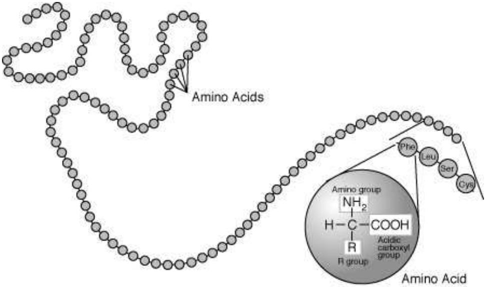
The context tree weighting kernel

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

Centre for Computational Biology
Ecole des Mines de Paris, ParisTech

"Context tree models" workshop, Telecom Paris, November 19, 2007

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	

Typical problem: supervised sequence classification

Data (training)

- Secreted proteins:

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

- Non-secreted proteins:

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

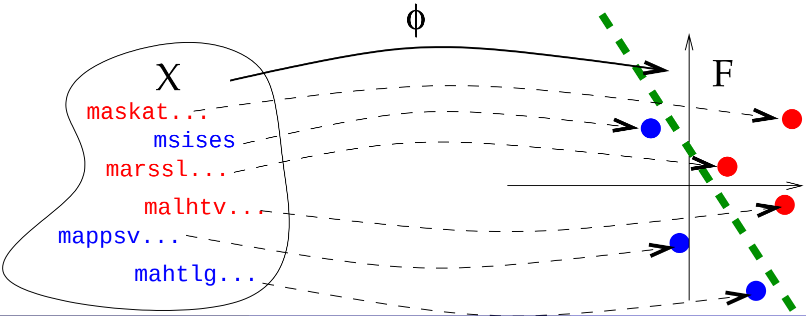
Goal

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

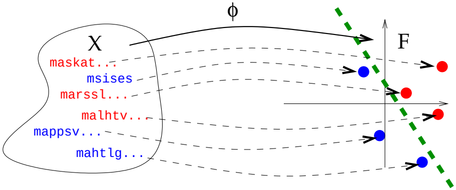
Strategy 1: Supervised classification with vector embedding

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...)



Strategy 1: Supervised classification with vector embedding



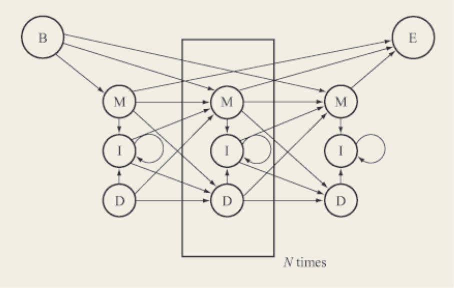
Pros

- Many algorithms exist
- Good performance in classification

Cons

- How to embed strings into vectors?
- How to include prior knowledge in the features?

Strategy 2: generative models



The idea

- Estimate a model $P_1(x)$ and $P_2(x)$ for each class
- Predict the class of a new sequence by comparing the probabilities of the sequence under both models

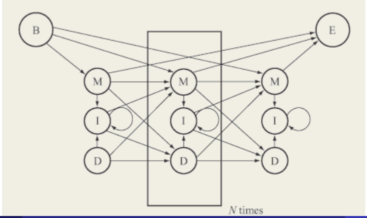
Strategy 2: generative models

Pros

- Many good models exist (Markov chains, HMM, SCFG...)
- Easy to include prior knowledge
- Good procedures to estimate models

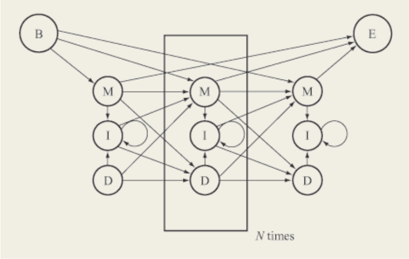
Cons

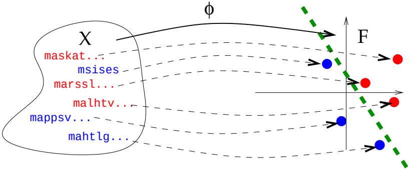
- Discrepancy between the modelling criterion and the classification criterion
- Discriminative methods often give better results



Contribution

1. A general framework to combine the pros of both approaches: kernel methods with mutual information kernels
2. A particular case where this framework can be applied efficiently: the context-tree weighting kernel





Outline

1. From generative models to kernel methods
2. Context-tree weighting kernel
3. Conclusion

Outline

Outline

1. From generative models to kernel methods
2. Context-tree weighting kernel
3. Conclusion

Outline

Ingredients

1. $\mathcal{X}$ the space of data
   - e.g., the set of finite-length strings
2. A parametric set of probability distributions over $\mathcal{X}$:

$$\{P_\theta, \theta \in \Theta \subset \mathbb{R}^m\}$$

   - e.g., a Markov chain, HMM, SCFG, ...
3. A prior distribution $w(d\theta)$ over Θ
   - e.g., Dirichlet prior...

