Completion of biological networks: the output kernel trees approach

Pierre Geurts[1,2] Nizar Touleimat [1,3] Marie Dutreix [3]
Florence d'Alché-Buc [1]

[1]IBISC FRE CNRS 2873 & Epigenomics Project, GENOPOLE, Evry, France

[2]Dept. of EECS & CBIG/GIGA, University of Liège, Belgium

[3]UMR 2027 CNRS-IC, Institut Curie, Orsay, France

Probabilistic modeling and Machine Learning in Structural and Systems Biology
June 17, 2006

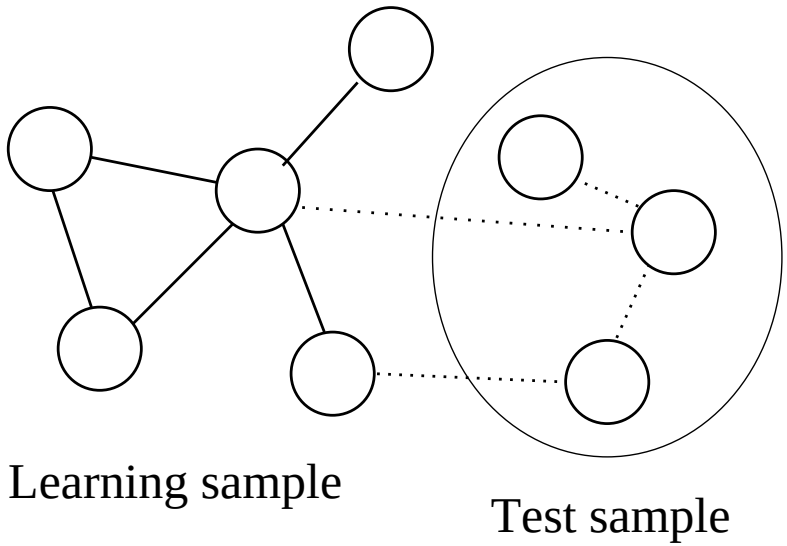
Introduction

- **Biological networks**
  - gene regulatory networks
  - protein-protein interaction networks
  - metabolic networks and enzyme networks
- **Motivation for the inference of biological networks from data**
  - reverse-modeling of gene regulatory networks
  - completion of existing networks and inference of properties of ”new” genes/proteins (Yamashini et al. 04, Kato et al. 05, Noble et al. 05)
- Focus on completion of partially known networks

Outline

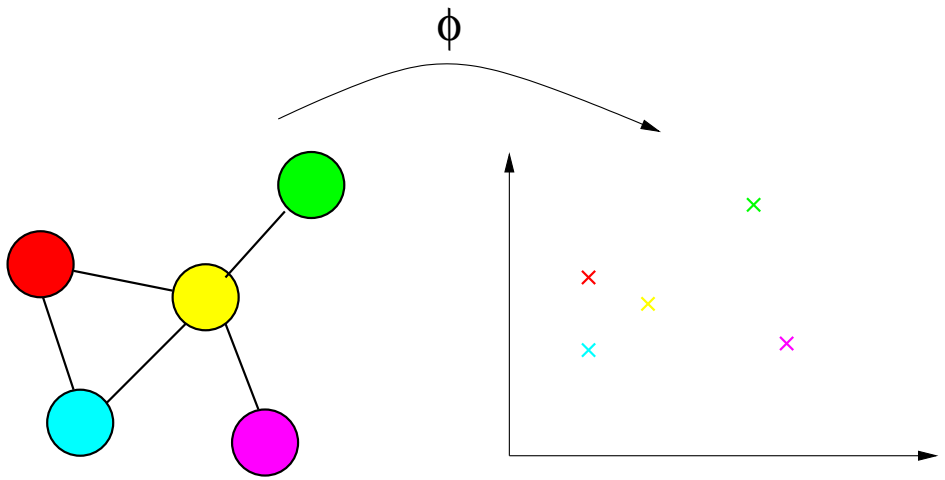
Outline

Supervised graph inference (Yamanishi and Vert, 2004)



