\hat{f}(\omega)|^2 e^{\frac{\sigma^2 \omega^2}{2}} d\omega. \end{aligned}$$

## A simple inequality

- The norm of a function in the RKHS controls **how fast** the function varies over  $\mathcal{X}$  with respect to the **geometry defined by the kernel**:

$$|f(\mathbf{x}) - f(\mathbf{x}')| \leq \|f\|_{\mathcal{H}} \times d_K(\mathbf{x}, \mathbf{x}') .$$

- $f$  is **Lipschitz** with constant  $\|f\|_{\mathcal{H}}$  w.r.t.  $d_K$ .

## An important message

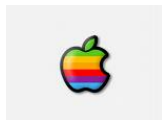
The RKHS norm is therefore a **smoothness functional**:

**Small norm  $\implies$  slow variations.**

# Pattern recognition



APPLE



APPLE



PEAR



PEAR



???



???



APPLE



APPLE



APPLE



PEAR



???

- **Input** variables  $\mathbf{x} \in \mathcal{X}$
- **Output**  $y \in \{-1, 1\}$ .
- **Training set**  $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ .

## General setting

- **Observation:**  $\{z_1, \dots, z_n\}$  where  $z_i = (\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$
- **Goal:** learn a function  $f : \mathcal{X} \rightarrow \mathbb{R}$
- **Examples:** density estimation, pattern recognition, regression, outlier detection, clustering, compression, embedding...

## Empirical risk minimization (ERM)

- 1 Define a **loss function**  $l(f, z)$  and a **space of functions**  $\mathcal{F}$ .
- 2 Minimize the **empirical average loss** over  $\mathcal{F}$ :

$$\hat{f} \in \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n l(f, z_i).$$

## General properties of ERM

- If  $\mathcal{F}$  is **not “too large”** then the ERM is **consistent** ( $\hat{f}$  is close to the best possible  $f \in \mathcal{F}$  as the number of observations increases).
- If  $\mathcal{F}$  is **not “too small”** then the best possible  $f \in \mathcal{F}$  is a **“good” solution**.
- **Challenge**: choose a “small”  $\mathcal{F}$  that contains “good” functions.

## ERM in RKHS

- Take  $\mathcal{F}$  to be a ball in the RKHS:

$$\mathcal{F}_B = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq B\} .$$

- Advantage: by **controlling the “size”** of  $\mathcal{F}$  (related to  $B$ ) the ERM principle works (**consistency** and **theoretical rates of convergence**).
- The **kernel** should be chosen s.t. some “good” functions have a small RKHS norm.

## General setting

- For pattern recognition  $\mathcal{Y} = \{-1, 1\}$ .
- Goal: estimate a function  $f : \mathcal{X} \rightarrow \mathbb{R}$  to predict  $\mathbf{y}$  from the sign of  $f(\mathbf{x})$
- The **margin** for a pair  $(\mathbf{x}, \mathbf{y})$  is  $\mathbf{y}f(\mathbf{x})$ .
- Focusing on **large margins** ensures that  $f(\mathbf{x})$  has the same sign as  $\mathbf{y}$  and a large absolute value (confidence).
- Leads to a loss function

$$l(f, (\mathbf{x}, \mathbf{y})) = \phi(\mathbf{y}f(\mathbf{x})) ,$$

where  $\phi : \mathbb{R} \rightarrow \mathbb{R}$  is non-increasing.

## Theoretical results

- The ERM estimator  $\hat{f}_n$  solves:

$$\begin{cases} \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{y}_i f(\mathbf{x}_i)) \\ \text{subject to } \|f\|_{\mathcal{H}} \leq B. \end{cases}$$

- Let  $P$  an unknown distribution over  $\mathcal{X} \times \mathcal{Y}$ , assume  $\mathcal{S} = (\mathbf{x}_i, y_i)_{i=1, \dots, n}$  i.i.d. according to  $P$ .
- Assume  $K$  upper bounded by  $\kappa$  and  $\phi$  Lipschitz with constant  $L_\phi$ .
- For the  $\phi$ -risk  $R_\phi(f) = \mathbf{E} \phi(Yf(X))$  we have:

$$\mathbf{E} R_\phi(\hat{f}_n) \leq \inf_{f \in \mathcal{F}_B} R_\phi(f) + \frac{8L_\phi \kappa B}{\sqrt{n}}.$$



## Reformulation as penalized minimization

- We must solve the constrained minimization problem:

$$\begin{cases} \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{y}_i f(\mathbf{x}_i)) \\ \text{subject to } \|f\|_{\mathcal{H}} \leq B. \end{cases}$$

- To make this practical we assume that  $\phi$  is convex.
- The problem is then a convex problem in  $f$  for which strong duality holds. In particular  $f$  solves the problem if and only if it solves for some dual parameter  $\lambda$  the unconstrained problem:

$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{y}_i f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2 \right\},$$

and complimentary slackness holds ( $\lambda = 0$  or  $\|f\|_{\mathcal{H}} = B$ ).

# Optimization in RKHS

- By the **representer theorem**, the solution of the unconstrained problem can be expanded as:

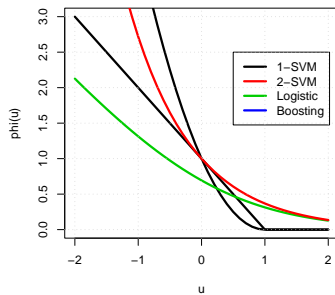
$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}) .$$

- Plugging into the original problem we obtain the following **unconstrained and convex optimization problem in  $\mathbb{R}^n$** :

$$\min_{\alpha \in \mathbb{R}^n} \left\{ \frac{1}{n} \sum_{i=1}^n \phi \left( \mathbf{y}_i \sum_{j=1}^n \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \right) + \lambda \sum_{i,j=1}^n \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \right\} .$$

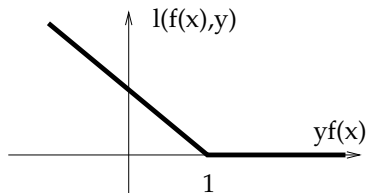
- This can be **implemented** using general packages for **convex optimization** or specific algorithms (e.g., for SVM).

# Loss function examples



Method	$\phi(u)$
Kernel logistic regression	$\log(1 + e^{-u})$
Support vector machine (1-SVM)	$\max(1 - u, 0)$
Support vector machine (2-SVM)	$\max(1 - u, 0)^2$
Boosting	$e^{-u}$

# Example: Support vector machines



- The loss function is the **hinge loss**:

$$\phi_{\text{hinge}}(u) = \max(1 - u, 0) .$$

- SVM solve the problem:

$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^n \phi_{\text{hinge}}(\mathbf{y}_i f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2 \right\} .$$

# Problem reformulation (2/2)

## Finite-dimensional expansion

Replacing  $\hat{f}$  by

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}),$$

the problem can be rewritten as an optimization problem in  $\alpha$ :

$$\min_{\alpha \in \mathbb{R}^n, \xi \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^n \xi_i + \lambda \alpha^\top K \alpha,$$

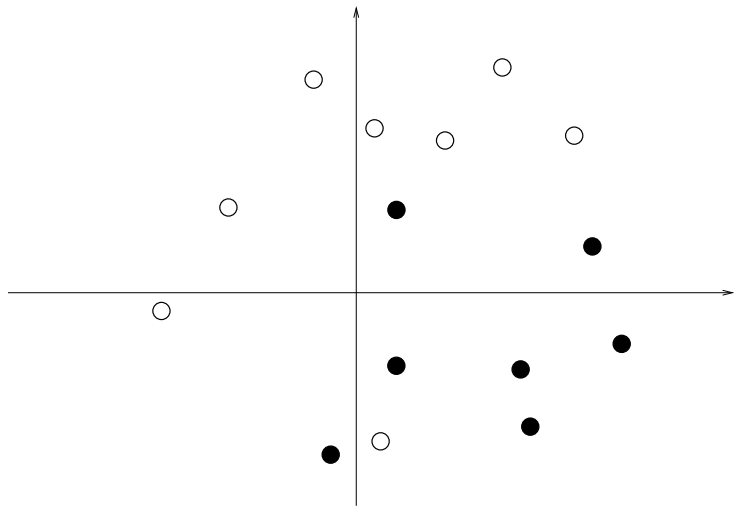
subject to:

$$\begin{cases} y_i \sum_{j=1}^n \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \xi_i - 1 \geq 0, & \text{for } i = 1, \dots, n, \\ \xi_i \geq 0, & \text{for } i = 1, \dots, n. \end{cases}$$

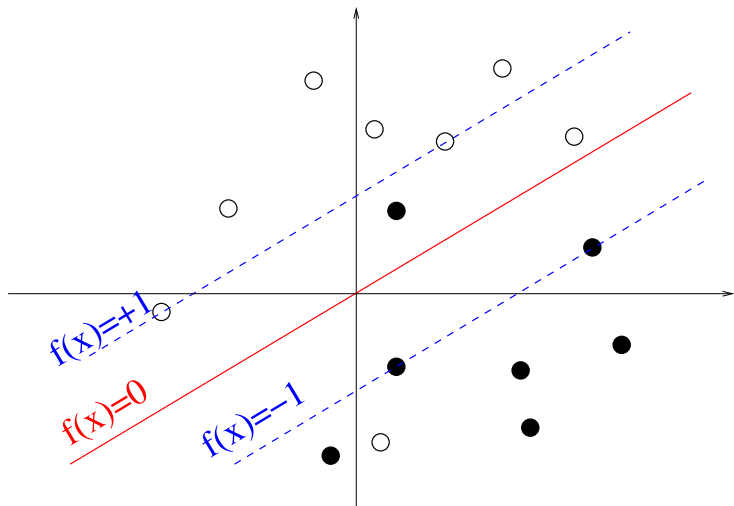
## Remarks

- This is a classical **quadratic program** (minimization of a convex quadratic function with linear constraints) for which any out-of-the-box optimization package can be used.
- The **dimension** of the problem and the **number of constraints**, however, are  $2n$  where  $n$  is the number of points. General-purpose QP solvers will have difficulties when  $n$  exceeds a few thousands.
- Solving the **dual** of this problem (also a QP) will be more convenient and lead to faster algorithms (due to the sparsity of the final solution).