Ingredients

1. $\mathcal{X}$ the space of data
   - e.g., the set of finite-length strings
2. A parametric set of probability distributions over $\mathcal{X}$:

$$\{P_\theta, \theta \in \Theta \subset \mathbb{R}^m\}$$

   - e.g., a Markov chain, HMM, SCFG, ...
3. A prior distribution $w(d\theta)$ over Θ
   - e.g., Dirichlet prior...

Ingredients

1. $\mathcal{X}$ the space of data
   - e.g., the set of finite-length strings
2. A parametric set of probability distributions over $\mathcal{X}$:

$$\{P_\theta, \theta \in \Theta \subset \mathbb{R}^m\}$$

   - e.g., a Markov chain, HMM, SCFG, ...
3. A prior distribution $w(d\theta)$ over Θ
   - e.g., Dirichlet prior...

Fitting a generative model

The problem

- Given a training set of n data $D = (x_1, \ldots, x_n)$ in $\mathcal{X}$,
- Estimate a distribution $P_D(dx)$ over $\mathcal{X}$ to model D

Estimation strategy

- Parameter estimation: take $P_D = P_{\hat{\theta}}$, where $\theta \in \Theta$ is estimated, e.g., by maximum likelihood or MAP.
- Bayesian approach: take $P_D = \int_\Theta P_\theta w(d\theta|D)$, where $w(d\theta|D)$ is the posterior distribution.

Result

- P_D is a distribution over $\mathcal{X}$, in the convex hull of the model.
- The probability of the strings in the training set under P is "large".

Fitting a generative model

The problem

- Given a training set of n data $D = (x_1, \ldots, x_n)$ in $\mathcal{X}$,
- Estimate a distribution $P_D(dx)$ over $\mathcal{X}$ to model D

Estimation strategy

- Parameter estimation: take $P_D = P_{\hat{\theta}}$, where $\theta \in \Theta$ is estimated, e.g., by maximum likelihood or MAP.
- Bayesian approach: take $P_D = \int_\Theta P_\theta w(d\theta|D)$, where $w(d\theta|D)$ is the posterior distribution.

Result

- P_D is a distribution over $\mathcal{X}$, in the convex hull of the model.
- The probability of the strings in the training set under P is "large".

Fitting a generative model

The problem

- Given a training set of n data $D = (x_1, \ldots, x_n)$ in $\mathcal{X}$,
- Estimate a distribution $P_D(dx)$ over $\mathcal{X}$ to model D

Estimation strategy

- Parameter estimation: take $P_D = P_{\hat{\theta}}$, where $\theta \in \Theta$ is estimated, e.g., by maximum likelihood or MAP.
- Bayesian approach: take $P_D = \int_\Theta P_\theta w(d\theta|D)$, where $w(d\theta|D)$ is the posterior distribution.

Result

- P_D is a distribution over $\mathcal{X}$, in the convex hull of the model.
- The probability of the strings in the training set under P is "large".

Fitting a generative model

Summary

In both cases

- MODELLING: The model $\{P_\theta, \theta \in \Theta\}$ defines a set of basic distributions
- LEARNING: The fitting procedure finds a convex combination:

$$P_D = \int_{\theta \in \Theta} P_\theta w_D(d\theta)\,,$$

where w_D is a distribution over Θ that depends on the training set D, following some principle (ML, MAP, Bayes...)

Generative models for discrimination

The problem

Given two sets D_1 and D_2 representing two populations (e.g., secreted vs. non-secreted proteins), estimate a score function $f(x)$ that discrimates both populations.

The generative approach

- Estimate P_{D_1} and P_{D_2} using the methodology to fit generative models on D_1 and D_2
- Form the score:

$$f(x) = P_{D_1}(x) - P_{D_2}(x) + cte\,.$$

- f is an affine function of the $\{P_\theta, \theta \in \Theta\}$

Fitting generative model for classification

Summary

In both cases

- MODELLING: The model $\{P_\theta, \theta \in \Theta\}$ defines a set of basic distributions
- LEARNING: The fitting procedure finds an affine combination:

$$f = \int_{\theta \in \Theta} P_\theta w_D(d\theta) \,,$$

where w_D is a signed measure, following some principle.