- From a partially known network where each vertex v is described by some input feature vector $x(v)$, predict the edges involving new vertices described by their input feature vector
- Example: each vertex corresponds to a gene/protein/enzyme, each input feature to a gene/protein/enzyme descriptor

A general solution based on considering a kernelized output space



- Define a kernel k on pairs of vertices (with output feature map ϕ) such that $k(v, v')$ is large ($||\phi(v) - \phi(v')||^2$ is small) whenever v and v' are connected in the graph
- Use a ML method that can handle a kernelized output space to get an approximation $\hat{k}(v, v')$ of the kernel value between two vertices v and v' described by their feature input vector $x(v)$ and $x(v')$
- Connect these two vertices if $\hat{k}(v, v') > k_{th}$

(by varying k_{th} we get different tradeoffs between true positive and false positive rates)

A kernel on graph nodes

- Diffusion kernel (Kondor and Lafferty, 2002):
  The Gram matrix K with $K_{i,j} = k(v_i, v_j)$ is given by:

$$K = \exp(-\beta L)$$

  where the graph Laplacian L is defined by:

$$L_{i,j} = \begin{cases} d_i & \text{the degree of node } v_i \text{ if } i = j; \\ -1 & \text{if } v_i \text{ and } v_j \text{ are connected;} \\ 0 & \text{otherwise.} \end{cases}$$

- As the diffusion coefficient β increases, kernel values diffuse more completely through the graph

Outline

Output kernel trees

- Output kernel trees (OK3) are a kernelization of the output of standard regression trees
- It allows these methods to handle any structured output space over which a kernel may be defined
- By extension, this method allows to learn a kernel as a function of a couple of input vectors.

see (Geurts, Wehenkel, d'Alch é, ICML 2006) for general features of this method

Back to the classic framework of supervised learning

- Supervised learning problem:

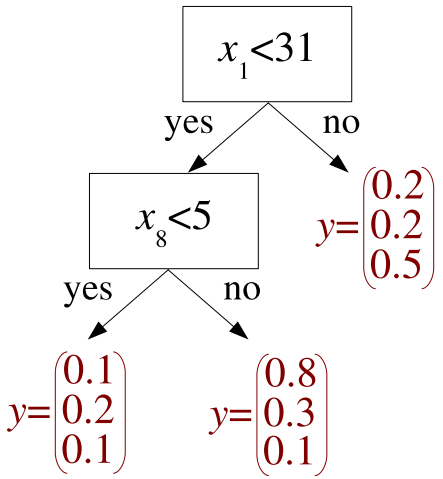
  From a learning sample $\{(x_i, y_i)|i = 1, \ldots, N\}$ *with* $x_i \in \mathcal{X}$ *and* $y_i \in \mathcal{Y}$*, find a function* $f : \mathcal{X} \to \mathcal{Y}$ *that minimizes the expectation of some loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to I\!R$ *over the joint distribution of input/output pairs:*

$$E_{x,y}\{\ell(f(x), y)\}$$

- Classification: $\mathcal{Y}$ is a finite set and $\ell(y, y') = 1(y \neq y')$.
- Regression: $\mathcal{Y} = I\!R$ and $\ell(y, y') = (y - y')^2$
- Multiple outputs regression : $\mathcal{Y} = I\!R^I$ and $\ell(y, y') = ||y - y'||^2$

Regression trees on multiple outputs

$\mathcal{Y} = I\!R^n$ and $\ell(y, y') = ||y - y'||^2$



Algorithm:

- Recursively split the learning sample with tests based on one attribute of the inputs trying to reduce as much as possible the empirical variance of the outputs
- Stop when the outputs are constant in the leaf (or as soon as some stopping criterion is met)

Multiple outputs regression trees

- The best split for the data S corresponds to the test T that maximises the ”variance” reduction:

$$\text{Score}_R(T, S) = \text{var}\{y|S\} - \frac{N_l}{N}\text{var}\{y|S_l\} - \frac{N_r}{N}\text{var}\{y|S_r\},$$

where N is the size of S, N_l (resp. N_r) the size of subset S_l (resp. subset S_r) for which the test is true (resp. false)

- $\text{var}\{Y|S\}$ denotes the empirical “variance” of the output Y in the subset S:

$$\text{var}\{y|S\} = \frac{1}{N}\sum_{i=1}^{N} ||y_i - \overline{y}||^2 \text{ with } \overline{y} = \frac{1}{N}\sum_{i=1} y_i$$

which is the average squared distance to the center of mass (or the sum of the empirical variances taken on each dimension).

