Probabilistic kernels for structured objects

Jean-Philippe.Vert@mines.org

Ecole des Mines de Paris
Computational Biology group

UC Davis, May 6, 2003.

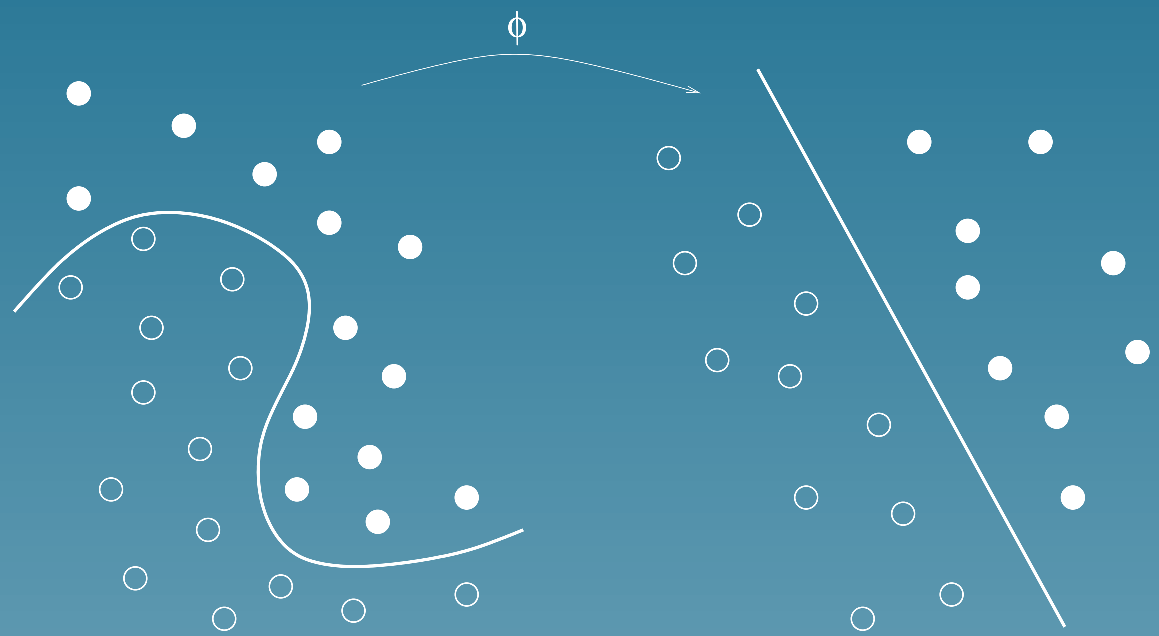
Outline

1. SVM and kernel methods
2. Making a kernel from a graphical model
3. Application: gene function prediction from phylogenetic profiles

Part 1

SVM and kernel methods

Support vector machines



- Objects to classified x mapped to a feature space
- Largest margin separating hyperplan in the feature space

The kernel trick

- Implicit definition of $x \longrightarrow \Phi(x)$ through the kernel:

$$K(x,y) \stackrel{def}{=} < \Phi(x), \Phi(y) >$$

The kernel trick

- Implicit definition of $x \longrightarrow \Phi(x)$ through the kernel:

$$K(x, y) \stackrel{def}{=} < \Phi(x), \Phi(y) >$$

- Simple kernels can represent complex Φ

The kernel trick

- Implicit definition of $x \rightarrow \Phi(x)$ through the kernel:

$$K(x,y) \stackrel{def}{=} < \Phi(x), \Phi(y) >$$

- Simple kernels can represent complex Φ
- For a given kernel, not only SVM but also clustering, PCA, ICA... possible in the feature space = kernel methods

Kernel examples

- “Classical” kernels: polynomial, Gaussian, sigmoid...
  but the objects x must be vectors

Kernel examples

- "Classical" kernels: polynomial, Gaussian, sigmoid... but the objects x must be vectors
- "Exotic" kernels for strings:

Kernel examples

- "Classical" kernels: polynomial, Gaussian, sigmoid...
  but the objects x must be vectors
- "Exotic" kernels for strings:
  - ⋆ Fisher kernel (Jaakkoola and Haussler 98)

Kernel examples

- "Classical" kernels: polynomial, Gaussian, sigmoid...
  but the objects x must be vectors

- "Exotic" kernels for strings:
  - ⋆ Fisher kernel (Jaakkoola and Haussler 98)
  - ⋆ Convolution kernels (Haussler 99, Watkins 99)

Kernel examples

- "Classical" kernels: polynomial, Gaussian, sigmoid...
  but the objects x must be vectors
- "Exotic" kernels for strings:
  - ⋆ Fisher kernel (Jaakkoola and Haussler 98)
  - ⋆ Convolution kernels (Haussler 99, Watkins 99)
  - ⋆ String kernel (Lodhi et al. 00)

Kernel examples

- "Classical" kernels: polynomial, Gaussian, sigmoid...
  but the objects x must be vectors
- "Exotic" kernels for strings:
  - ⋆ Fisher kernel (Jaakkoola and Haussler 98)
  - ⋆ Convolution kernels (Haussler 99, Watkins 99)
  - ⋆ String kernel (Lodhi et al. 00)
  - ⋆ Spectrum, mismatch kernels (Leslie et al.), rational kernels (Cortes et al.)...

Kernel engineering

- A fonction $K : \mathcal{X}^2 \rightarrow \mathbb{R}$ is a valid kernel on a set $\mathcal{X}$ if it is:
  - ⋆ symmetric : $K(x,y) = K(y,x)$,
  - ⋆ positive semi-definite: $\sum_{i,j} a_i a_j K(x_i, x_j) \geq 0$ for all $a_i \in \mathbb{R}$ and $x_i \in \mathcal{X}$
- Kernel engineering:Use prior knowledge to build the geometry of the feature space through $K(.,.)$

Part 2

Making a kernel from a graphical model

A general problem

- $\mathcal{X}$ a (finite) set

A general problem

- $\mathcal{X}$ a (finite) set
- $p(x)$ a probability distribution on $\mathcal{X}$

A general problem

- $\mathcal{X}$ a (finite) set
- $p(x)$ a probability distribution on $\mathcal{X}$
- How to build $K(x, y)$ from $p(x)$?