# Geometric interpretation

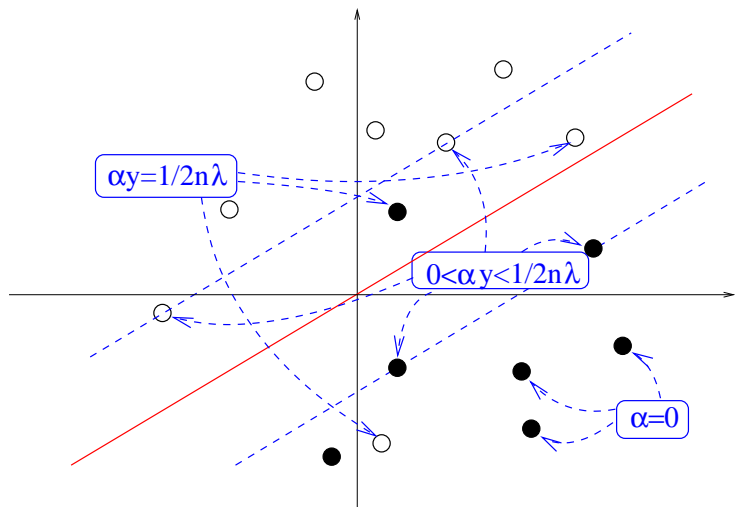


# Geometric interpretation





# Geometric interpretation



# Kernel methods: Summary

- Positive definite kernels can be thought of as:
  - **Embedding** the data to a Hilbert space,
  - Defining a **Hilbert space of real-valued functions** over the data.
- The **kernel trick** allows to extend many linear algorithms to **non-linear settings** and to **general data** (even non-vectorial).
- The **norm in the RKHS** can be used as **regularization** for empirical risk minimization. This is **theoretically justified** and leads to **efficient algorithms** (often finite-dimensional convex problem thanks to the representer theorem).

## Kernels and RKHS: general



N. Aronszajn.

Theory of reproducing kernels.

*Trans. Am. Math. Soc.*, 68:337 – 404, 1950.



C. Berg, J. P. R. Christensen, and P. Ressel.

*Harmonic analysis on semigroups.*

Springer-Verlag, New-York, 1984.



G. Wahba.

*Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*.

SIAM, Philadelphia, 1990.

## Learning with kernels



V. N. Vapnik.

*Statistical Learning Theory.*

Wiley, New-York, 1998.



B. Schölkopf and A. J. Smola.

*Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.*

MIT Press, Cambridge, MA, 2002.



J. Shawe-Taylor and N. Cristianini.

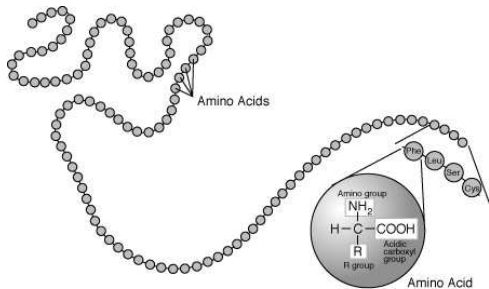
*Kernel Methods for Pattern Analysis.*

Cambridge University Press, 2004.

# Kernels for biological sequences

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
  - **Motivations**
  - Feature space approach
  - Using generative models
  - Derive from a similarity measure
  - Application: remote homology detection
- 3 Kernels on graphs

# Proteins



**A** : Alanine

**F** : Phenylalanine

**E** : Acide glutamique

**T** : Threonine

**H** : Histidine

**I** : Isoleucine

**D** : Acide aspartique

**V** : Valine

**P** : Proline

**K** : Lysine

**C** : Cysteine

**V** : Thyrosine

**S** : Sérine

**G** : Glycine

**L** : Leucine

**M** : Méthionine

**R** : Arginine

**N** : Asparagine

**W** : Tryptophane

**Q** : Glutamine

# Challenges with protein sequences

- A protein sequences can be seen as a **variable-length sequence** over the **20-letter alphabet** of amino-acids, e.g., insuline:  
FVNQHLCGSHLVEALYLVCGERGFFYTPKA
- These sequences are produced at a fast rate (result of the **sequencing programs**)
- Need for algorithms to **compare, classify, analyze** these sequences
- Applications: classification into **functional or structural** classes, prediction of **cellular localization** and **interactions**, ...



# Example: supervised sequence classification

## Data (training)

- **Secreted proteins:**

```
MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNI EA...
MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...
MALHTVLIMLSLLPMLQAQNPEHANITIGEPITNETLGWL...
...
```

- **Non-secreted proteins:**

```
MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVN LGVG...
MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...
MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..
...
```

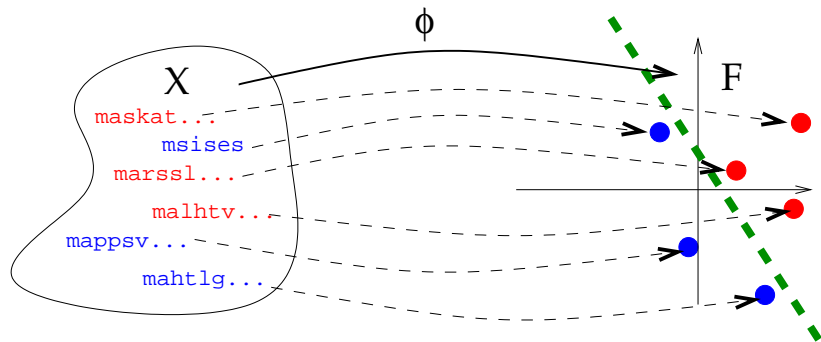
## Goal

- Build a **classifier** to **predict** whether new proteins are secreted or not.

# Supervised classification with vector embedding

## The idea

- Map each string  $x \in \mathcal{X}$  to a **vector**  $\Phi(x) \in \mathbb{R}^p$ .
- Train a **classifier for vectors** on the images  $\Phi(x_1), \dots, \Phi(x_n)$  of the training set (nearest neighbor, linear perceptron, logistic regression, support vector machine...)



## Generalities

- **Kernel methods** have been widely investigated since Jaakkola et al.'s seminal paper (1998).
- What is a **good kernel**?
  - it should be **mathematically valid** (symmetric, p.d. or c.p.d.)
  - **fast to compute**
  - **adapted to the problem** (give good performances), e.g., the unknown decision function should be smooth w.r.t. to the norm induced by the kernel.

## Kernel engineering strategies

- Define a (possibly high-dimensional) **feature space** of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels
- Derive a kernel from a **generative model**
  - Fisher kernel
  - Mutual information kernel
  - Marginalized kernel
- Derive a kernel from a **similarity measure**
  - Local alignment kernel

## Kernel engineering strategies

- Define a (possibly high-dimensional) **feature space** of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels
- Derive a kernel from a **generative model**
  - Fisher kernel
  - Mutual information kernel
  - Marginalized kernel
- Derive a kernel from a **similarity measure**
  - Local alignment kernel

## Kernel engineering strategies

- Define a (possibly high-dimensional) **feature space** of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels
- Derive a kernel from a **generative model**
  - Fisher kernel
  - Mutual information kernel
  - Marginalized kernel
- Derive a kernel from a **similarity measure**
  - Local alignment kernel

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
  - Motivations
  - **Feature space approach**
  - Using generative models
  - Derive from a similarity measure
  - Application: remote homology detection
- 3 Kernels on graphs

# Vector embedding for strings

## The idea

Represent each sequence  $\mathbf{x}$  by a **fixed-length numerical vector**  $\Phi(\mathbf{x}) \in \mathbb{R}^p$ . How to perform this embedding?

## Physico-chemical kernel

Extract **relevant features**, such as:

- length of the sequence
- **time series analysis of numerical physico-chemical properties** of amino-acids along the sequence (e.g., polarity, hydrophobicity), using for example:
  - Fourier transforms (Wang et al., 2004)
  - Autocorrelation functions (Zhang et al., 2003)

$$r_j = \frac{1}{n-j} \sum_{i=1}^{n-j} h_i h_{i+j}$$



# Vector embedding for strings

## The idea

Represent each sequence  $\mathbf{x}$  by a **fixed-length numerical vector**  $\Phi(\mathbf{x}) \in \mathbb{R}^p$ . How to perform this embedding?