Prediction with multiple outputs regression trees

- Predictions at leaf nodes L are computed as:

$$\hat{y}_L = \frac{1}{N_L} \sum_{i=1} y_i$$

- the approximation is piece-wise constant

Output kernel trees (OK3): use the kernel trick in the output space

- Assume some (semi-definite positive) kernel k with a corresponding mapping ϕ ($k(y, y') =< \phi(y), \phi(y') >$) that maps the outputs y in a characteristic feature space H endowed with k.

Output kernel trees (OK3): use the kernel trick in the output space

- Learning stage: the algorithm is similar to standard regression trees except that the variance reduction is computed by:

$$\text{var}\{\phi(y)|S\} = \frac{1}{N}\sum_{i=1}^{N} k(y_i, y_i) - \frac{1}{N^2}\sum_{i,j=1}^{N} k(y_i, y_j)$$

- Kernel predictions are obtained by:

$$\hat{k}(y, y') = \frac{1}{N_L N_{L'}}\sum_{i=1}^{N_L}\sum_{j=1}^{N_{L'}} k(y_i^L, y_j^{L'})$$

Algorithm devoted to the supervised network inference

- Assume a mapping ϕ that "projects" vertices into a Hilbert space endowed with the corresponding kernel k and that we have actually access to a learning sample $\{x(v_1), ..., x(v_n), k(v_i, v_j), i, j = 1...n\}$.
- Learning an approximation of k is perfomed using the kernel trick:
  - grow a multiple output regression tree to get an approximation $\hat{\phi}(x(v))$ of $\phi(v)$
  - compute a kernel prediction between two nodes v and v' as:

$$\hat{k}(v, v') = \langle \hat{\phi}(x(v)), \hat{\phi}(x(v')) \rangle$$

Regression trees in output feature space

- The ”variance” in score computation becomes:

$$\text{var}\{\phi(v)|S\} = \frac{1}{N}\sum_{i=1}^{N}||\phi(v_i) - \frac{1}{N}\sum_{i=1}\phi(v_i)||^2$$

- Using decomposition of the norm into inner products we have:

$$\text{var}\{\phi(v)|S\} = \frac{1}{N}\sum_{i=1}^{N}k(v_i, v_i) - \frac{1}{N^2}\sum_{i,j=1}^{N}k(v_i, v_j)$$

- From kernel values only, we can thus grow a regression tree that minimizes output feature space variance

Regression trees in output feature space

- The prediction at a leaf node L becomes:

$$\hat{\phi}_L = \frac{1}{N_L} \sum_{i=1}^{N_L} \phi(v_i^L),$$

  where $\{v_1^L, \ldots, v_{N_L}^L\}$ are the vertices from the training sample contained in that leaf
- If $x(v)$ (resp. $x(v')$) reaches leaf L (resp. L'), the kernel prediction becomes:

$$\hat{k}(v, v') = \langle \hat{\phi}_L, \hat{\phi}_{L'} \rangle = \frac{1}{N_L N_{L'}} \sum_{i=1}^{N_L} \sum_{j=1}^{N_{L'}} \langle \phi(v_i^L), \phi(v_j^{L'}) \rangle,$$

  which may also be rewritten in terms of kernel values only:

$$\hat{k}(v, v') = \frac{1}{N_L N_{L'}} \sum_{i=1}^{N_L} \sum_{j=1}^{N_{L'}} k(v_i^L, v_j^{L'}).$$

Ensemble methods

- Ensemble methods based on randomization can be applied directly (e.g., bagging, random forests, extra-trees (Geurts et al. 05))
  - Grow several trees in parallel and average their predictions
  - Greatly improve accuracy of single trees by reducing their variance
- Ensemble predictions in the output feature space may be written:

$$\hat{\phi}_{ens}(v) = \sum_{i=1}^{N} k_{ens}(v, v_i)\phi(v_i), \text{ with } k_{ens}(v, v') = M^{-1} \sum_{j=1}^{M} k_{t_j}(v, v'),$$

where $k_{t_j}(v, v') = N_L^{-1}$ if v and v' reach the same leaf L in the tree t_j and 0 otherwise.