A general problem

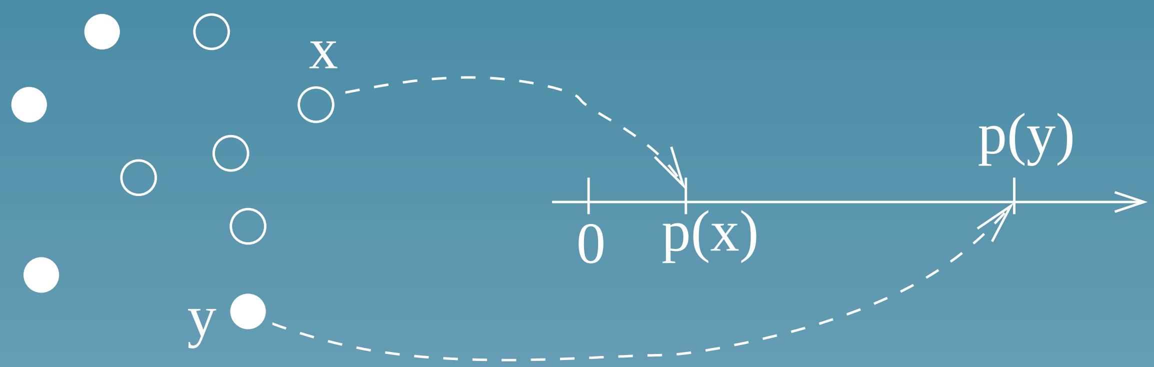
- $\mathcal{X}$ a (finite) set
- $p(x)$ a probability distribution on $\mathcal{X}$
- How to build $K(x, y)$ from $p(x)$?
- *Remark: up to translation and scaling, we can restrict* K *to be a probability on* $\mathcal{X} \times \mathcal{X}$ *(P-kernel)*