## Physico-chemical kernel

Extract **relevant features**, such as:

- length of the sequence
- **time series analysis of numerical physico-chemical properties** of amino-acids along the sequence (e.g., polarity, hydrophobicity), using for example:
  - Fourier transforms (Wang et al., 2004)
  - Autocorrelation functions (Zhang et al., 2003)

$$r_j = \frac{1}{n-j} \sum_{i=1}^{n-j} h_i h_{i+j}$$

## The approach

Alternatively, index the feature space by fixed-length strings, i.e.,

$$\Phi(\mathbf{x}) = (\Phi_u(\mathbf{x}))_{u \in \mathcal{A}^k}$$

where  $\Phi_u(\mathbf{x})$  can be:

- the number of occurrences of  $u$  in  $\mathbf{x}$  (without gaps) : **spectrum kernel** (Leslie et al., 2002)
- the number of occurrences of  $u$  in  $\mathbf{x}$  up to  $m$  mismatches (without gaps) : **mismatch kernel** (Leslie et al., 2004)
- the number of occurrences of  $u$  in  $\mathbf{x}$  allowing gaps, with a weight decaying exponentially with the number of gaps : **substring kernel** (Lohdi et al., 2002)

# Example: spectrum kernel (1/2)

## Kernel definition

- The 3-spectrum of

$\mathbf{x} = \text{CGGSLIAMMWFGV}$

is:

$(\text{CGG}, \text{GGS}, \text{GSL}, \text{SLI}, \text{LIA}, \text{IAM}, \text{AMM}, \text{MMW}, \text{MWF}, \text{WFG}, \text{FGV})$  .

- Let  $\Phi_u(\mathbf{x})$  denote the number of occurrences of  $u$  in  $\mathbf{x}$ . The  $k$ -spectrum kernel is:

$$K(\mathbf{x}, \mathbf{x}') := \sum_{u \in \mathcal{A}^k} \Phi_u(\mathbf{x}) \Phi_u(\mathbf{x}') .$$

# Example: spectrum kernel (2/2)

## Implementation

- The computation of the kernel is formally a sum over  $|\mathcal{A}|^k$  terms, but at most  $|\mathbf{x}| - k + 1$  terms are non-zero in  $\Phi(\mathbf{x}) \implies$   
**Computation in  $O(|\mathbf{x}| + |\mathbf{x}'|)$**  with pre-indexation of the strings.
- Fast classification of a sequence  $\mathbf{x}$  in  $O(|\mathbf{x}|)$ :

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_u w_u \Phi_u(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|-k+1} w_{x_i \dots x_{i+k-1}}.$$

## Remarks

- Work with any string (natural language, time series...)
- **Fast and scalable**, a good default method for string classification.
- Variants allow matching of  $k$ -mers up to  $m$  **mismatches**.

## Example 2: Substring kernel (1/5)

### Definition

- For  $1 \leq k \leq n \in \mathbb{N}$ , we denote by  $\mathcal{I}(k, n)$  the set of **sequences of indices**  $\mathbf{i} = (i_1, \dots, i_k)$ , with  $1 \leq i_1 < i_2 < \dots < i_k \leq n$ .
- For a string  $\mathbf{x} = x_1 \dots x_n \in \mathcal{X}$  of length  $n$ , for a sequence of indices  $\mathbf{i} \in \mathcal{I}(k, n)$ , we define a **substring** as:

$$\mathbf{x}(\mathbf{i}) := x_{i_1} x_{i_2} \dots x_{i_k}.$$

- The **length** of the substring is:

$$l(\mathbf{i}) = i_k - i_1 + 1.$$

## Example 2: Substring kernel (2/5)

### Example

ABRACADABRA

- $\mathbf{i} = (3, 4, 7, 8, 10)$
- $\mathbf{x}(\mathbf{i}) = \text{RADAR}$
- $l(\mathbf{i}) = 10 - 3 + 1 = 8$

## Example 2: Substring kernel (3/5)

### The kernel

- Let  $k \in \mathbb{N}$  and  $\lambda \in \mathbb{R}^+$  fixed. For all  $\mathbf{u} \in \mathcal{A}^k$ , let  $\Phi_{\mathbf{u}} : \mathcal{X} \rightarrow \mathbb{R}$  be defined by:

$$\forall \mathbf{x} \in \mathcal{X}, \quad \Phi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i} \in \mathcal{I}(k, |\mathbf{x}|): \mathbf{x}(\mathbf{i}) = \mathbf{u}} \lambda^{|\mathbf{i}|}.$$

- The **substring kernel** is the p.d. kernel defined by:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K_{k, \lambda}(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{u} \in \mathcal{A}^k} \Phi_{\mathbf{u}}(\mathbf{x}) \Phi_{\mathbf{u}}(\mathbf{x}').$$

## Example 2: Substring kernel (4/5)

### Example

u	ca	ct	at	ba	bt	cr	ar	br
$\Phi_u(\text{cat})$	$\lambda^2$	$\lambda^3$	$\lambda^2$	0	0	0	0	0
$\Phi_u(\text{car})$	$\lambda^2$	0	0	0	0	$\lambda^3$	$\lambda^2$	0
$\Phi_u(\text{bat})$	0	0	$\lambda^2$	$\lambda^2$	$\lambda^3$	0	0	0
$\Phi_u(\text{bar})$	0	0	0	$\lambda^2$	0	0	$\lambda^2$	$\lambda^3$

$$\begin{cases} K(\text{cat}, \text{cat}) = K(\text{car}, \text{car}) = 2\lambda^4 + \lambda^6 \\ K(\text{cat}, \text{car}) = \lambda^4 \\ K(\text{cat}, \text{bar}) = 0 \end{cases}$$



## Example 2: Substring kernel (5/5)

### Kernel computation

- We need to compute, for any pair  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , the kernel:

$$\begin{aligned} K_{n,\lambda}(\mathbf{x}, \mathbf{x}') &= \sum_{\mathbf{u} \in \mathcal{A}^k} \Phi_{\mathbf{u}}(\mathbf{x}) \Phi_{\mathbf{u}}(\mathbf{x}') \\ &= \sum_{\mathbf{u} \in \mathcal{A}^k} \sum_{\mathbf{i}: \mathbf{x}(\mathbf{i})=\mathbf{u}} \sum_{\mathbf{i}': \mathbf{x}'(\mathbf{i}')=\mathbf{u}} \lambda^{l(\mathbf{i})+l(\mathbf{i}')}. \end{aligned}$$

- Enumerating the substrings is **too slow** (of order  $|\mathbf{x}|^k$ ).
- The kernel can be factorized and computed by **dynamic programming** in  $O(|\mathbf{x}| \times |\mathbf{x}'|)$ .

# Dictionary-based indexation

## The approach

- Chose a **dictionary** of sequences  $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$
- Chose a **measure of similarity**  $s(\mathbf{x}, \mathbf{x}')$
- Define the mapping  $\Phi_{\mathcal{D}}(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_i))_{\mathbf{x}_i \in \mathcal{D}}$

## Examples

This includes:

- **Motif kernels** (Logan et al., 2001): the dictionary is a library of motifs, the similarity function is a matching function
- **Pairwise kernel** (Liao & Noble, 2003): the dictionary is the training set, the similarity is a classical measure of similarity between sequences.

# Dictionary-based indexation

## The approach




- Chose a **dictionary** of sequences  $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$
- Chose a **measure of similarity**  $s(\mathbf{x}, \mathbf{x}')$
- Define the mapping  $\Phi_{\mathcal{D}}(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_i))_{\mathbf{x}_i \in \mathcal{D}}$

## Examples

This includes:

- **Motif kernels** (Logan et al., 2001): the dictionary is a library of motifs, the similarity function is a matching function
- **Pairwise kernel** (Liao & Noble, 2003): the dictionary is the training set, the similarity is a classical measure of similarity between sequences.

## Substring kernels

-  C. Leslie, E. Eskin, and W.S. Noble.  
The spectrum kernel: a string kernel for SVM protein classification.  
In *PSB 2002*, pages 564–575. World Scientific, 2002.
-  H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins.  
Text classification using string kernels.  
*J. Mach. Learn. Res.*, 2:419–444, 2002.
-  C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble.  
Mismatch string kernels for discriminative protein classification.  
*Bioinformatics*, 20(4):467–476, 2004.

## Dictionary-based string kernels



B. Logan, P. Moreno, B. Suzek, Z. Weng, and S. Kasif.  
A Study of Remote Homology Detection.  
Technical Report CRL 2001/05, Compaq Cambridge Research  
laboratory, June 2001.

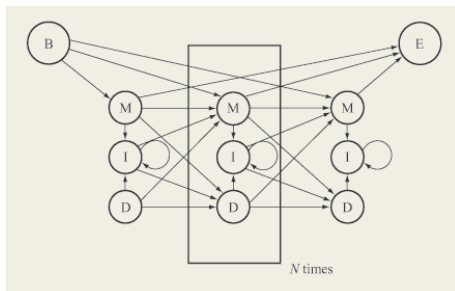


L. Liao and W.S. Noble.  
Combining Pairwise Sequence Similarity and Support Vector  
Machines for Detecting Remote Protein Evolutionary and  
Structural Relationships.  
*J. Comput. Biol.*, 10(6):857–868, 2003.

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
  - Motivations
  - Feature space approach
  - **Using generative models**
  - Derive from a similarity measure
  - Application: remote homology detection
- 3 Kernels on graphs

# Probabilistic models for sequences

**Probabilistic modeling** of biological sequences is older than kernel designs. Important models include **HMM** for protein sequences, **SCFG** for RNA sequences.