- Kernel predictions can then be computed by:

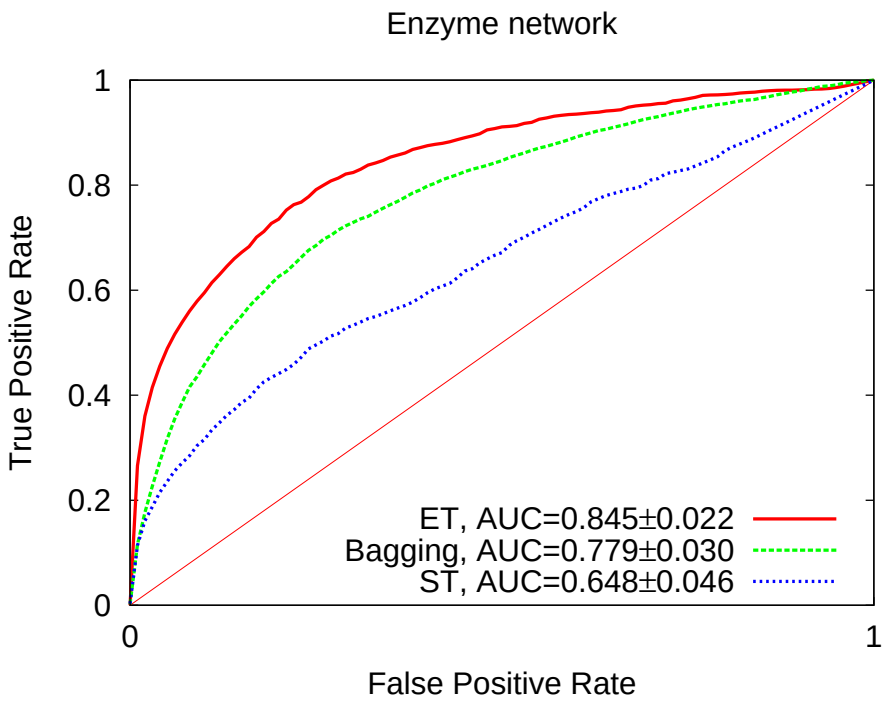
$$\hat{k}(v, v') = \sum_{i=1}^{N} \sum_{j=1}^{n} k_{ens}(v, v_i) k_{ens}(v', v_j) k(v_i, v_j)$$

Outline

Biological networks

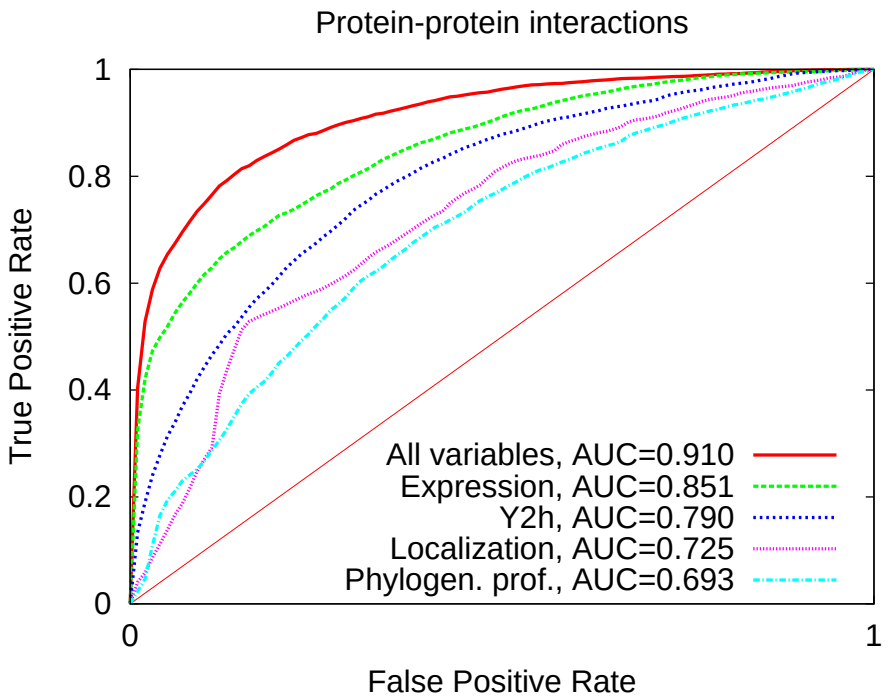
- Application to two networks in the Yeast:
  - Protein-protein interaction network: 984 proteins, 2478 edges (Kato et al., 2005)
  - Enzyme network: 668 enzymes and 2782 edges (Yamanishi et al., 2005)
- Input features:
  - Expression data: expression of the gene in 325 experiments
  - Phylogenetic profiles: presence or absence of an ortholog in 145 species
  - Localization data: presence or absence of the protein in 23 intracellular location
  - Yeast two hybrid data: data from a high-throughput experiment to detect protein-protein interactions

