Kernels and Kernel Methods for Biological Sequences

Jean-Philippe Vert
`Jean-Philippe.Vert@ensmp.fr`

Centre for Computational Biology
Ecole des Mines de Paris, ParisTech

Current Challenges in Kernel Methods workshop (CCKM'06),
Bruxelles, Belgium, November 27th, 2006

Outline

Outline

Part 1

Kernels and Kernels Methods

Overview

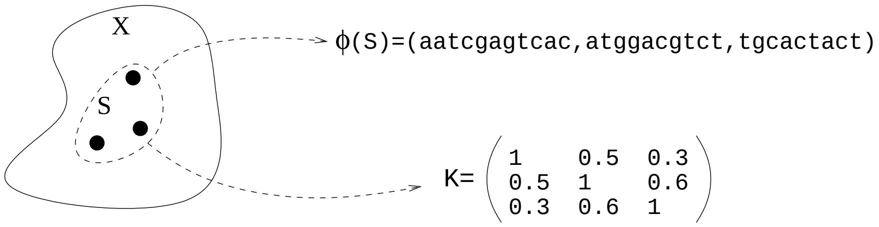
Motivations

- 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



$$K=\begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{pmatrix}$$

Idea

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

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

Positive Definite (p.d.) Kernels

Definition

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

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

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

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

General remarks

Remark

- Equivalently, a kernel K is p.d. if and only if, for any $N \in \mathbb{N}$ and any set of points $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \in \mathcal{X}^N$, the similarity matrix $[K]_{ij} := K\left(\mathbf{x}_i, \mathbf{x}_j\right)$ 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?)

Examples

Kernels for vectors

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

- The linear kernel

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

- The polynomial kernel

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

- The Gaussian RBF kernel:

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

Geometric interpretation: Kernels are inner products

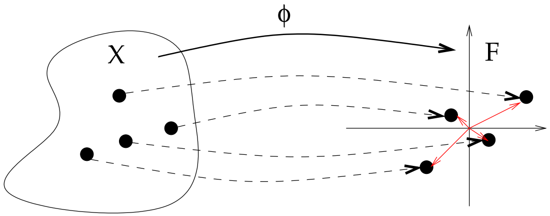
Theorem (Aronszajn, 1950)

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 $\mathcal{X}$:

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



Corollary: The kernel trick

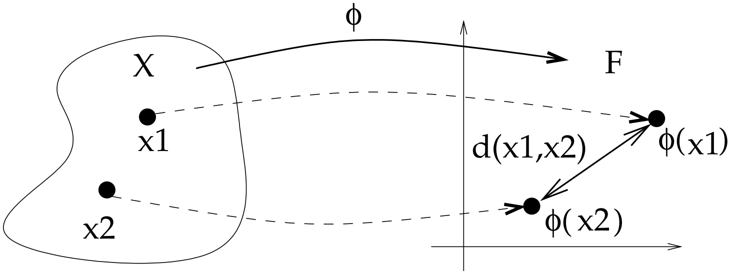
Kernel trick

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



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

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

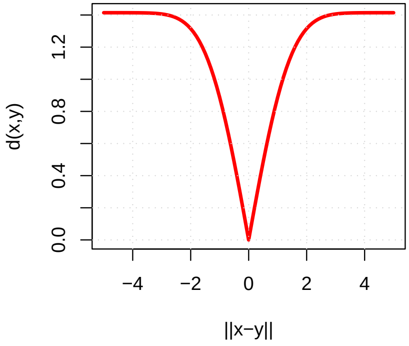
Distance for the Gaussian kernel

- The Gaussian kernel with bandwidth σ on $\mathbb{R}^d$ is:

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

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

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



Functional interpretation: RKHS

RKHS definition

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

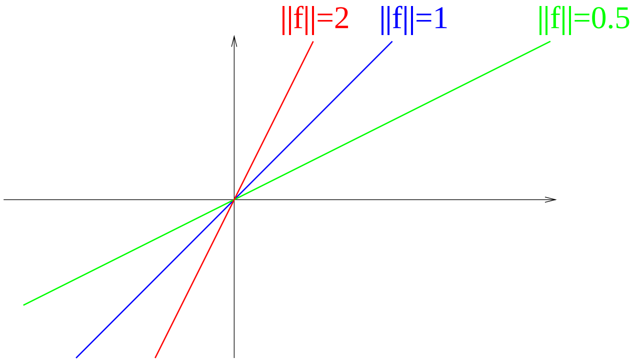
$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \ ,$$

with norm

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

Example: Linear kernel

$$\begin{cases} K_{lin}\left(\mathbf{x},\mathbf{x}'\right) & = \mathbf{x}^\top \mathbf{x}' \ . \\ f\left(\mathbf{x}\right) & = w^\top x \ , \\ \parallel f \parallel_{\mathcal{H}} & = \parallel w \parallel_2 \ . \end{cases}$$



Examples: Gaussian RBF kernel

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

Smoothness functional

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:

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

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

An important message

The RKHS norm is therefore a smoothness functional:

$$\text{Small norm} \implies \text{slow variations.}$$

Learning from data

General setting

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

Learning from data

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.