## Parametric model

A **model** is a family of distribution

$$\{P_{\theta}, \theta \in \Theta \subset \mathbb{R}^m\} \subset \mathcal{M}_1^+(\mathcal{X})$$

# Strategy 1: Fisher kernel

## Definition

- Fix a parameter  $\theta_0 \in \Theta$  (e.g., by maximum likelihood over a training set of sequences)
- For each sequence  $\mathbf{x}$ , compute the Fisher score vector:

$$\Phi_{\theta_0}(\mathbf{x}) = \nabla_{\theta} \log P_{\theta}(\mathbf{x})|_{\theta=\theta_0} .$$

- Form the kernel (Jaakkola et al., 1998):

$$K(\mathbf{x}, \mathbf{x}') = \Phi_{\theta_0}(\mathbf{x})^{\top} I(\theta_0)^{-1} \Phi_{\theta_0}(\mathbf{x}') ,$$

where  $I(\theta_0) = E_{\theta_0} [\Phi_{\theta_0}(\mathbf{x})\Phi_{\theta_0}(\mathbf{x})^{\top}]$  is the Fisher information matrix.



# Fisher kernel properties

- The Fisher score describes how **each parameter contributes** to the process of generating a particular example
- The Fisher kernel is **invariant** under change of parametrization of the model
- A kernel classifier employing the Fisher kernel derived from a model that contains the label as a latent variable is, asymptotically, **at least as good a classifier as the MAP labelling** based on the model (Jaakkola and Haussler, 1998).
- A variant of the Fisher kernel (called the Tangent of Posterior kernel) can also improve over the direct posterior classification by helping to **correct the effect of estimation errors** in the parameter (Tsuda et al., 2002).

- $\Phi_{\theta_0}(\mathbf{x})$  can be computed explicitly for many models (e.g., HMMs)
- $I(\theta_0)$  is often replaced by the identity matrix
- Several different models (i.e., different  $\theta_0$ ) can be trained and combined
- Feature vectors are explicitly computed

## Fisher kernels

 T. Jaakkola, M. Diekhans, and D. Haussler.

A Discriminative Framework for Detecting Remote Protein Homologies.

*J. Comput. Biol.*, 7(1,2):95–114, 2000.

 K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller.

A new discriminative kernel from probabilistic models.

*Neural Computation*, 14(10):2397–2414, 2002.

## Strategy 2: Mutual information kernels

### Definition

- Chose a prior  $w(d\theta)$  on the measurable set  $\Theta$
- Form the kernel (Seeger, 2002):

$$K(\mathbf{x}, \mathbf{x}') = \int_{\theta \in \Theta} P_{\theta}(\mathbf{x}) P_{\theta}(\mathbf{x}') w(d\theta) .$$

- **No explicit computation** of a finite-dimensional feature vector
- $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{L_2(w)}$  with

$$\phi(\mathbf{x}) = (P_{\theta}(\mathbf{x}))_{\theta \in \Theta} .$$

# Example: coin toss

- Let  $P_\theta(X = 1) = \theta$  and  $P_\theta(X = 0) = 1 - \theta$  a model for random coin toss, with  $\theta \in [0, 1]$ .
- Let  $d\theta$  be the Lebesgue measure on  $[0, 1]$
- The mutual information kernel between  $\mathbf{x} = 001$  and  $\mathbf{x}' = 1010$  is:

$$\begin{cases} P_\theta(\mathbf{x}) &= \theta(1-\theta)^2, \\ P_\theta(\mathbf{x}') &= \theta^2(1-\theta)^2, \end{cases}$$

$$K(\mathbf{x}, \mathbf{x}') = \int_0^1 \theta^3 (1-\theta)^4 d\theta = \frac{3!4!}{8!} = \frac{1}{280}.$$

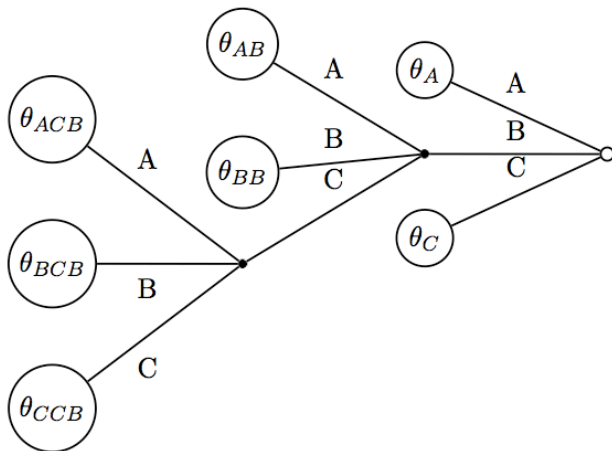
## Definition

A context-tree model is a **variable-memory Markov chain**:

$$P_{\mathcal{D},\theta}(\mathbf{x}) = P_{\mathcal{D},\theta}(x_1 \dots x_D) \prod_{i=D+1}^n P_{\mathcal{D},\theta}(x_i | x_{i-D} \dots x_{i-1})$$

- $\mathcal{D}$  is a suffix tree
- $\theta \in \Sigma^{\mathcal{D}}$  is a set of conditional probabilities (multinomials)

# Context-tree model: example



$$P(AABACBACC) = P(AAB)\theta_{AB}(A)\theta_A(C)\theta_C(B)\theta_{ACB}(A)\theta_A(C)\theta_C(A).$$

## Theorem (Cuturi et al., 2004)

- For particular choices of priors, the context-tree kernel:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{\mathcal{D}} \int_{\theta \in \Sigma^{\mathcal{D}}} P_{\mathcal{D}, \theta}(\mathbf{x}) P_{\mathcal{D}, \theta}(\mathbf{x}') w(d\theta | \mathcal{D}) \pi(\mathcal{D})$$

can be computed in  $O(|\mathbf{x}| + |\mathbf{x}'|)$  with a variant of the **Context-Tree Weighting algorithm**.

- This is a **valid mutual information kernel**.
- The similarity is related to information-theoretical measure of **mutual information** between strings.



## Mutual information kernels



M. Seeger.

Covariance Kernels from Bayesian Generative Models.

*In Adv. Neural Inform. Process. Syst.*, volume 14, pages 905–912, 2002.



M. Cuturi and J.-P. Vert.

The context-tree kernel for strings.

*Neural Network.*, 18(4):1111–1123, 2005.



M. Cuturi, K. Fukumizu, and J.P. Vert.

Semigroup Kernels on Measures.

*J. Mach. Learn. Res.*, 6:1169–1198, 2005.

## Strategy 3: Marginalized kernels

### Definition

- For any **observed data**  $\mathbf{x} \in \mathcal{X}$ , let a **latent variable**  $\mathbf{y} \in \mathcal{Y}$  be associated probabilistically through a **conditional probability**  $P_{\mathbf{x}}(d\mathbf{y})$ .
- Let  $K_{\mathcal{Z}}$  be a **kernel for the complete data**  $\mathbf{z} = (\mathbf{x}, \mathbf{y})$
- Then the following kernel is a valid kernel on  $\mathcal{X}$ , called a **marginalized kernel** (Tsuda et al., 2002):

$$\begin{aligned} K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') &:= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') \\ &= \int \int K_{\mathcal{Z}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) P_{\mathbf{x}}(d\mathbf{y}) P_{\mathbf{x}'}(d\mathbf{y}') . \end{aligned}$$

# Marginalized kernels: proof of positive definiteness

- $K_{\mathcal{Z}}$  is p.d. on  $\mathcal{Z}$ . Therefore there exists a Hilbert space  $\mathcal{H}$  and  $\Phi_{\mathcal{Z}} : \mathcal{Z} \rightarrow \mathcal{H}$  such that:

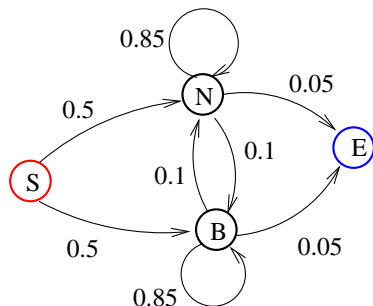
$$K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') = \langle \Phi_{\mathcal{Z}}(\mathbf{z}), \Phi_{\mathcal{Z}}(\mathbf{z}') \rangle_{\mathcal{H}} .$$

- Marginalizing therefore gives:

$$\begin{aligned} K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') \\ &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} \langle \Phi_{\mathcal{Z}}(\mathbf{z}), \Phi_{\mathcal{Z}}(\mathbf{z}') \rangle_{\mathcal{H}} \\ &= \langle E_{P_{\mathbf{x}}(d\mathbf{y})} \Phi_{\mathcal{Z}}(\mathbf{z}), E_{P_{\mathbf{x}'}(d\mathbf{y}')} \Phi_{\mathcal{Z}}(\mathbf{z}') \rangle_{\mathcal{H}} , \end{aligned}$$

therefore  $K_{\mathcal{X}}$  is p.d. on  $\mathcal{X}$ .  $\square$

# Example: HMM for normal/biased coin toss



- Normal ( $N$ ) and biased ( $B$ ) coins (not observed)

- Observed output are 0/1 with probabilities:

$$\begin{cases} \pi(0|N) = 1 - \pi(1|N) = 0.5, \\ \pi(0|B) = 1 - \pi(1|B) = 0.8. \end{cases}$$

- Example of realization (complete data):

NNNNNBBBBBBBBBNNNNNNNNNNNNBBBBBBB  
1001011101111010010111001111011

# 1-spectrum kernel on complete data

- If both  $\mathbf{x} \in \mathcal{A}^*$  and  $\mathbf{y} \in \mathcal{S}^*$  were observed, we might rather use the 1-spectrum kernel on the complete data  $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ :

$$K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') = \sum_{(a,s) \in \mathcal{A} \times \mathcal{S}} n_{a,s}(\mathbf{z}) n_{a,s}(\mathbf{z}'),$$

where  $n_{a,s}(\mathbf{x}, \mathbf{y})$  for  $a = 0, 1$  and  $s = N, B$  is the number of occurrences of  $s$  in  $\mathbf{y}$  which emit  $a$  in  $\mathbf{x}$ .