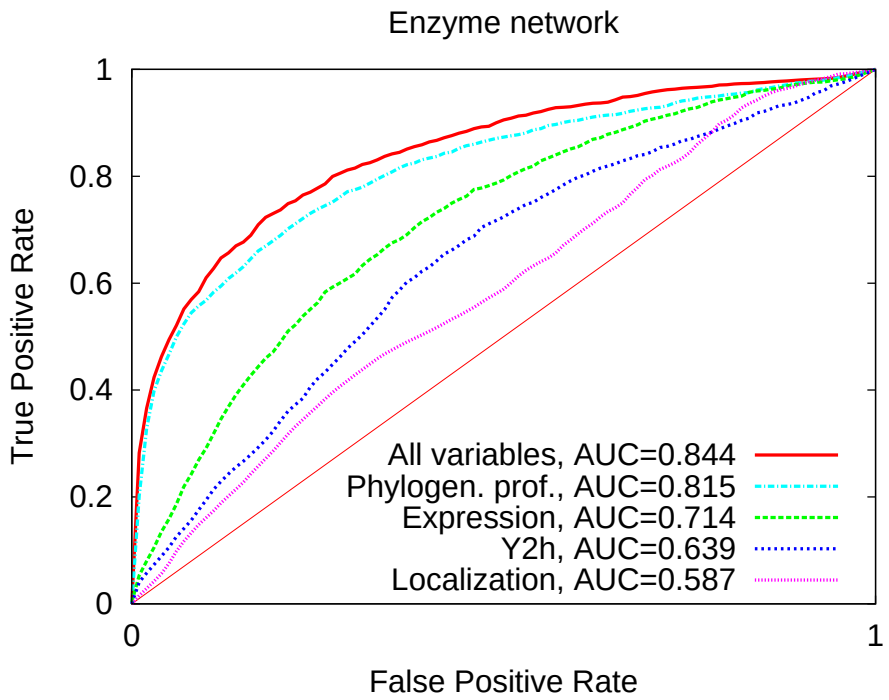
Comparison of different tree based methods



Diffusion kernel as a graph kernel, 10-fold cross-validation, ensembles of 100 output kernel trees

Comparison of different sets of features





Diffusion kernel as a graph kernel, 10-fold cross-validation, ensembles of 100 output kernel trees, extra-trees randomization method

Comparison with full kernel based methods

Protein network

Inputs	OK3+ET	[1]
expr	0.851	0.776
phy	0.693	0.767
loc	0.725	0.788
y2h	0.790	0.612
All	**0.910**	**0.939**

Enzyme network

Inputs	OK3+ET	[2]
expr	0.714	0.706
phy	0.815	0.747
loc	0.587	0.577
All	**0.847**	**0.804**