Learning with kernels

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.

Example: Large-margin classifiers

General setting

- For pattern recognition $\mathcal{Y} = \{-1, 1\}$.
- Goal: estimate a function $f : \mathcal{X} \to \mathbb{R}$ 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 $\phi : \mathbb{R} \to \mathbb{R}$ is non-increasing.

ERM in for large-margin classifiers: Theory

Theoretical results

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

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

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

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

ERM in for large-margin classifiers: Practise

Reformulation as penalized minimization

- We must solve the constrained minimization problem:

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

- To make this practical we assume that ϕ 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 λ the unconstrained problem:

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

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:

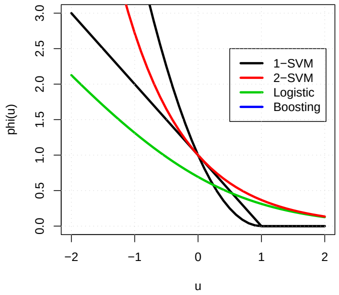
$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \,.$$

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

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

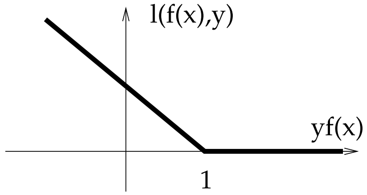
- This can be implemented using general packages for convex optimization or specific algorithms (e.g., for SVM).

Loss function examples



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

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

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 $\boldsymbol{\alpha}$:

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

subject to:

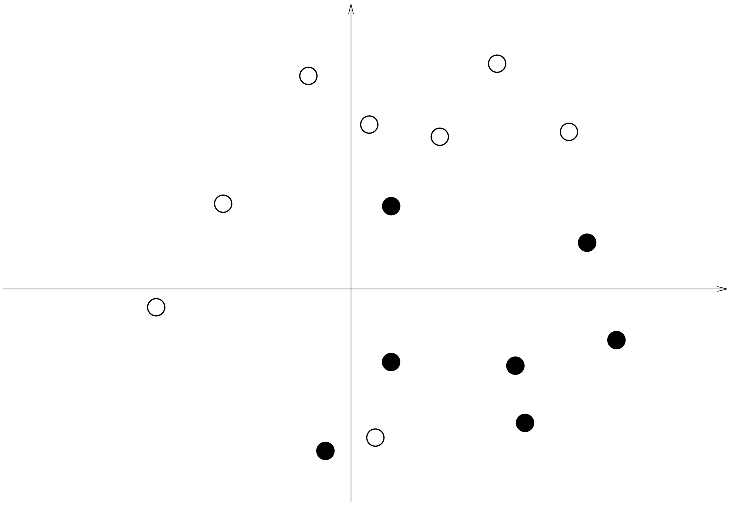
$$\begin{cases} y_i \sum_{j=1}^{n} \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \xi_i - 1 \geq 0 \,, & \text{for } i = 1, \ldots, n \,, \\ \xi_i \geq 0 \,, & \text{for } i = 1, \ldots, n \,. \end{cases}$$

Solving the problem

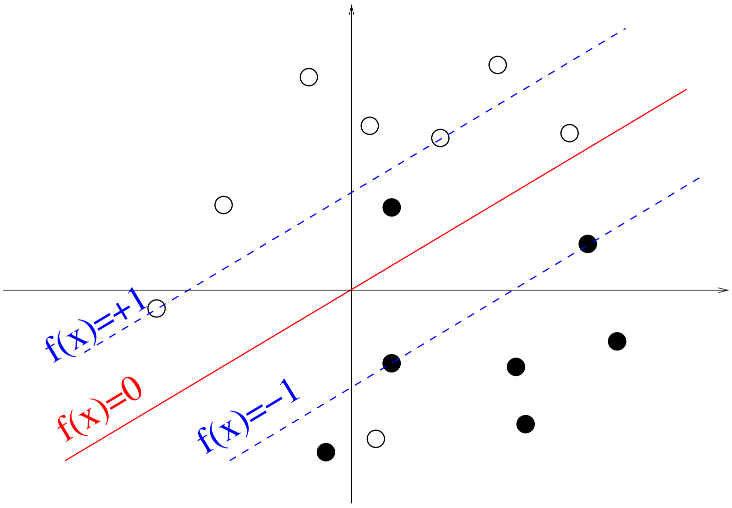
Remarks

- 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).