- Example:

$$\begin{aligned}\mathbf{z} &= 1001011101111010010111001111011, \\ \mathbf{z}' &= 0011010110011111011010111101100101,\end{aligned}$$

$$\begin{aligned}K_{\mathcal{Z}}(\mathbf{z}, \mathbf{z}') &= n_0(\mathbf{z}) n_0(\mathbf{z}') + n_1(\mathbf{z}) n_1(\mathbf{z}') + n_N(\mathbf{z}) n_N(\mathbf{z}') + n_B(\mathbf{z}) n_B(\mathbf{z}') \\ &= 7 \times 15 + 9 \times 12 + 13 \times 6 + 2 \times 1 = 293.\end{aligned}$$

# 1-spectrum marginalized kernel on observed data

- The marginalized kernel for observed data is:

$$\begin{aligned} K_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') &= \sum_{\mathbf{y}, \mathbf{y}' \in \mathcal{S}^*} K_{\mathcal{Z}}((\mathbf{x}, \mathbf{y}), (\mathbf{x}, \mathbf{y}')) P(\mathbf{y}|\mathbf{x}) P(\mathbf{y}'|\mathbf{x}') \\ &= \sum_{\mathbf{y}, \mathbf{y}' \in \mathcal{S}^*} \left[ \sum_{(a,s) \in \mathcal{A} \times \mathcal{S}} n_{a,s}(\mathbf{z}) n_{a,s}(\mathbf{z}') \right] P(\mathbf{y}|\mathbf{x}) P(\mathbf{y}'|\mathbf{x}') \\ &= \sum_{(a,s) \in \mathcal{A} \times \mathcal{S}} \Phi_{a,s}(\mathbf{x}) \Phi_{a,s}(\mathbf{x}'), \end{aligned}$$

with

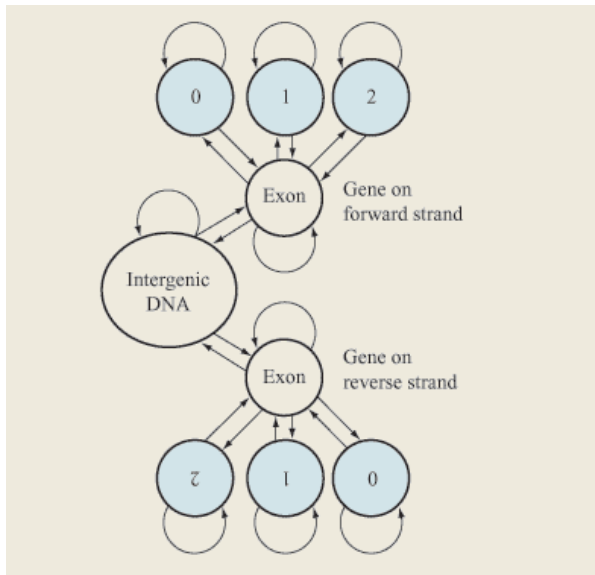
$$\Phi_{a,s}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) n_{a,s}(\mathbf{x}, \mathbf{y})$$

# Computation of the 1-spectrum marginalized kernel

$$\begin{aligned}\Phi_{a,s}(\mathbf{x}) &= \sum_{\mathbf{y} \in \mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) n_{a,s}(\mathbf{x}, \mathbf{y}) \\ &= \sum_{\mathbf{y} \in \mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) \left\{ \sum_{i=1}^n \delta(x_i, a) \delta(y_i, s) \right\} \\ &= \sum_{i=1}^n \delta(x_i, a) \left\{ \sum_{\mathbf{y} \in \mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) \delta(y_i, s) \right\} \\ &= \sum_{i=1}^n \delta(x_i, a) P(y_i = s|\mathbf{x}).\end{aligned}$$

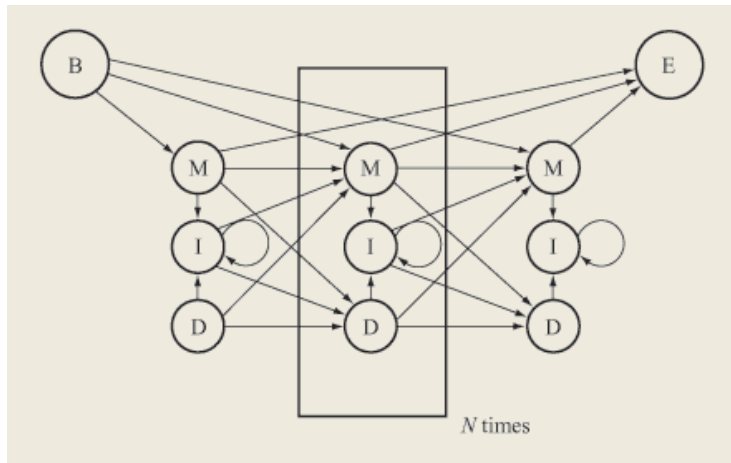
and  $P(y_i = s|\mathbf{x})$  can be computed efficiently by forward-backward algorithm!

# HMM example (DNA)

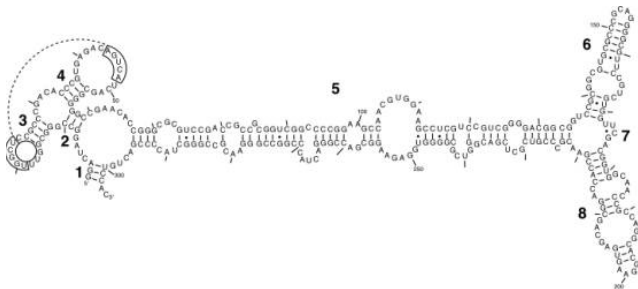




# HMM example (protein)



# SCFG for RNA sequences



## SFCG rules

- $S \rightarrow SS$
- $S \rightarrow aSa$
- $S \rightarrow aS$
- $S \rightarrow a$

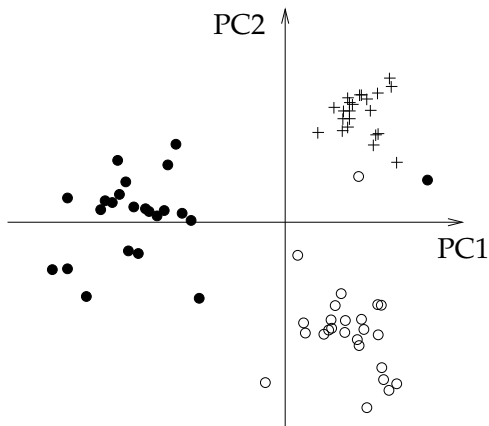
## Marginalized kernel (Kin et al., 2002)

- Feature: number of occurrences of each (base,state) combination
- Marginalization using classical inside/outside algorithm

## Examples





- Spectrum kernel on the hidden states of a HMM for **protein sequences** (Tsuda et al., 2002)
- Kernels for **RNA sequences** based on SCFG (Kin et al., 2002)
- Kernels for **graphs** based on random walks on graphs (Kashima et al., 2003)
- Kernels for **multiple alignments** based on phylogenetic models (Vert et al., 2006)

# Marginalized kernels: example



A set of 74 human tRNA sequences is analyzed using a kernel for sequences (the second-order marginalized kernel based on SCFG). This set of tRNAs contains three classes, called Ala-AGC (*white circles*), Asn-GTT (*black circles*) and Cys-GCA (*plus symbols*) (from Tsuda et al., 2002).

## Marginalized kernels

-  K. Tsuda, T. Kin, and K. Asai.  
Marginalized Kernels for Biological Sequences.  
*Bioinformatics*, 18:S268–S275, 2002.
-  T. Kin, K. Tsuda, and K. Asai.  
Marginalized kernels for RNA sequence data analysis.  
In *GIW 2002*, pages 112–122, 2002.
-  H. Kashima, K. Tsuda, and A. Inokuchi.  
Marginalized Kernels between Labeled Graphs.  
In *ICML'03*, pages 321–328, 2003.
-  J.-P. Vert, R. Thurman, and W. S. Noble.  
Kernels for gene regulatory regions.  
In *NIPS'05*, volume 18, pages 1401–1408, 2006.

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
  - Motivations
  - Feature space approach
  - Using generative models
  - **Derive from a similarity measure**
  - Application: remote homology detection
- 3 Kernels on graphs

## Motivation

How to compare 2 sequences?

$\mathbf{x}_1 = \text{CGGSLIAMMWFGV}$

$\mathbf{x}_2 = \text{CLIVMMNRLMWFGV}$

Find a good **alignment**:

CGGSLIAMM----WFGV

|...|||||...|||

C---LIVMMNRLMWFGV

# Alignment score

In order to quantify the relevance of an alignment  $\pi$ , define:

- a **substitution matrix**  $S \in \mathbb{R}^{\mathcal{A} \times \mathcal{A}}$
- a **gap penalty** function  $g : \mathbb{N} \rightarrow \mathbb{R}$

Any alignment is then scored as follows

```
CGGSLIAMM----WFGV
|...|||||...||||
C---LIVMMNRLMWFVG
```

$$s_{S,g}(\pi) = S(C, C) + S(L, L) + S(I, I) + S(A, V) + 2S(M, M) \\ + S(W, W) + S(F, F) + S(G, G) + S(V, V) - g(3) - g(4)$$



# Local alignment kernel

## Smith-Waterman score

- The widely-used Smith-Waterman local alignment score is defined by:

$$SW_{S,g}(\mathbf{x}, \mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x}, \mathbf{y})} s_{S,g}(\pi).$$

- It is symmetric, but not positive definite...