Product kernel

$$K_{prod}(x, y) = p(x)p(y)$$

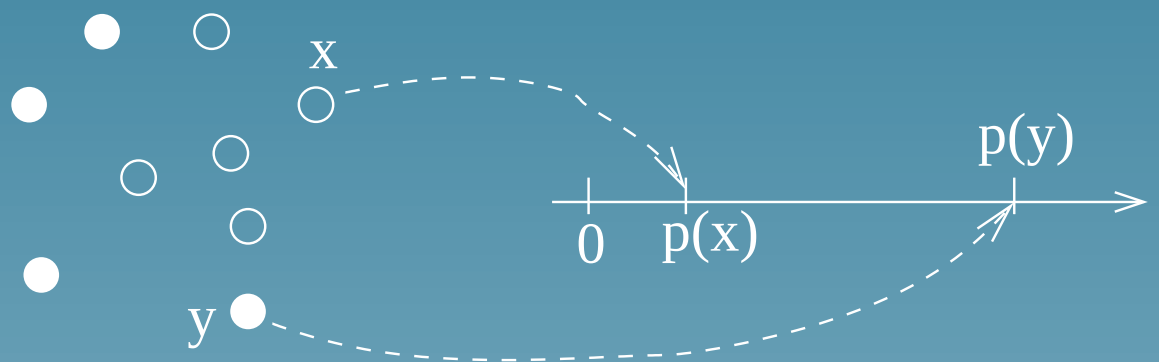
Product kernel

$$K_{prod}(x, y) = p(x)p(y)$$



Product kernel

$$K_{prod}(x, y) = p(x)p(y)$$



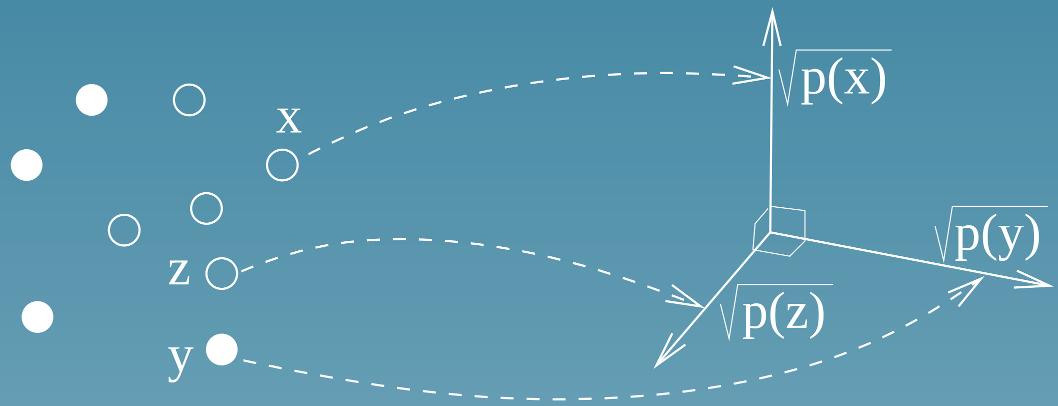
SVM = probability threshold classifier

Diagonal kernel

$$K_{diag}(x, y) = p(x)\delta(x, y)$$

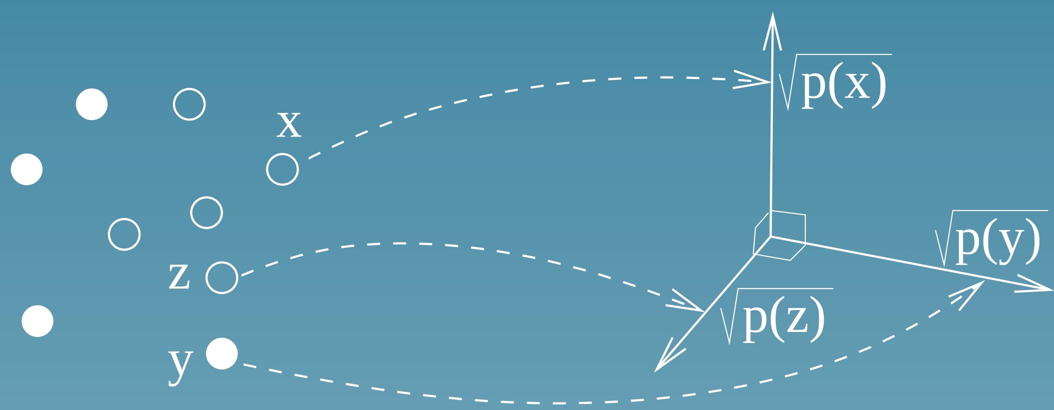
Diagonal kernel

$$K_{diag}(x, y) = p(x)\delta(x, y)$$



Diagonal kernel

$$K_{diag}(x, y) = p(x)\delta(x, y)$$



No learning

Interpolated kernel

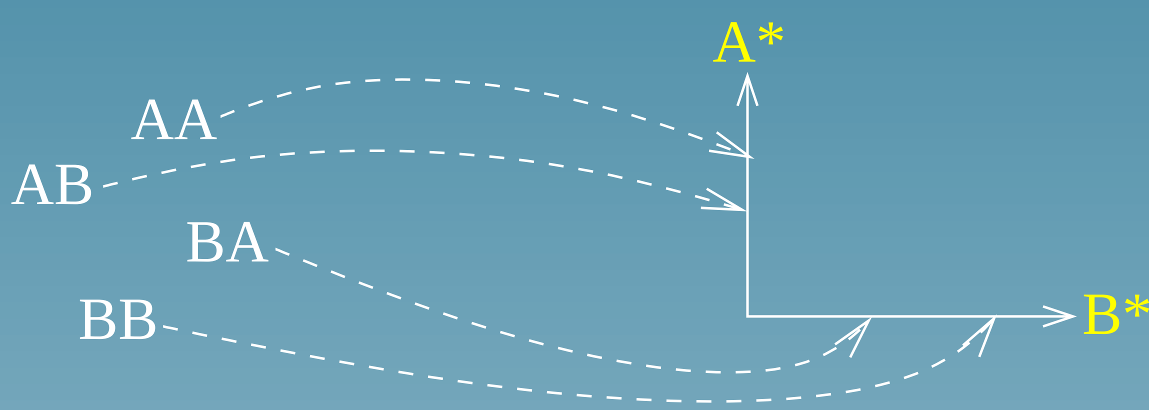
If objects are composite: $x = (x_1, x_2)$:

$$K(x, y) = K_{diag}(x_1, y_1) K_{prod}(x_2, y_2)$$

Interpolated kernel

If objects are composite: $x = (x_1, x_2)$:

$$K(x,y) = K_{diag}(x_1,y_1)K_{prod}(x_2,y_2)$$
$$= p(x_1)\delta(x_1,y_1) \times p(x_2|x_1)p(y_2|y_1)$$



General interpolated kernel

- Composite objects $x = (x_1, \ldots, x_n)$

General interpolated kernel

- Composite objects $x = (x_1, \ldots, x_n)$
- A list of index subsets: $\mathcal{V} = \{I_1, \ldots, I_v\}$ where $I_i \subset \{1, \ldots, n\}$ for $i = 1, \ldots, v$.

General interpolated kernel

- Composite objects $x = (x_1, \ldots, x_n)$
- A list of index subsets: $\mathcal{V} = \{I_1, \ldots, I_v\}$ where $I_i \subset \{1, \ldots, n\}$ for $i = 1, \ldots, v$.
- Interpolated kernel:

$$K_{\mathcal{V}}(x, y) = \frac{1}{|\mathcal{V}|} \sum_{I \in \mathcal{V}} K_{diag}(x_I, y_I) K_{prod}(x_{I^c}, y_{I^c})$$

Examples

- If $\mathcal{V} = \{\emptyset\}$, then:

$$K_{\mathcal{V}}(x, y) = K_{prod}(x, y).$$

- If $\mathcal{V} = \{[1, n]\}$, then:

$$K_{\mathcal{V}}(x, y) = K_{diag}(x, y).$$

Rare common subparts

For a given $p(x)$ and $p(y)$, we have:

$$K_{\mathcal{V}}(x, y) = K_{prod}(x, y) \times \frac{1}{|\mathcal{V}|} \sum_{I \in \mathcal{V}} \frac{\delta(x_I, y_I)}{p(x_I)}$$

Rare common subparts

For a given $p(x)$ and $p(y)$, we have:

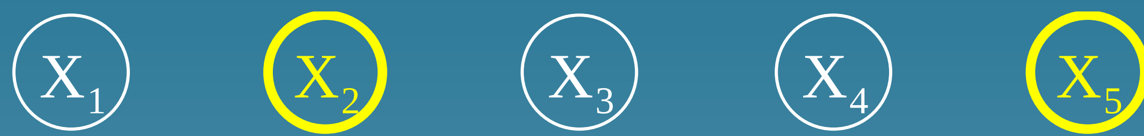
$$K_{\mathcal{V}}(x,y) = K_{prod}(x,y) \times \frac{1}{|\mathcal{V}|} \sum_{I \in \mathcal{V}} \frac{\delta(x_I, y_I)}{p(x_I)}$$

x and y get closer in the feature space when they share rare common subparts

Implementation

- For many applications, computation time of the kernel is a limiting factor
- The sum in the interpolated might involve up to 2^n terms...
- Good news: factorization possible for particular choices of $p(.)$ and $\mathcal{V}$ (in particular graphical models)

Example 1: Weight matrix kernel



Independent variables, all subsets:

$$p(x) = \prod_{i=1}^{n} p_i(x_i)$$

$$\mathcal{V} = \mathcal{P}([1, n])$$

Weight matrix kernel: Computation

$$K_{\mathcal{V}}(x, y) = \frac{1}{2^n} \prod_{i=1}^{n} \phi_i(x_i, y_i),$$

with:

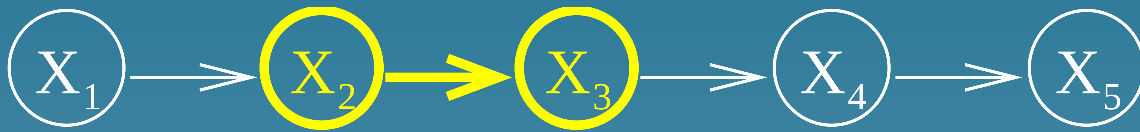
$$\phi_i(x_i, y_i) = \begin{cases} p_i(x_i) + p_i(x_i)^2 & \text{if } x_i = y_i \\ p_i(x_i)p_i(y_i) & \text{if } x_i \neq y_i \end{cases}$$