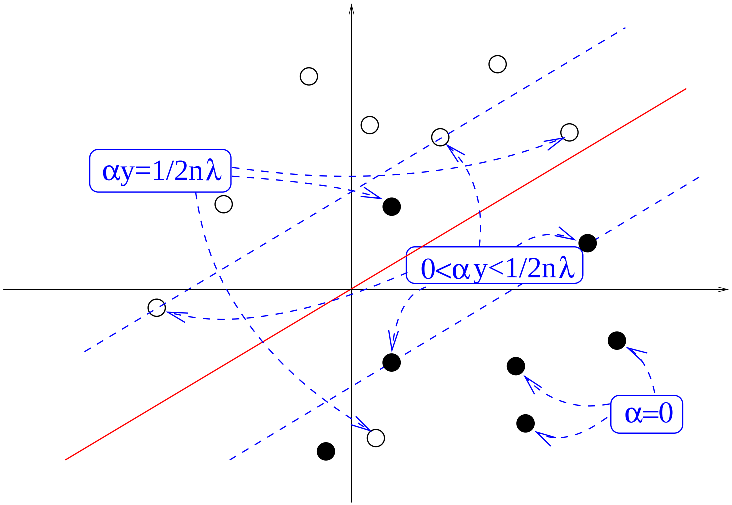
Geometric interpretation



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).

Further reading

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.

Further reading

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.

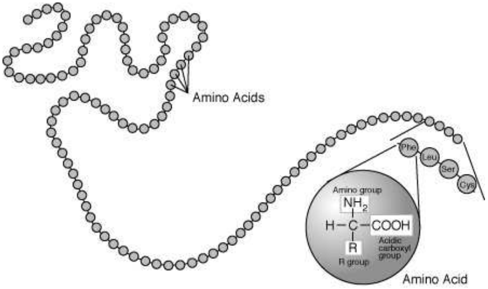
J. Shawe-Taylor and N. Cristianini.
Kernel Methods for Pattern Analysis.
Cambridge University Press, 2004.

Part 2

Kernels for biological sequences

Outline

Proteins



A : Alanine	V : Valine	L : Leucine
F : Phenylalanine	P : Proline	M : Méthionine
E : Acide glutamique	K : Lysine	R : Arginine
T : Threonine	C : Cysteine	N : Asparagine
H : Histidine	V : Thyrosine	W : Tryptophane
I : Isoleucine	S : Sérine	Q : Glutamine
D : Acide aspartique	G : Glycine	

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)

- Secured proteins:

```
MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNIEA...
MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...
MALHTVLIMLSLLPMLEAQNPEHANITIGEPITNETLGWL...
...
```

- Non-secreted proteins:

```
MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVNLGVG...
MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...
MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..
...
```

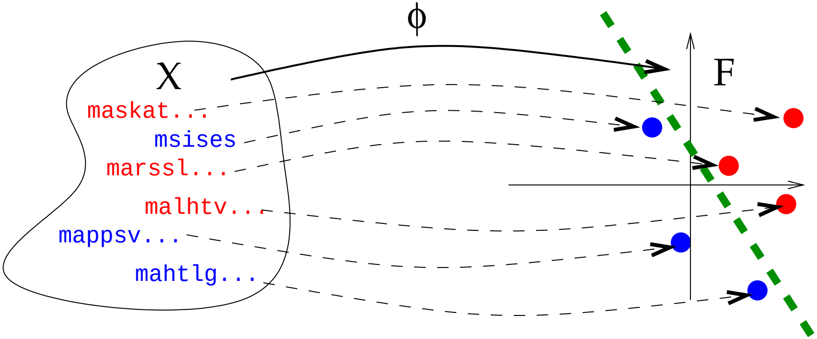
Goal

- 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 $\Phi(x_1), \ldots, \Phi(x_n)$ of the training set (nearest neighbor, linear perceptron, logistic regression, support vector machine...)