Discrimination with generative models

Good

- Modelling

Bad

- Learning principles not adapted to classification

A natural idea

- Keep the model for MODELLING: f should be an affine function of $\{P_\theta, \theta \in \Theta\}$, i.e.:

$$f(x) = \int_{\theta \in \Theta} P_\theta(x) w_D(d\theta),$$

where w_D is a signed measure that depends on the training set D.
- Change the procedure for LEARNING: Replace the generative model fitting principles by learning principles for discrimination.

Discrimination with generative models

Good

- Modelling

Bad

- Learning principles not adapted to classification

A natural idea

- Keep the model for MODELLING: f should be an affine function of $\{P_\theta, \theta \in \Theta\}$, i.e.:

$$f(x) = \int_{\theta \in \Theta} P_\theta(x) w_D(d\theta)\,,$$

  where w_D is a signed measure that depends on the training set D.
- Change the procedure for LEARNING: Replace the generative model fitting principles by learning principles for discrimination.

Reformulation

Learning in Hilbert space

- Let $\mathcal{H} = L^2(\Theta, w)$ be the Hilbert space of functions $f : \Theta \to \mathbb{R}$ with inner product:

$$\langle f, g \rangle_{\mathcal{H}} = \int_{\Theta} f(\theta) g(\theta) w(d\theta) \,.$$

- Let the embedding $\Phi : \mathcal{X} \to \mathcal{H}$ defined by:

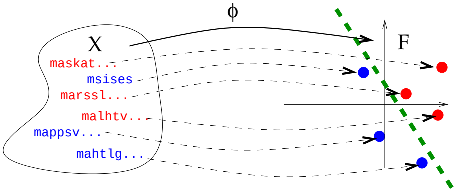
$$\Phi(x) = \{\theta \mapsto P_\theta(x)\} \,.$$

- We want to find a linear function in $\mathcal{H}$, i.e., a vector $u \in \mathcal{H}$ with:

$$f(x) = \langle \Phi(x), u \rangle_{\mathcal{H}} = \int_{\Theta} P_\theta(x) u(\theta) w(d\theta) \,,$$

that discriminates between the two classes.

Example: support vector machine



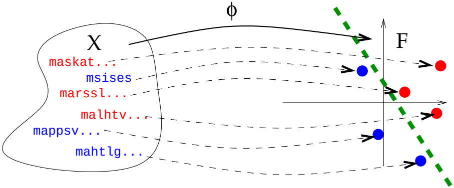
SVM algorithm

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \langle \Phi(x_i), \Phi(x) \rangle_{\mathcal{H}}\right) ,$$

where $\alpha_1, \ldots, \alpha_n$ solve, under the constraints $0 \leq \alpha_i \leq C$:

$$\min_{\alpha}\left(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}} - \sum_{i=1}^{n} \alpha_i\right) .$$

Kernel trick



Problem

$\Phi(x) = \{\theta \mapsto P_\theta(x)\}$ is infinite-dimensional, can not be computed nor manipulated.

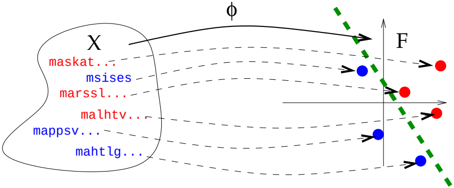
Kernel

- The kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is:

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

- If $K(x, x')$ can be computed, learning algorithms can be used! (kernel methods)

Kernel trick



Problem

$\Phi(x) = \{\theta \mapsto P_\theta(x)\}$ is infinite-dimensional, can not be computed nor manipulated.