Weight matrix kernel: Proof

$$K(x,y) = \frac{1}{2^n} \sum_{\mathcal{V} \subset [1,n]} \left[\prod_{i \in \mathcal{V}} p(x_i)\delta(x_i, y_i) \times \prod_{i \notin \mathcal{V}} p(x_i)p(y_i) \right]$$

$$= \frac{1}{2^n} \prod_{i=1}^{n} \left[p(x_i)\delta(x_i, y_i) + p(x_i)p(y_i) \right].$$

Example 2: Markov block kernel



Markov model, all blocks:

$$p(x) = p_1(x_1) \prod_{i=2}^{n} p_i(x_i | x_{i-1})$$

$$\mathcal{V} = \{[k, l] : 1 \leq k \leq l \leq n\} \cup \{\emptyset\}$$

Markov block kernel: computation

$$K_{\mathcal{V}}(x, y) = \phi_0(n) + \phi_1(n) + \phi_2(n),$$

with:

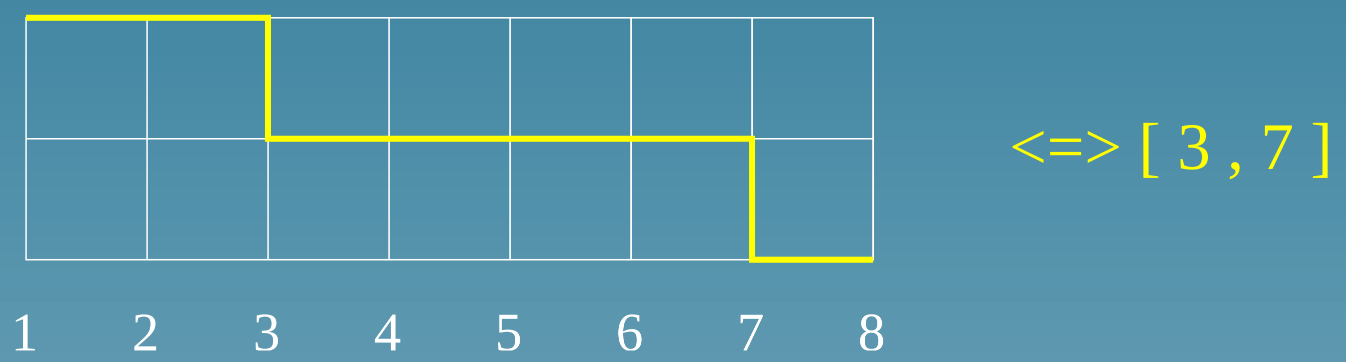
$$\begin{cases} \phi_0(1) = p_1(x_1)p_1(y_1) \\ \phi_1(1) = p_1(x_1)\delta(x_1, y_1) \\ \phi_2(1) = 0 \end{cases}$$

and for $i = 2, \ldots, n$:

$$\begin{cases} \phi_0(i) = p_i(x_i|x_{i-1})p_i(y_i|y_{i-1}) \times \phi_0(i-1) \\ \phi_1(i) = p_i(x_i|x_{i-1})\delta(x_i, y_i) \\ \qquad\qquad \times \left[\phi_1(i-1) + \frac{p_i(y_i|y_{i-1})}{p_i(x_i)}\phi_0(i-1)\right] \\ \phi_2(i) = p_i(x_i|x_{i-1})p_i(y_i|y_{i-1}) \times [\phi_1(i-1) + \phi_2(i-1)] \end{cases}$$

Markov kernel: Proof

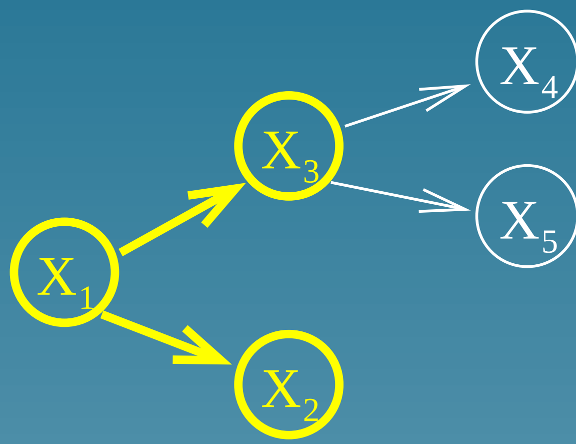
- Bijection between the set of intervals and the set of paths



- Factorization along each path

- Classical dynamic programming for the summation

Example 3: common subtree kernel



Bayesian tree model, all rooted subtrees:

$$p(x) = p_\lambda(x_\lambda) \prod_{s \in T \setminus \{\lambda\}} p_s(x_s | x_{f(s)})$$

$$\mathcal{V} = \{S \text{ rooted subtree of T}\}$$

Common subtree kernel: computation

$$K(x,y) = \sum_{S \in \mathcal{V}} \Big[\prod_{s \in S} p(x_s | x_{f(s)}) \delta(x_s, y_s) \\ \times \prod_{s \notin S} p(x_s | x_{f(s)} p(y_s | y_{f(s)}) \Big]$$

Can be computed in linear time by one post-order traversal of the tree (similar to the CTW algorithm by Willems et al.)