Kernels for protein sequences

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 for protein sequences

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 for protein sequences

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 for protein sequences

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

Outline

Vector embedding for strings

The idea

Represent each sequence **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)

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

Substring indexation

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

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

is:

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

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

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

- The length of the substring is:

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

Example 2: Substring kernel (2/5)

Example

ABRACADABRA

- $\mathbf{i} = (3, 4, 7, 8, 10)$
- $\mathbf{x}(\mathbf{i}) =$ 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 $\Phi_{\mathbf{u}} : \mathcal{X} \to \mathbb{R}$ be defined by:

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

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

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

Example 2: Substring kernel (4/5)

Example

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

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

Example 2: Substring kernel (5/5)

Kernel computation

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

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

- 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 $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$
- 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 $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$
- 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.

Further reading

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.

Further reading

Dictionary-based string kernels

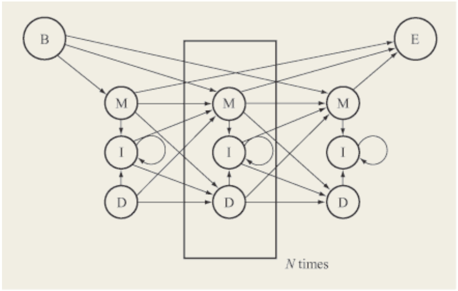
B. Logan, P. Moreno, B. Suzek, Z. Weng, and S. Kasif.
A Study of Remote Homology Detection.
Technical Report CRL 2001/05, Compaq Cambridge Research laboratory, June 2001.

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.

Outline

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):

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

where $I(\theta_0) = E_{\theta_0}\left[\Phi_{\theta_0}(\mathbf{x})\Phi_{\theta_0}(\mathbf{x})^\top\right]$ 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).

Fisher kernel in practice

- $\Phi_{\theta_0}(\mathbf{x})$ 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 θ_0) can be trained and combined
- Feature vectors are explicitly computed

Further reading

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 Θ
- 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
- $K(\mathbf{x}, \mathbf{x}') = < \phi(\mathbf{x}), \phi(\mathbf{x}') >_{L_2(w)}$ 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:

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

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

Context-tree model

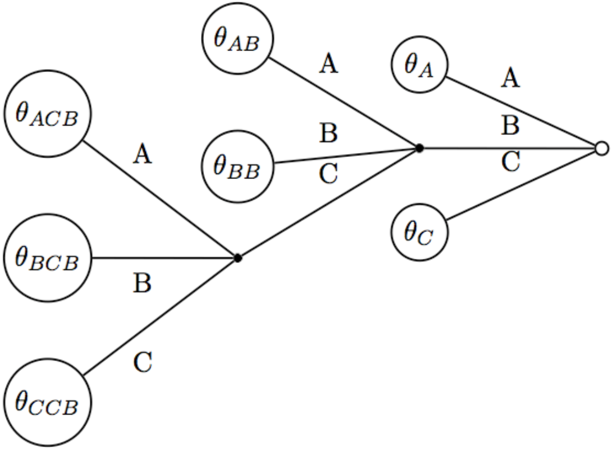
Definition

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

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

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

Context-tree model: example



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

The context-tree kernel

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.*

Further reading

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):

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

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:

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

- Marginalizing therefore gives:

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

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

Example: HMM for normal/biased coin toss

$$\begin{tikzcd}
 & N \arrow[loop above, "0.85"] \arrow[dd, bend left, "0.1"] \arrow[dr, bend left, "0.05"] & \\
S \arrow[ur, bend left, "0.5"] \arrow[dr, bend right, "0.5"'] & & E \\
 & B \arrow[loop below, "0.85"] \arrow[uu, bend left, "0.1"] \arrow[ur, bend right, "0.05"'] &
\end{tikzcd}$$