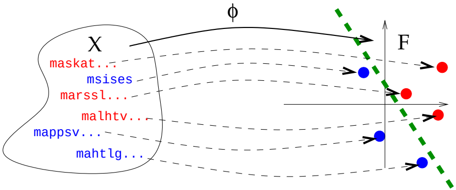
Kernel

- The kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is:

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

- If $K(x, x')$ can be computed, learning algorithms can be used! (kernel methods)

Example: support vector machine with kernels



SVM algorithm

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x)\right) ,$$

where $\alpha_1, \ldots, \alpha_n$ solve, under the constraints $0 \leq \alpha_i \leq C$:

$$\min_{\alpha}\left(\frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i\right) .$$

From generative models to kernel methods

Summary

- A model defines a family of distributions $\mathcal{M} = \{P_\theta, \theta \in \Theta \subset \mathbb{R}^m\}$.
- Fitting a model to empirical data usually means finding a function f in the convex hull or linear span of $\mathcal{M}$:

$$f = \int_{\theta \in \Theta} P_\theta w_D(d\theta) \,,$$

- Equivalently f is a linear function in the Hilbert space $\mathcal{X}$ after the embedding $\Phi(x) = \{\theta \mapsto P_\theta(x)\}$
- Powerful kernel methods (e.g., SVM) can be used to infer such a linear function as soon as the mutual information kernel (Seeger, 2002) can be computed:

$$K(x, x') = \left\langle \Phi(x), \Phi(x') \right\rangle_{\mathcal{H}} = \int_\Theta P_\theta(x) P_\theta(x') w(d\theta) \,.$$

Outline

1. From generative models to kernel methods
2. Context-tree weighting kernel
3. Conclusion

Warm-up example

- $\mathcal{X}$ the set of finite-length binary strings
- $P_\theta(X = 1) = \theta$ and $P_\theta(X = 0) = 1 - \theta$ a model for independent 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

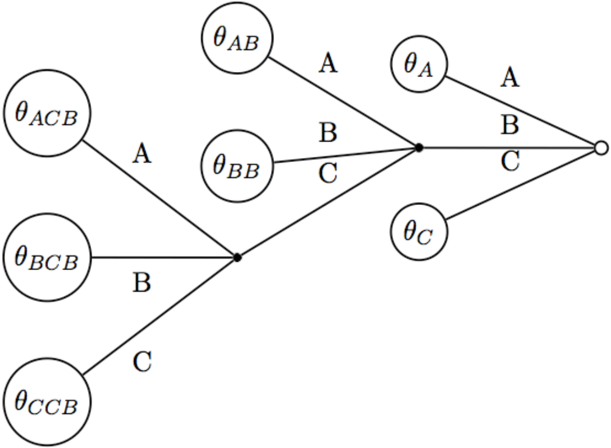
Definition

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 weighting kernel

Priors

- We have a family of models:

$$\mathcal{M} = \left\{ P_{\mathcal{D},\theta} \, , \, \mathcal{D} \in \mathcal{T}, \theta \in \Sigma^{\mathcal{D}} \right\}$$

- We define a prior over $\mathcal{M}$ that factorizes as:

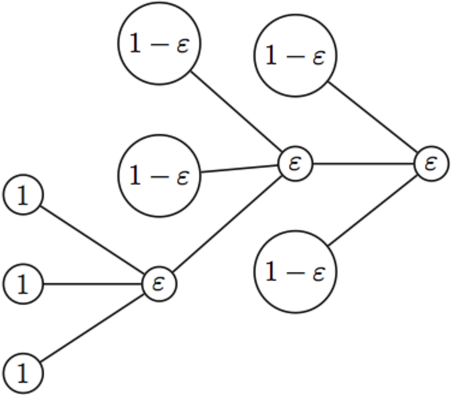
$$\pi(\mathcal{D}, d\theta) = \pi(\mathcal{D})\pi(d\theta|\mathcal{D}) \, .$$

- The resulting context-tree weighting kernel is then

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

Prior $\pi(\mathcal{D})$

The set of suffix trees of depth up to *D* is endowed with the distribution of a branching process



$$P(\mathcal{D}) = \epsilon^3(1-\epsilon)^4$$

Prior $\pi(d\theta \,|\, \mathcal{D})$

- θ is made of $|\mathcal{D}|$ multinomial parameters. We endow them independently with a Dirichlet prior:

$$\pi(d\theta \,|\, \mathcal{D}) = \prod_{s \in \mathcal{D}} \omega_\beta(d\theta_s)$$

with

$$\omega_\beta(d\theta) \sim \prod_{i=1}^{d} \theta_i^{\beta_i - 1} \lambda(d\theta)\,.$$

- We can also consider Dirichlet mixtures:

$$\omega_{\gamma,\beta}(d\theta_s) = \sum_{k=1}^{n} \gamma^{(k)} \omega_{\beta^{(k)}}(d\theta_s)\,.$$

The context-tree kernel

Theorem (Cuturi et al., 2004)

- *For these choices of priors, the context-tree kernel:*

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

  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.*

Computation

- The CTW algorithm (Willems et al., 1995) provides a linear-time algorithm to compute the coding probability:

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

- The extension to $K(x, x')$ is obvious: it roughly corresponds to computing the coding probability of the concatenation of x and x' because

$$P_{\mathcal{D},\theta}(x)P_{\mathcal{D},\theta}(x') = P_{\mathcal{D},\theta}(xx')\,.$$

- The extension from Dirichlet priors to Dirichlet mixture does not increase the complexity of the algorithm.