Common subtree kernel: proof

$$K(x,y) = \sum_{S \in \mathcal{V}} \left[\prod_{s \in S} f(s) \times \prod_{s \notin S} g(s) \right] = \alpha(\lambda) + \beta(\lambda),$$

where:

$$\beta(s) = \begin{cases} g(s) & \text{if } s \text{ is a leaf} \\ g(s) \prod_{s' < s} \beta(s') & \text{otherwise ;} \end{cases}$$

$$\alpha(s) = \begin{cases} f(s) & \text{if } s \text{ is a leaf} \\ f(s) \left(\prod_{s' < s} \beta(s') + \prod_{s' < s} \alpha(s') \right) & \text{otherwise .} \end{cases}$$

Example 4: common subtree kernel with latent variables

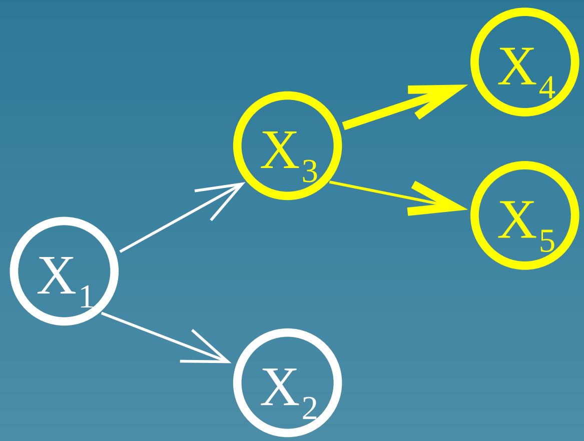
- Same as example 3 but some variables are not observed:

$$K(x_{obs}, y_{obs}) = \sum_{S \in \mathcal{V}} \sum_{z_S \in \mathcal{A}^S} p(z_S) p(x_{obs}|z_S) p(y_{obs}|z_S)$$

- A bit longer to write, but still possible

- Linear time computation

Example 5: general common subtree kernel



- Same as example 3 but subtrees not necessarily rooted
- A bit longer to write, but still possible
- Linear time computation (using three states per node)

Part 3

Application: Gene functional prediction from phylogenetic profiles

Mini introduction

- Genes are small parts of the DNA which encode proteins.
- About 6,000 genes in the baker yeast, 30,000 in human
- The sequence of the genes are (almost) known (sequencing projets)
- Next big challenge: understand the function of the genes

Definition

- The phylogenetic profile of a gene is a vector of bits which indicates the presence (1) or absence (0) of the gene in every fully sequenced genome.

Gene	human	yeast	...	HIV	E. coli
YAL001C	1	1	...	0	0
YAB002W	0	0	...	0	1
⋮	⋮	⋮	⋮	⋮	⋮

- Can be estimated *in silico* by sequence similarity search

From profile to function

- Genes are likely to be transmitted together during evolution when they participate:
  - ⋆ to a common structural complex,
  - ⋆ to a common pathway.
- Consequently genes with similar phylogenetic profiles are likely to have similar functions
- How to measure the similarity between profiles?

Naive approach

- Count the number of bits in common:

x	1	1	0	1	0	0	0	1	1	0
y	1	0	1	0	0	0	0	1	0	1

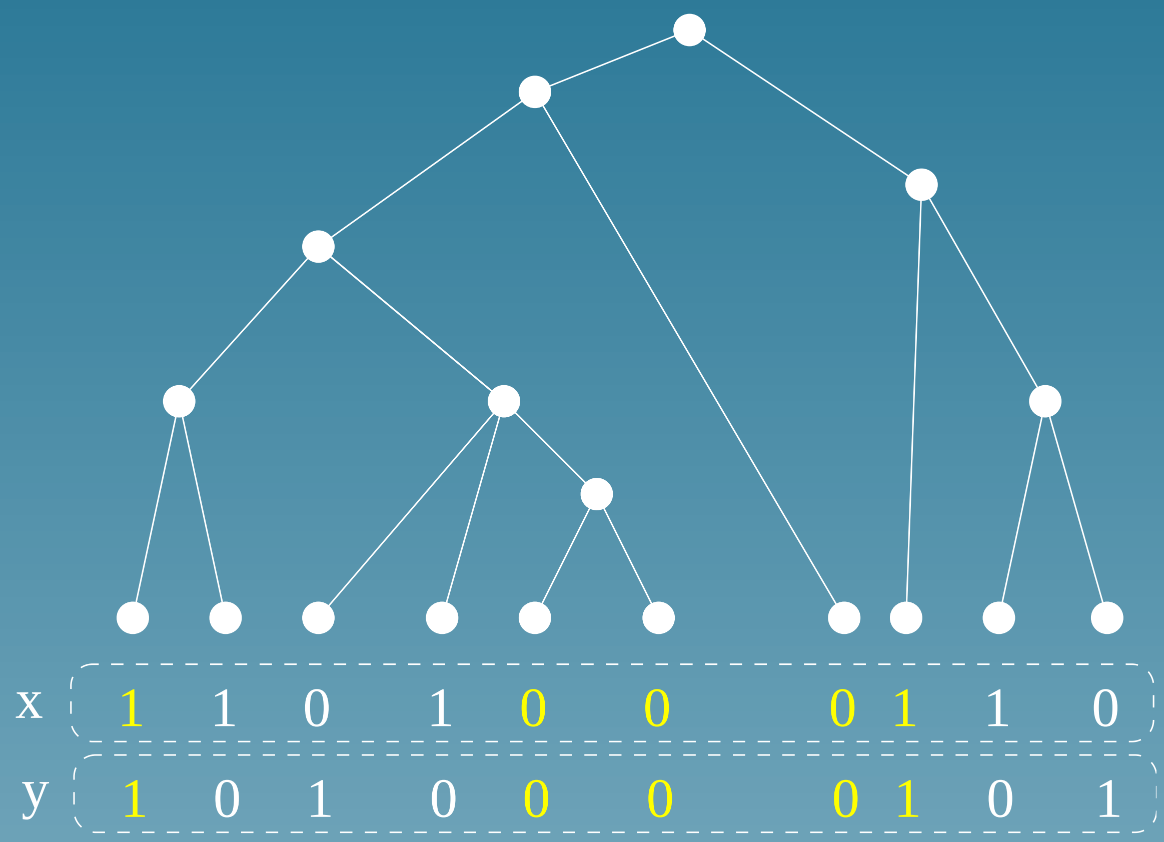
$$s(x, y) = 5$$

- Cluster or use k-NN for gene function prediction with this similarity measure (Pellegrini et al., 1999)