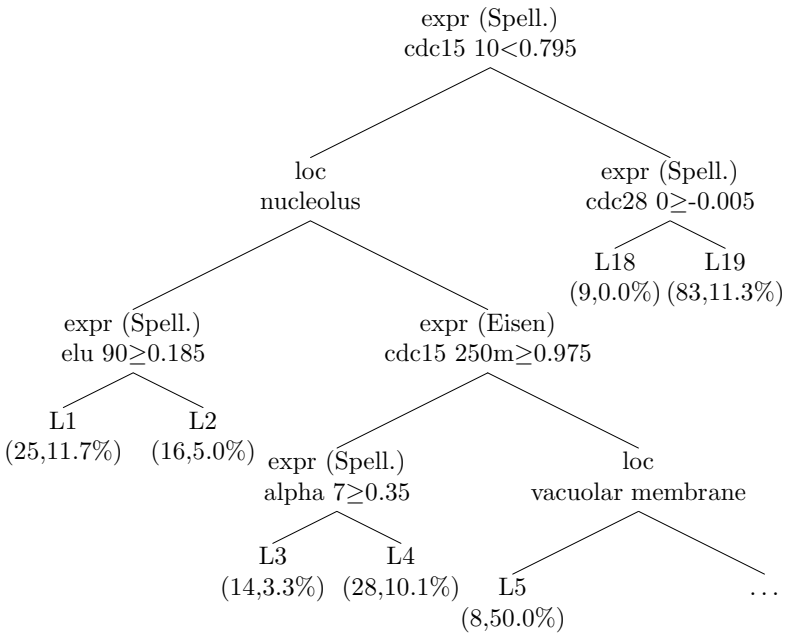
- [1] Kato et al., ISMB 2005: EM based algorithm for kernel matrix completion
- [2] Yamanishi et al., ISMB 2005: compare a kernel canonical correlation analysis based solution and a metric learning approach