## LA kernel

The **local alignment kernel**:

$$K_{LA}^{(\beta)}(\mathbf{x}, \mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x}, \mathbf{y})} \exp(\beta s_{S,g}(\mathbf{x}, \mathbf{y}, \pi)),$$

is symmetric positive definite (Vert et al., 2004).

# Local alignment kernel

## Smith-Waterman score

- The widely-used Smith-Waterman local alignment score is defined by:

$$SW_{S,g}(\mathbf{x}, \mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x}, \mathbf{y})} s_{S,g}(\pi).$$

- It is symmetric, but not positive definite...

## LA kernel

The **local alignment kernel**:

$$K_{LA}^{(\beta)}(\mathbf{x}, \mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x}, \mathbf{y})} \exp(\beta s_{S,g}(\mathbf{x}, \mathbf{y}, \pi)),$$

is symmetric positive definite (Vert et al., 2004).

# LA kernel is p.d.: proof

- If  $K_1$  and  $K_2$  are p.d. kernels for strings, then their **convolution** defined by:

$$K_1 \star K_2(\mathbf{x}, \mathbf{y}) := \sum_{\mathbf{x}_1 \mathbf{x}_2 = \mathbf{x}, \mathbf{y}_1 \mathbf{y}_2 = \mathbf{y}} K_1(\mathbf{x}_1, \mathbf{y}_1) K_2(\mathbf{x}_2, \mathbf{y}_2)$$

is also p.d. (Haussler, 1999).

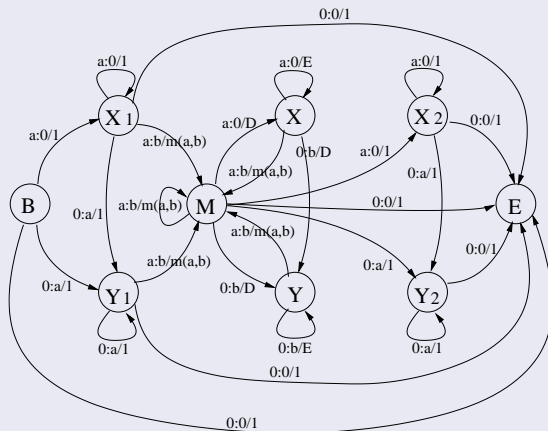
- LA kernel is p.d. because it is a **convolution kernel** (Haussler, 1999):

$$K_{LA}^{(\beta)} = \sum_{n=0}^{\infty} K_0 \star \left( K_a^{(\beta)} \star K_g^{(\beta)} \right)^{(n-1)} \star K_a^{(\beta)} \star K_0.$$

where  $K_0$ ,  $K_a$  and  $K_g$  are three basic p.d. kernels (Vert et al., 2004).

# LA kernel in practice

- Implementation by dynamic programming in  $O(|x| \times |x'|)$



- In practice, **values are too large** (exponential scale) so taking its logarithm is a safer choice (but not p.d. anymore!)

## Convolution kernels



D. Haussler.

Convolution Kernels on Discrete Structures.

Technical Report UCSC-CRL-99-10, UC Santa Cruz, 1999.



C. Watkins.

Dynamic alignment kernels.

In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50. MIT Press, Cambridge, MA, 2000.



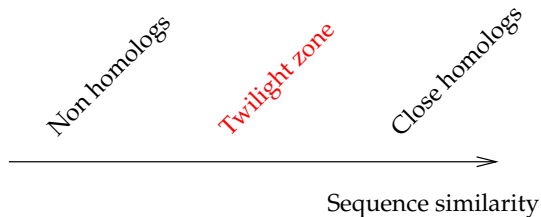
J.-P. Vert, H. Saigo, and T. Akutsu.

Local alignment kernels for biological sequences.

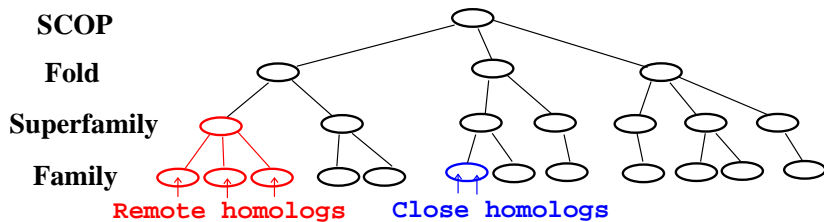
In B. Schölkopf, K. Tsuda, and J.P. Vert, editors, *Kernel Methods in Computational Biology*, pages 131–154. MIT Press, 2004.

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
  - Motivations
  - Feature space approach
  - Using generative models
  - Derive from a similarity measure
  - Application: remote homology detection
- 3 Kernels on graphs

# Remote homology



- Homologs have **common ancestors**
- Structures and functions are more conserved than sequences
- **Remote homologs** can not be detected by direct sequence comparison

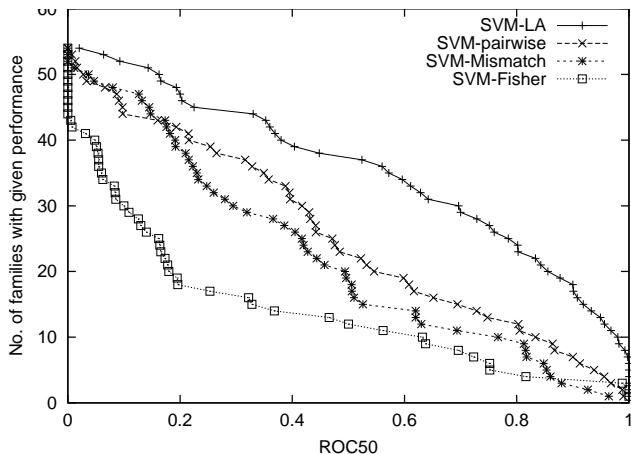




# A benchmark experiment

- **Goal:** recognize directly the superfamily
- **Training:** for a sequence of interest, positive examples come from the same superfamily, but different families. Negative from other superfamilies.
- **Test:** predict the superfamily.

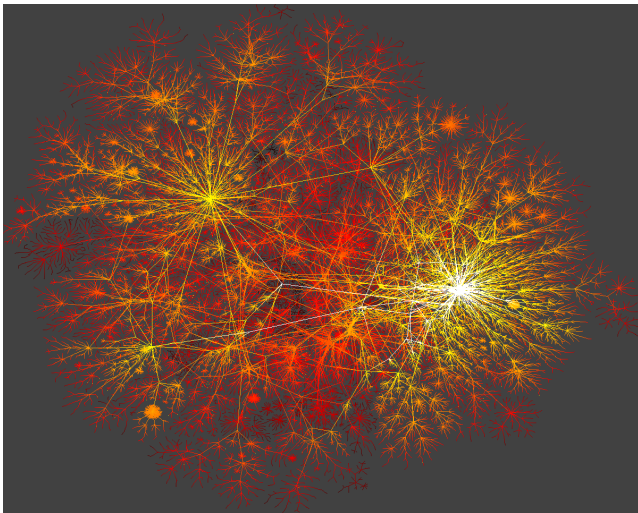
# Difference in performance



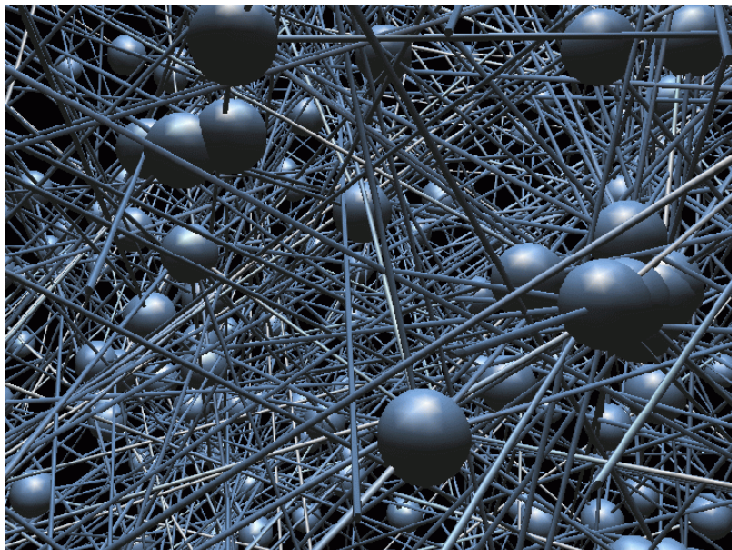
Performance on the SCOP superfamily recognition benchmark (from Vert et al., 2004).