Length normalization

- In practice $K(x, x')$ decreases exponentially with the length of x and x'
- This means that the sequences are almost orthogonal in the Hilbert space, which prevents learning (issue of diagonal dominance)
- Possible length normalization:

$$K_\sigma\left(x, x'\right) = \sum_{\mathcal{D}} \int_{\theta \in \Sigma^{\mathcal{D}}} P_{\mathcal{D},\theta}(x)^{\frac{\sigma}{|x|}} P_{\mathcal{D},\theta}(x')^{\frac{\sigma}{|x'|}} \pi(d\theta|\mathcal{D})\pi(\mathcal{D})\,.$$

Coding interpretation

- The cosine between $\Phi(x)$ and $\Phi(x')$ in $\mathcal{H}$ is:

$$c(x,x') = \frac{\langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}}{\parallel \Phi(x) \parallel_{\mathcal{H}} \parallel \Phi(x') \parallel_{\mathcal{H}}} = \frac{K(x,x')}{\sqrt{K(x,x)K(x,x')}} .$$

- Therefore:

$$-\log_2 c(x,x') = -\log K(x,x') + \frac{1}{2}\left(-\log_2 K(x,x') - \log_2 K(x',x')\right) .$$

- $-\log_2 K(x,x')$ is the length of the code of x and x' coded together
- $-logc(x,x')$ is therefore the gain in compression when x and x' are coded together, compared to the situation where they are compressed independently to each other.
- This is known as the mutual information between x and x'.

Semigroup kernel interpretation

- Let $\Psi(x)$ the statistics of x needed to compute the kernel.
- The CTW kernel has the particularity that $K(x, x')$ is a function of $\Psi(x) + \Psi(x')$ (i.e., roughly speaking a function of the concatenation of x and x').
- The set of strings endowed with the concatenation is a semigroup (more precisely the set of $\Psi(x)$ endowed with addition is a semigroup)
- the CTW kernel is a semigroup positive definite function;

$$K(x, x') = g(\Psi(x) + \Psi(x'))$$

- Such semigroup kernels can be characterized in more generality (representation as convex combination of semigroup characters), and more kernel can be imagined (Cuturi et al., 2006).

Application: SCOP classification benchmark

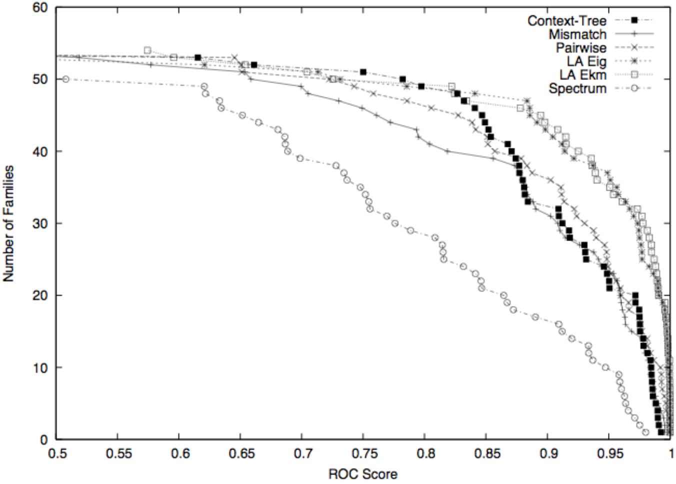


Fig. 4. Performance of all considered kernels on the problem of recognizing domain's superfamily. The curve shows the total number of families for which a given methods exceeds a ROC score threshold. CTK denotes the context-tree kernel set with $\sigma = 2$, $\varepsilon = 1/20$, Jeffrey's prior and depth $D = 4$.

Outline

Conclusion

- Mutual information kernels allow to use well-designed probabilistic models with a variety of learning algorithms
- The CTW kernel is a practical way to compute such a kernel
- Suggests systematic ways to make kernels with other compression algorithms
- Can be extended in the context of semigroup kernels

Thanks Marco Cuturi!