- 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):

```
NNNNNBBBBBBBBBNNNNNNNNNNNBBBBBB
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 $\mathbf{z} = (\mathbf{x}, \mathbf{y})$:

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

  where $n_{a,s}\left(\mathbf{x}, \mathbf{y}\right)$ 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:

$$\mathbf{z} = 1001011101111010010111001111011,$$
$$\mathbf{z}' = 00110101100111110110101111101100101,$$

$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = n_0\left(\mathbf{z}\right) n_0\left(\mathbf{z}'\right) + n_0\left(\mathbf{z}\right) n_0\left(\mathbf{z}'\right) + n_1\left(\mathbf{z}\right) n_1\left(\mathbf{z}'\right) + n_1\left(\mathbf{z}\right) n_1\left(\mathbf{z}\right.$$
$$= 7 \times 15 + 9 \times 12 + 13 \times 6 + 2 \times 1 = 293.$$

1-spectrum marginalized kernel on observed data

- The marginalized kernel for observed data is:

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

with

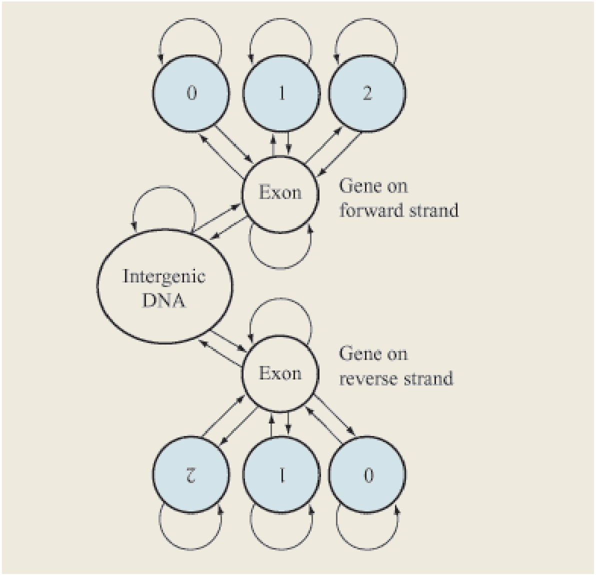
$$\Phi_{a,s}\left(\mathbf{x}\right) = \sum_{\mathbf{y}\in\mathcal{S}^*} P\left(\mathbf{y}|\mathbf{x}\right) n_{a,s}\left(\mathbf{x},\mathbf{y}\right)$$

Computation of the 1-spectrum marginalized kernel

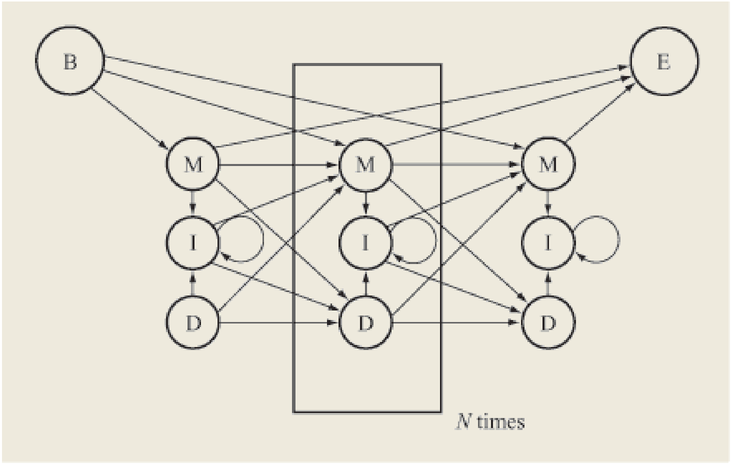
$$
\begin{aligned}
\Phi_{a,s}(\mathbf{x}) &= \sum_{\mathbf{y}\in\mathcal{S}^*} P(\mathbf{y}|\mathbf{x})\, n_{a,s}(\mathbf{x},\mathbf{y}) \\
&= \sum_{\mathbf{y}\in\mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) \left\{ \sum_{i=1}^{n} \delta(x_i, a)\, \delta(y_i, s) \right\} \\
&= \sum_{i=1}^{n} \delta(x_i, a)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

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

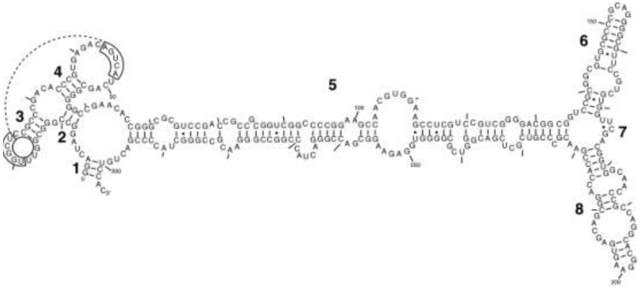
HMM example (DNA)



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)

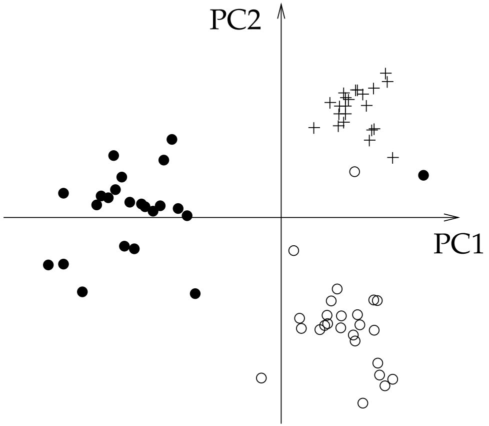
- Feature: number of occurrences of each (base,state) combination
- Marginalization using classical inside/outside algorithm

Marginalized kernels in practice

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).

Further reading

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.

Outline

Sequence alignment

Motivation

How to compare 2 sequences?

$$\mathbf{x}_1 = \texttt{CGGSLIAMMWFGV}$$
$$\mathbf{x}_2 = \texttt{CLIVMMNRLMWFGV}$$

Find a good alignment:

```
CGGSLIAMM----WFGV
|...|||||....||||
C---LIVMMNRLMWFGV
```

Alignment score

In order to quantify the relevance of an alignment π, define:

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

Any alignment is then scored as follows

```
CGGSLIAMM----WFGV
|...|||||....||||
C---LIVMMNRLMWFGV
```

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

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:

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

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:

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

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:

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

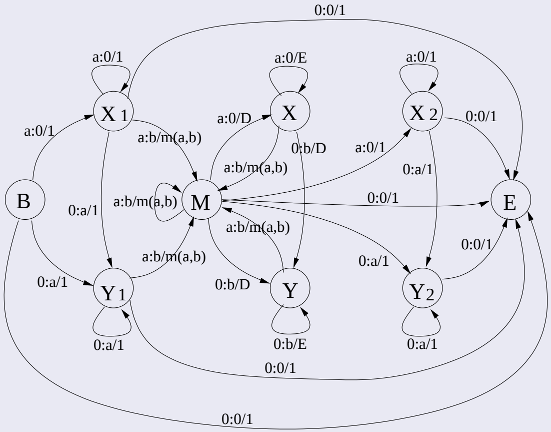
  is also p.d. (Haussler, 1999).
- LA kernel is p.d. because it is a convolution kernel (Haussler, 1999):

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

  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 $O(|\mathbf{x}| \times |\mathbf{x}'|)$



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

Further reading

Convolution kernels

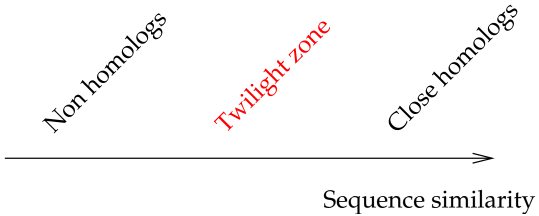
D. Haussler.
Convolution Kernels on Discrete Structures.
Technical Report UCSC-CRL-99-10, UC Santa Cruz, 1999.

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.

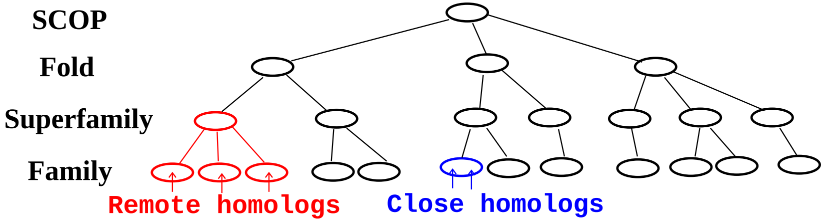
Outline

Remote homology



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

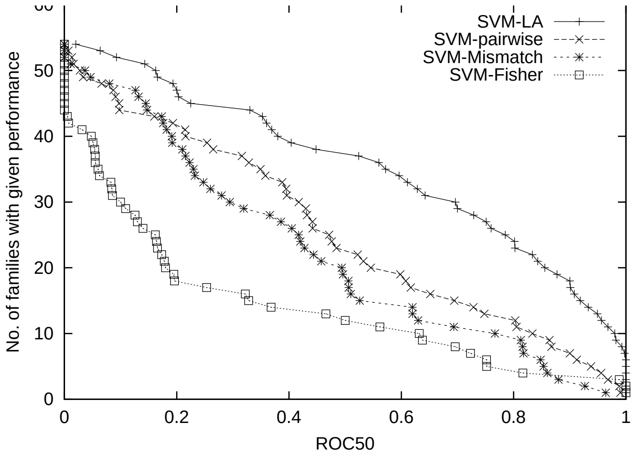
SCOP database



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).

Conclusion

Conclusion (1/2)

Kernel design

- A variety of principles for string 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.

Conclusion (2/2)

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?