Limitations of the naive approach

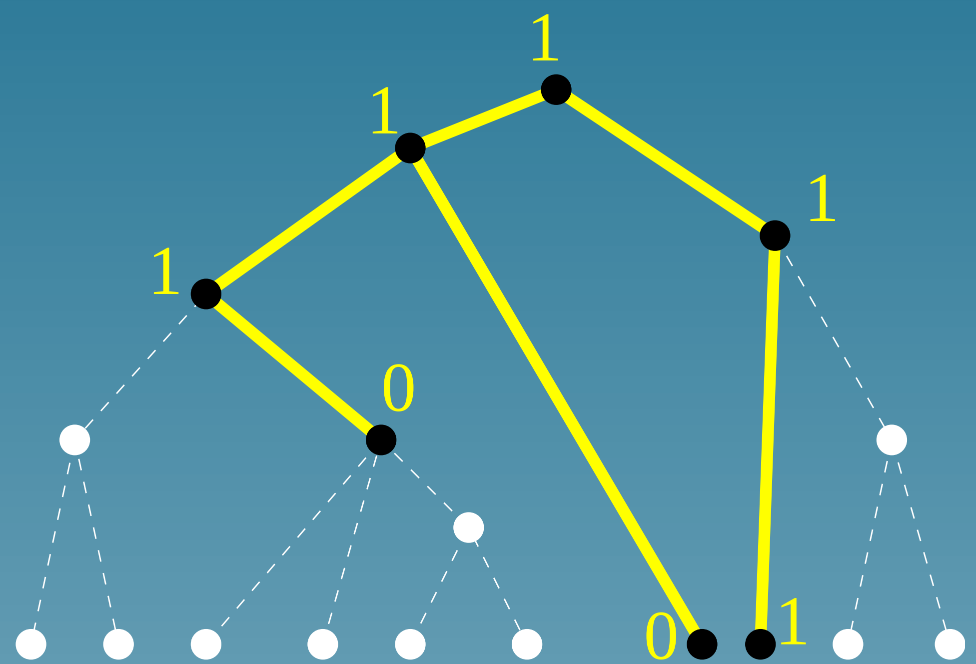
- The set of sequenced organisms has a strong influence on the similarity score (e.g., eukaryotes are under-represented)
- A more detailed understanding of when two proteins were transmitted together or not during evolution could be useful

What is not used in the naive approach



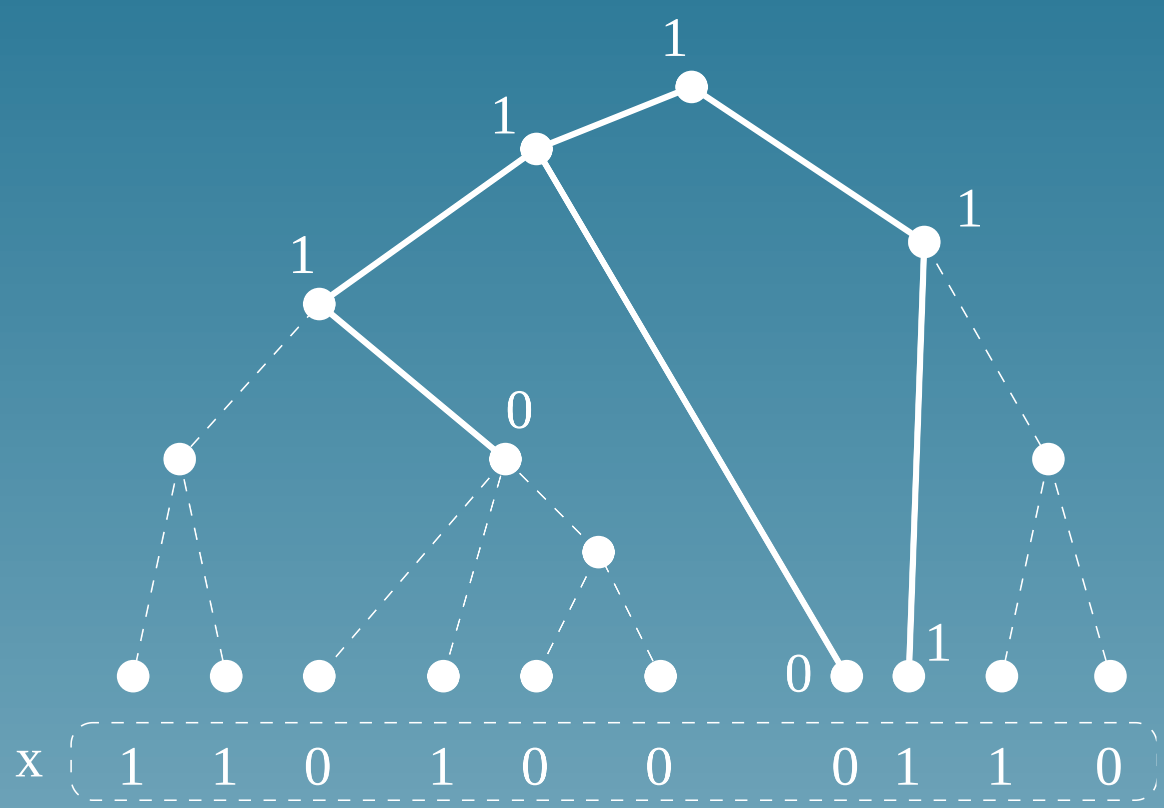
The knowledge of the phylogenetic tree.

Evolution pattern



A possible pattern of transmission during evolution defined by a rooted subtree with nodes labeled 0 or 1.

Evolution patterns and phylogenetic profiles



Is it the true story? We don't know, but...