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
- 3 Kernels on graphs
  - **Motivations**
  - Construction by regularization
  - The diffusion kernel
  - Harmonic analysis on graphs
  - Applications

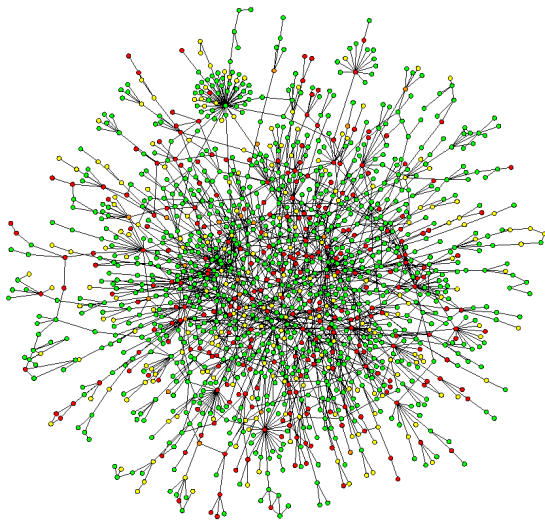
# Example: web



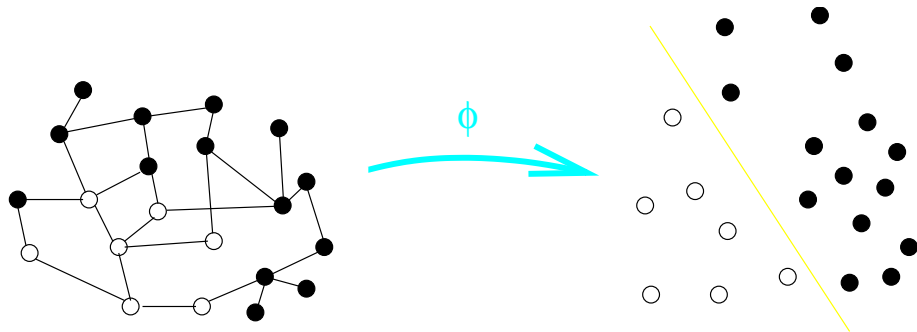
# Example: social network



# Example: protein-protein interaction



# Kernel on a graph



- We need a **kernel  $K(\mathbf{x}, \mathbf{x}')$**  between nodes of the graph.
- Example: predict gene protein functions from high-throughput protein-protein interaction data.

## Strategies to make a kernel on a graph

- $\mathcal{X}$  being finite, **any symmetric semi-definite matrix  $K$**  defines a valid p.d. kernel on  $\mathcal{X}$ .
- How to “translate” the graph topology into the kernel?
  - **Direct geometric approach:**  $K_{i,j}$  should be “large” when  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are “close” to each other on the graph?
  - **Functional approach:**  $\|f\|_K$  should be “small” when  $f$  is “smooth” on the graph?
  - **Link discrete/continuous:** is there an equivalent to the continuous Gaussian kernel on the graph (e.g., limit by fine discretization)?



## Strategies to make a kernel on a graph

- $\mathcal{X}$  being finite, **any symmetric semi-definite matrix  $K$**  defines a valid p.d. kernel on  $\mathcal{X}$ .
- How to “translate” the graph topology into the kernel?
  - **Direct geometric approach:**  $K_{i,j}$  should be “large” when  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are “close” to each other on the graph?
  - **Functional approach:**  $\|f\|_K$  should be “small” when  $f$  is “smooth” on the graph?
  - **Link discrete/continuous:** is there an equivalent to the continuous Gaussian kernel on the graph (e.g., limit by fine discretization)?

## A direct approach

- Remember : for  $\mathcal{X} = \mathbb{R}^n$ , the Gaussian RBF kernel is:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-d(\mathbf{x}, \mathbf{x}')^2 / 2\sigma^2\right),$$

where  $d(\mathbf{x}, \mathbf{x}')$  is the **Euclidean distance**.

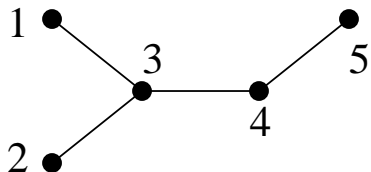
- If  $\mathcal{X}$  is a **graph**, let  $d(\mathbf{x}, \mathbf{x}')$  be the **shortest-path distance between  $\mathbf{x}$  and  $\mathbf{x}'$** .
- Problem:** the shortest-path distance is **not** a Hilbert distance (except for special graphs, e.g., trees)...

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
- 3 Kernels on graphs
  - Motivations
  - **Construction by regularization**
  - The diffusion kernel
  - Harmonic analysis on graphs
  - Applications

## Idea

- Define a priori a **smoothness functional** on the functions  $f : \mathcal{X} \rightarrow \mathbb{R}$ .
- Show that **it defines a RKHS** and identify the corresponding kernel

# Notations

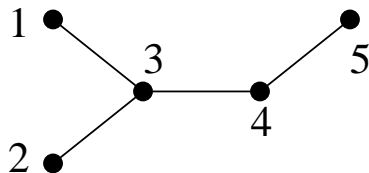


$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

# Graph Laplacian

## Definition

The Laplacian of the graph is the matrix  $L = A - D$ .



$$L = A - D = \begin{pmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 1 & 1 & -3 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

# Properties of the Laplacian

## Lemma

Let  $L = A - D$  be the Laplacian of the graph:

- For any  $f : \mathcal{X} \rightarrow \mathbb{R}$ ,

$$\Omega(f) := \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = -f^\top L f$$

- $-L$  is a **symmetric positive semi-definite** matrix
- 0 is an **eigenvalue** with multiplicity 1 associated to the constant eigenvector  $\mathbf{1} = (1, \dots, 1)$
- The **image** of  $L$  is

$$\text{Im}(L) = \left\{ f \in \mathbb{R}^m : \sum_{i=1}^m f_i = 0 \right\}$$

## Theorem

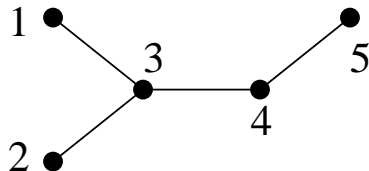
The set  $\mathcal{H} = \{f \in \mathbb{R}^m : \sum_{i=1}^m f_i = 0\}$  endowed with the norm:

$$\Omega(f) = \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2$$

is a RKHS whose reproducing kernel is  $(-L)^*$ , the pseudo-inverse of the graph Laplacian.



# Example



$$(-L)^* = \begin{pmatrix} 0.88 & -0.12 & 0.08 & -0.32 & -0.52 \\ -0.12 & 0.88 & 0.08 & -0.32 & -0.52 \\ 0.08 & 0.08 & 0.28 & -0.12 & -0.32 \\ -0.32 & -0.32 & -0.12 & 0.48 & 0.28 \\ -0.52 & -0.52 & -0.32 & 0.28 & 1.08 \end{pmatrix}$$

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
- 3 Kernels on graphs
  - Motivations
  - Construction by regularization
  - **The diffusion kernel**
  - Harmonic analysis on graphs
  - Applications

# The diffusion equation

## Lemma

For any  $\mathbf{x}_0 \in \mathbb{R}^d$ , the function:

$$K_{\mathbf{x}_0}(\mathbf{x}, t) = K_t(\mathbf{x}_0, \mathbf{x}) = \frac{1}{(4\pi t)^{\frac{d}{2}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{4t}\right).$$

is solution of the *diffusion equation*:

$$\frac{\partial}{\partial t} K_{\mathbf{x}_0}(\mathbf{x}, t) = \Delta K_{\mathbf{x}_0}(\mathbf{x}, t).$$

with initial condition  $K_{\mathbf{x}_0}(\mathbf{x}, 0) = \delta_{\mathbf{x}_0}(\mathbf{x})$ .

# Discrete diffusion equation

- For finite-dimensional  $f_t \in \mathbb{R}^m$ , the diffusion equation becomes:

$$\frac{\partial}{\partial t} f_t = L f_t$$

which admits the following solution:

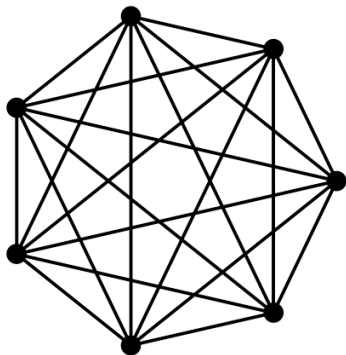
$$f_t = f_0 e^{tL}$$

- This suggest to consider:

$$K = e^{tL}$$

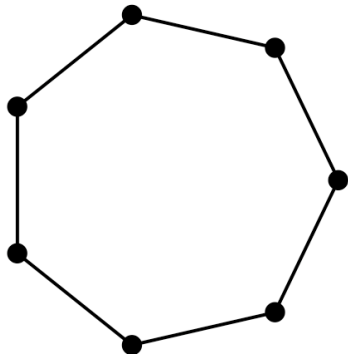
which is indeed symmetric positive semi-definite. We call it the **diffusion kernel** or **heat kernel**.

# Example: complete graph



$$K_{i,j} = \begin{cases} \frac{1+(m-1)e^{-tm}}{m} & \text{for } i = j, \\ \frac{1-e^{-tm}}{m} & \text{for } i \neq j. \end{cases}$$

## Example: closed chain



$$K_{i,j} = \frac{1}{m} \sum_{\nu=0}^{m-1} \exp \left[ -2t \left( 1 - \cos \frac{2\pi\nu}{m} \right) \right] \cos \frac{2\pi\nu(i-j)}{m}.$$

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
- 3 Kernels on graphs
  - Motivations
  - Construction by regularization
  - The diffusion kernel
  - **Harmonic analysis on graphs**
  - Applications

# Spectrum of the diffusion kernel

- Let  $0 = \lambda_1 > -\lambda_2 \geq \dots \geq -\lambda_m$  be the eigenvalues of the Laplacian:

$$L = \sum_{i=1}^m (-\lambda_i) u_i u_i^\top \quad (\lambda_i \geq 0)$$

- The diffusion kernel  $K_t$  is an **invertible** matrix because its eigenvalues are strictly positive:

$$K_t = \sum_{i=1}^m e^{-t\lambda_i} u_i u_i^\top$$



# Norm in the diffusion RKHS

- For any function  $f \in \mathbb{R}^m$ , let:

$$\hat{f}_i = u_i^\top f$$

be the Fourier coefficients of  $f$  (projection of  $f$  onto the eigenbasis of  $K$ ).