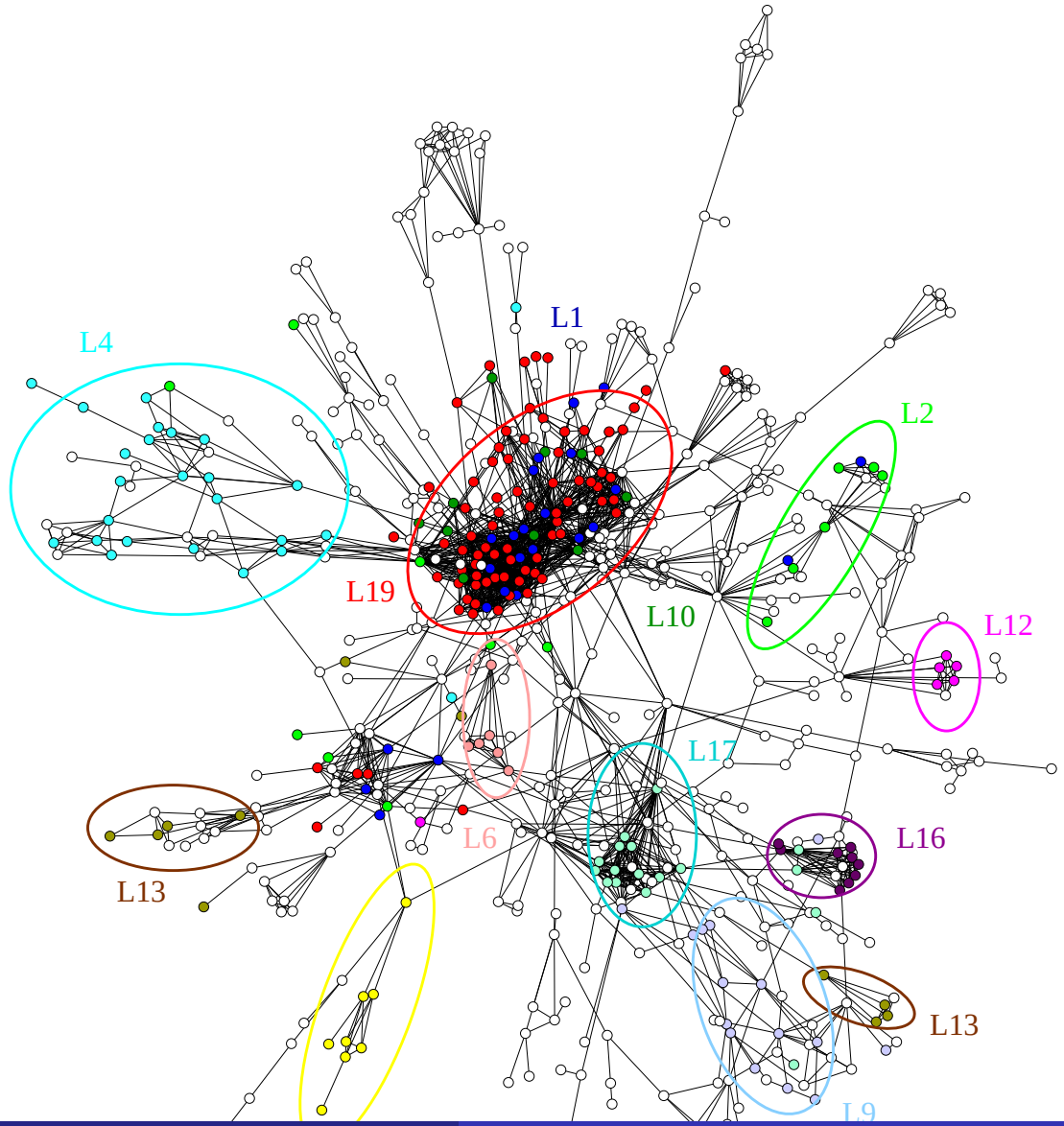
Robustness

Evolution of the AUC when x% of the edges are randomly deleted in the learning sample (OK3+ET, 100 trees)

	0%	20%	50%	80%
Protein network	0.910	0.906	0.896	0.883
Enzyme network	0.844	0.800	0.812	0.753

Interpretability: rules and clusters (an example with a protein-protein network)





Interpretability: feature ranking

Protein-protein interactions			Enzyme network		
#	Att.	Imp	#	Att.	Imp
1	loc - nucleolus	0.021	1	phy - dre	0.011
2	expr (Spell.) - elu 120	0.013	2	phy - rno	0.009
3	loc - cytoplasm	0.012	3	expr (Eisen) - cdc15 120m	0.008
4	expr (Eisen) - sporulation ndt80 early	0.012	4	phy - ecu	0.008
5	loc - nucleus	0.012	5	expr (Eisen) - cdc15 160m	0.008
6	expr (Eisen) - sporulation 30m	0.011	6	phy - pfa	0.007
7	expr (Eisen) - sporulation ndt80 middle	0.010	7	phy - mmu	0.007
8	expr (Spell.) - alpha 14	0.010	8	loc - cytoplasm	0.006
9	expr (Spell.) - elu 150	0.010	9	expr (Eisen) - cdc15 30m	0.005
10	loc - mitochondrion	0.009	10	expr (Eisen) - elutriation 5.5hrs	0.005

Conclusion

- We proposed a new method for the supervised inference of biological networks based on a kernelization of the output of tree-based methods
- We get competitive results with respect to full kernel based methods on two networks
- the method when used in a single tree can provide interpretable results in the form of a rule based clustering of the network
- the method when used in an ensemble of trees can provide a ranking of the features

- Future works:
  - Analysis of the role played by the output kernel
  - Develop other learning schemes with trees : boosting ...
  - (Biological) Analysis of results obtained with time series of gene expression of yeast stressed by irradiation (M. Dutreix, N. Touleimat)

Direct features from tree based methods/computational effi ciency

- Interpretability:
  - A single tree provides a rule-based model that is directly interpretable
  - Tree growing automatically selects the most relevant features (and hence is robust to irrelevant features)
  - A ranking of the features according to their relevance can be obtained by summing the total variance reduction over all nodes where the feature appears and normalizing over all variables

- Computational efficiency:
  - Learning stage: node splitting goes from $O(N)$ for standard trees to $O(N^2)$ for OK3
  - Prediction stage: kernel predictions are obtained in $O(N_L^2)$, where N_L^2 is the size of the leaf for a single tree and the support of $k_{ens}(v, .)$ for an ensemble (in practice $N_L << N$).