Probabilistic model of gene transmission

- The phylogenetic tree as a tree graphical model
- Simplified model:
  - ⋆ $P(1) = 1 - P(0) = 0.9$, at the root,
  - ⋆ Along each branch transmission follows the transition matrix:

$$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$$

Probabilistic assignment of evolution pattern

For a phylogenetic profile x and an evolution pattern e:

- $P(e)$ quantifies how "natural" the pattern is
- $P(x|e)$ quantifies how likely the pattern e is the "true history" of the profile x

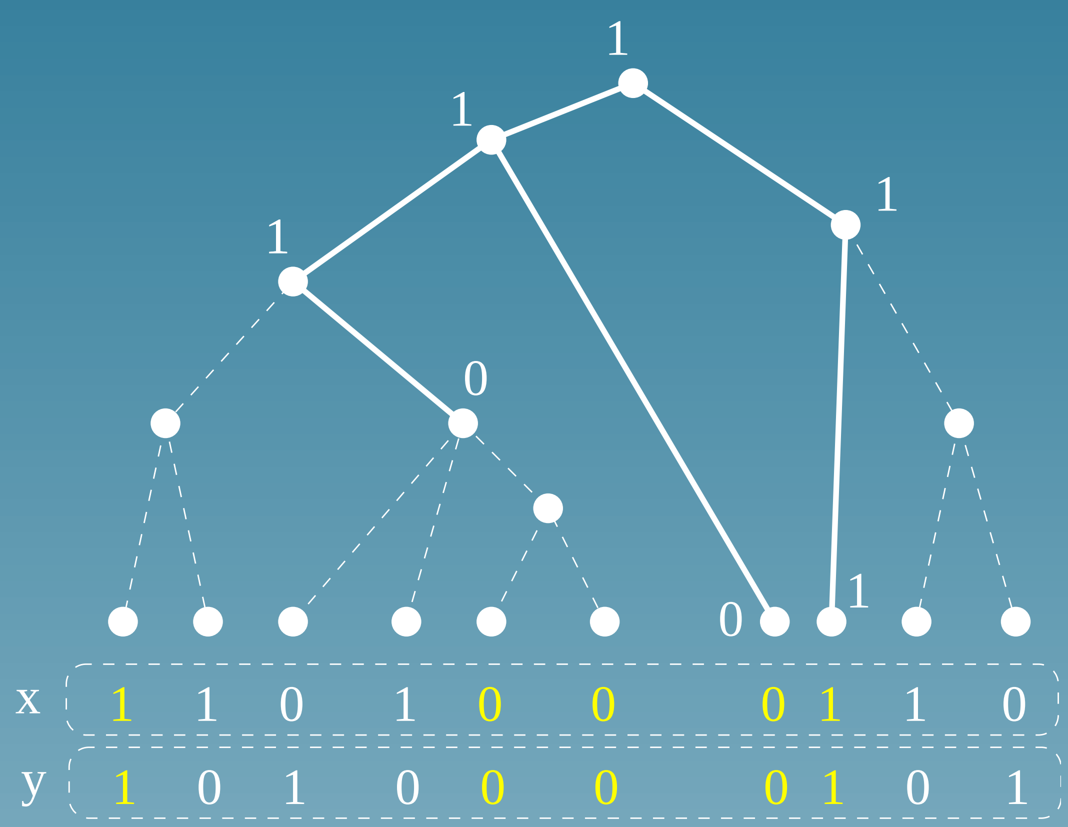
Representation of a profile in terms of evolution patterns

- Consider all possible evolution patterns $(e_1, \ldots, e_N)$, and represent each gene x by the vector:

$$\Phi(x) = \begin{pmatrix} \sqrt{P(e_1)}P(x|e_1) \\ \vdots \\ \sqrt{P(e_N)}P(x|e_N) \end{pmatrix}$$

- This leads to the probabilistic kernel described before

Comparing two profiles through evolution patterns



Gene function prediction with SVM

- Profiles for 2465 genes of *S. Cerevisiae* were computed by BLAST search (cf Pavlidis et al. 2001), using 24 genomes.
- Consensus phylogenetic tree (cf. Liberles et al. 2002) with simplified probabilistic model of gene transmission
- SVM trained to predict all functional classes of the MIPS catalog with at least 10 genes (cross-validation)
- Comparison of the tree kernel with the naive kernel

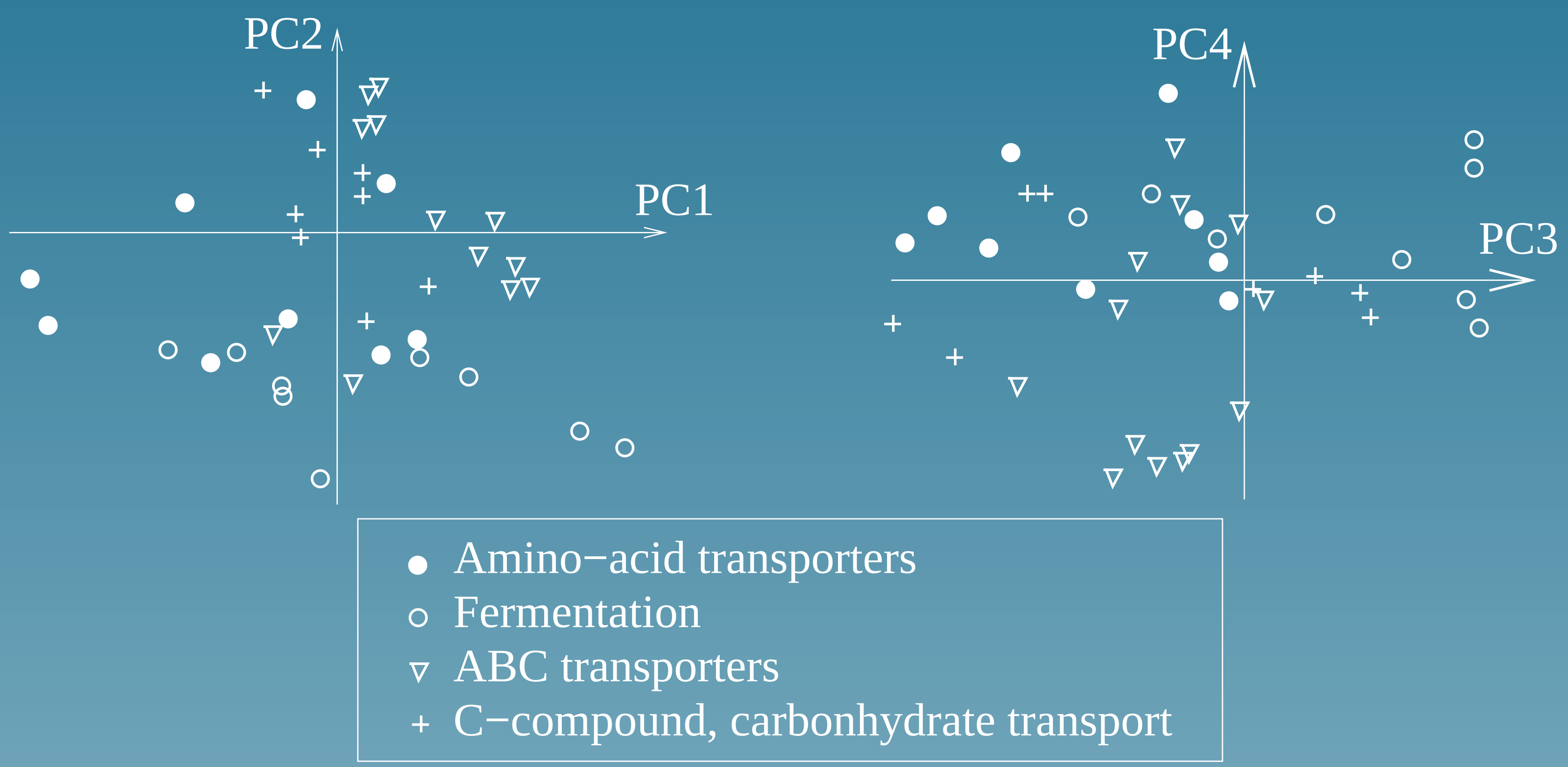
Results (ROC 50)