- The RKHS norm of  $f$  is then:

$$\|f\|_{K_t}^2 = f^\top K^{-1} f = \sum_{i=1}^m e^{t\lambda_i} \hat{f}_i^2.$$

This observation suggests to define a whole family of kernels:

$$K_r = \sum_{i=1}^m r(\lambda_i) u_i u_i^\top$$

associated with the following RKHS norms:

$$\|f\|_{K_r}^2 = \sum_{i=1}^m \frac{\hat{f}_i^2}{r(\lambda_i)}$$

where  $r : \mathbb{R}^+ \rightarrow \mathbb{R}_*^+$  is a **non-increasing** function.

## Example : regularized Laplacian

$$r(\lambda) = \frac{1}{\lambda + \epsilon}, \quad \epsilon > 0$$

$$K = \sum_{i=1}^m \frac{1}{\lambda_i + \epsilon} u_i u_i^\top = (-L + \epsilon I)^{-1}$$

$$\|f\|_K^2 = f^\top K^{-1} f = \sum_{i \sim j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 + \epsilon \sum_{i=1}^m f(\mathbf{x}_i)^2.$$

- 1 Kernels and kernel methods
- 2 Kernels for biological sequences
- 3 Kernels on graphs
  - Motivations
  - Construction by regularization
  - The diffusion kernel
  - Harmonic analysis on graphs
  - Applications

# Applications 1: graph partitioning

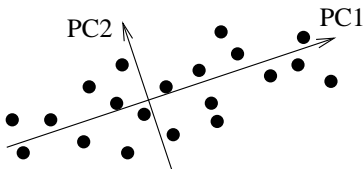
- A classical relaxation of graph partitioning is:

$$\min_{f \in \mathbb{R}^{\mathcal{X}}} \sum_{i \sim j} (f_i - f_j)^2 \quad \text{s.t.} \quad \sum_i f_i^2 = 1$$

- This can be rewritten

$$\max_f \sum_i f_i^2 \quad \text{s.t.} \quad \|f\|_{\mathcal{H}} \leq 1$$

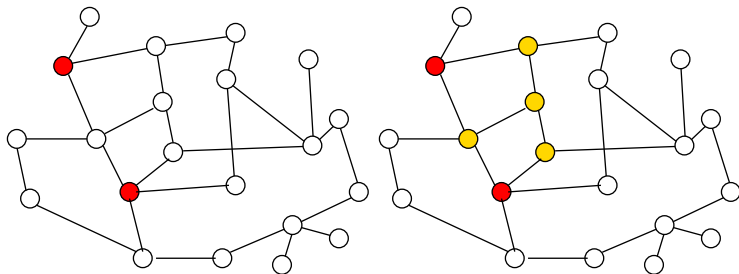
- This is **principal component analysis** in the RKHS (“kernel PCA”)



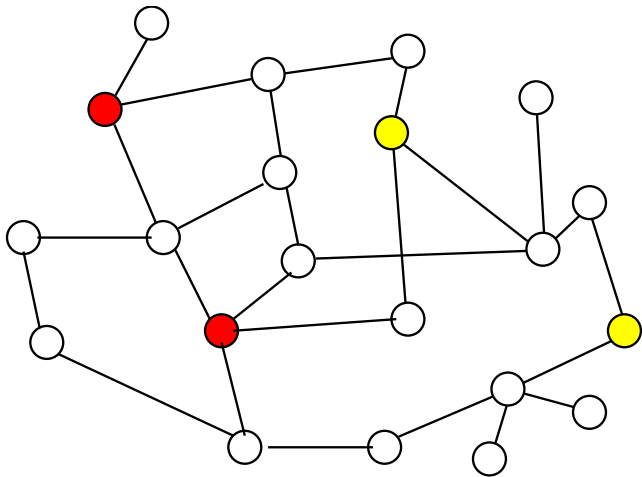
## Applications 2: search on a graph

- Let  $x_1, \dots, x_q$  a set of  $q$  nodes (the **query**). How to find “similar” nodes (and rank them)?
- One solution:

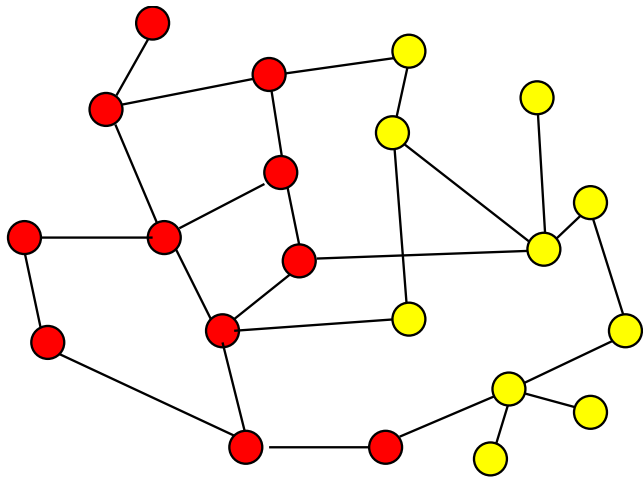
$$\min_f \|f\|_{\mathcal{H}} \quad \text{s.t.} \quad f(x_i) \geq 1 \text{ for } i = 1, \dots, q.$$



## Application 3: Semi-supervised learning



# Application 3: Semi-supervised learning





# Application 4: Tumor classification from microarray data

## Data available

- Gene expression measures for **more than 10k genes**
- Measured on **less than 100 samples** of two (or more) different classes (e.g., different tumors)

## Goal

- Design a **classifier** to automatically assign a class to future samples from their expression profile
- **Interpret** biologically the differences between the classes

# Application 4: Tumor classification from microarray data

## Data available

- Gene expression measures for **more than 10k genes**
- Measured on **less than 100 samples** of two (or more) different classes (e.g., different tumors)

## Goal

- Design a **classifier** to automatically assign a class to future samples from their expression profile
- **Interpret** biologically the differences between the classes

## The approach

- Each sample is represented by a vector  $x = (x_1, \dots, x_p)$  where  $p > 10^5$  is the number of probes
- **Classification**: given the set of labeled sample, learn a linear decision function:

$$f(x) = \sum_{i=1}^p \beta_i x_i + \beta_0 ,$$

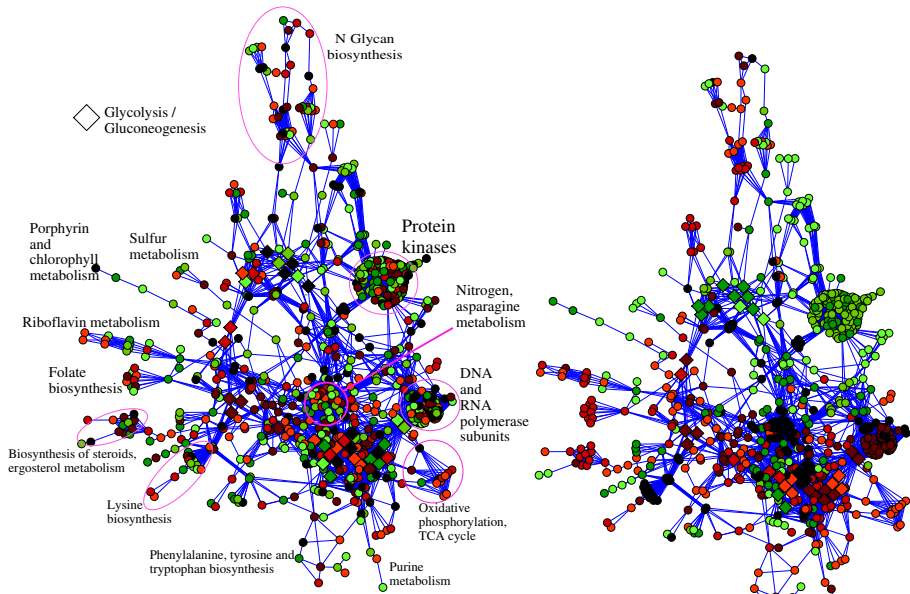
- **Interpretation**: the weight  $\beta_i$  quantifies the influence of gene  $i$  for the classification

## Pitfalls

- **No robust estimation procedure** exist for 100 samples in  $10^5$  dimensions!

- We know the functions of many genes, and how they interact together.
- This can be represented as a **graph of genes**, where connected genes perform some action together
- Prior knowledge: **constraint the weights of genes that work together to be similar**
- Mathematically: constrain the norm of the weight vector in the RKHS of the diffusion kernel.

# Comparison



## Kernels on graphs



R. I. Kondor and J. Lafferty.

Diffusion Kernels on Graphs and Other Discrete Input.

In *ICML 2002*, 2002.

## Applications



F. Rapaport, A. Zinovyev, M. Dutreix, E. Barillot and J.-P. Vert

Classification of Microarray Data using Gene Networks.

*BMC Bioinformatics*, 8:35, 2007.

# Conclusion

## Kernel design

- A variety of principles for string and graph kernel design have been proposed.
- Good **kernel design** is **important** for each data and each task. Performance is not the only criterion.
- Still an **art**, although principled ways have started to emerge.
- The **integration of “higher-order information”** is a hot topic! Kernel methods are promising to **combine generative and discriminative approaches**.
- Their application goes of course **beyond computational biology**.
- Their application goes of course **beyond strings and graphs**.



## Challenges

- How to **choose “the” best kernel** for a given task, or to **learn simultaneously** with different kernels?
- How to extend the methods to **non p.d.** and **non symmetric** kernels?
- How to design **scalable kernel methods** to process millions of points?