Functional class	Naive kernel	Tree kernel	Difference
Amino-acid transporters	0.74	0.81	**+ 9%**
Fermentation	0.68	0.73	**+ 7%**
ABC transporters	0.64	0.87	**+ 36%**
C-compound transport	0.59	0.68	**+ 15%**
Amino-acid biosynthesis	0.37	0.46	**+ 24%**
Amino-acid metabolism	0.35	0.32	- *9%*
Tricarboxylic-acid pathway	0.33	0.48	**+ 45%**
Transport Facilitation	0.33	0.28	- *15%*

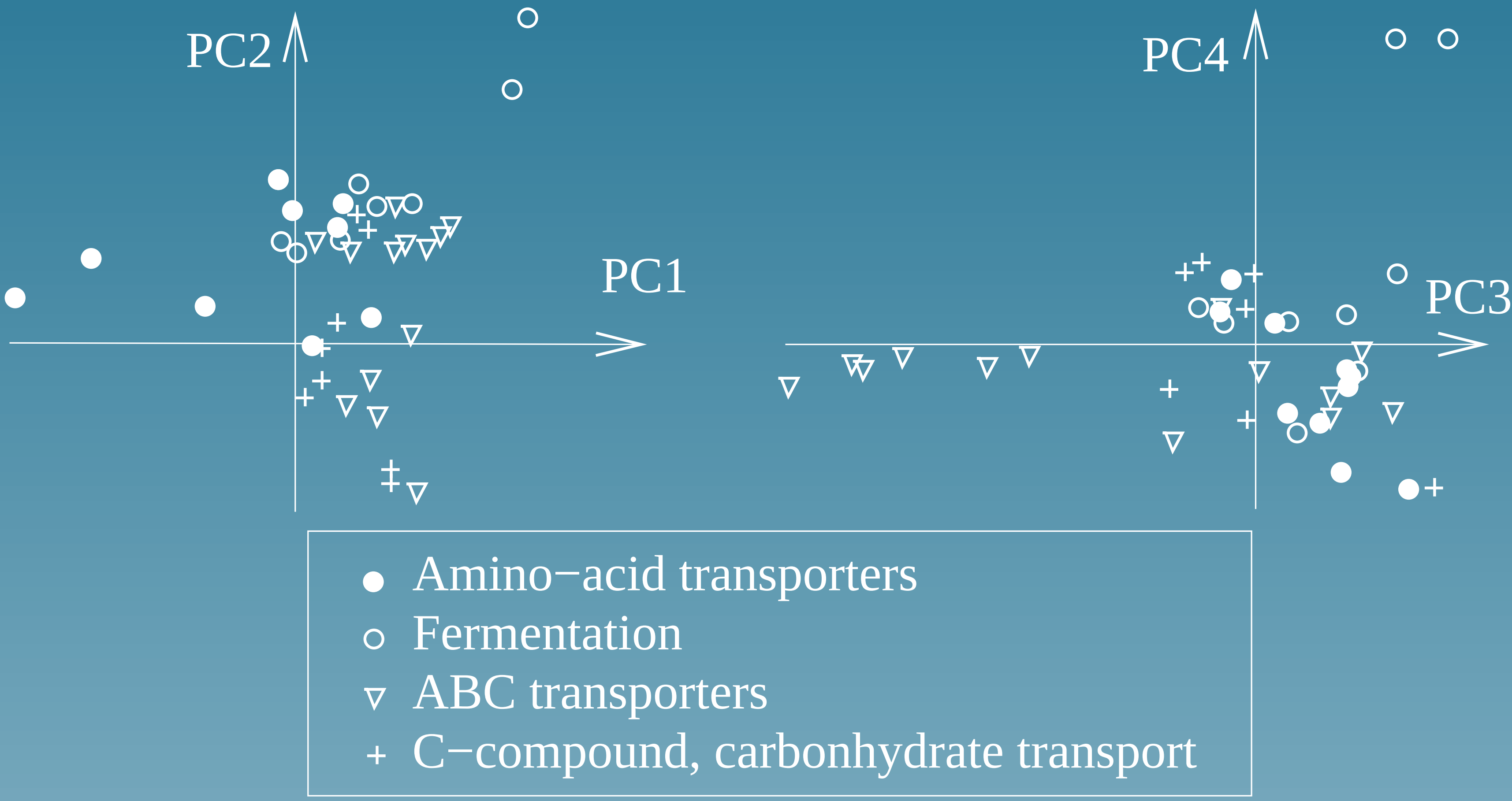
A insight into the feature space

- PCA can be performed implicitly in the feature space with a kernel function: kernel-PCA (Scholkopf et al. 1999)
- Projecting the genes on the first principal components gives an idea of the shape of the features space

Naive kernel PCA



Tree kernel PCA



Conclusion

Conclusion

- A general method to derive a kernel from a probability distribution
- Encouraging results
- Some problems and questions: diagonal dominance? Role of the prior distribution?
- Contributes to a general approach: encode genomic information into kernel functions.