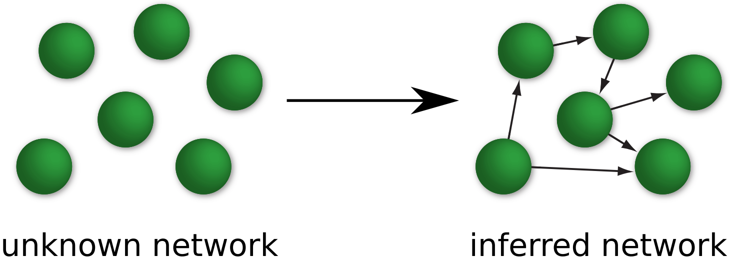
Combining tree-based and dynamical systems for the inference of gene regulatory networks

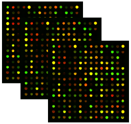
Vân Anh Huynh-Thu and Guido Sanguinetti

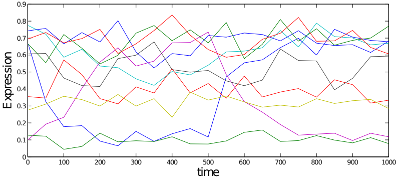
"Network Inference: New Methods and New Data"
3rd September, 2016

Inferring regulatory networks is a challenging problem

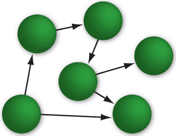


Expression data are used to infer networks





Network inference algorithm



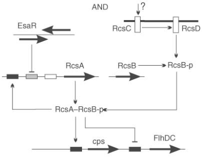
There are two main families of methods

Score-based: compute statistical dependencies between pairs of expression profiles (e.g. linear correlation)

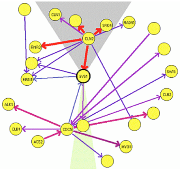
		Target gene			
		gene 1	**gene 2**	$\cdots$	**gene p**
Regulating gene	**gene 1**	-	0.05	$\cdots$	0.56
	gene 2	0.19	-	$\cdots$	0.03
	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	gene p	0.11	0.42	$\cdots$	-

$\rightarrow$ Fast, but can not make predictions

Model-based: learn a model capturing the dynamics of the network (e.g. differential equations)



(a)



$$\frac{dx_i}{dt} = m_i \cdot f_i(\mathbf{y}) - \lambda_i^{\mathrm{RNA}} \cdot x_i$$

$$\frac{dy_i}{dt} = r_i \cdot x_i - \lambda_i^{\mathrm{Prot}} \cdot y_i,$$

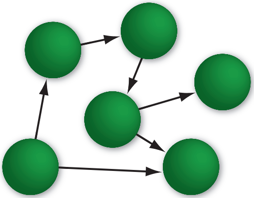
$\rightarrow$ Realistic, but are limited to small networks

Hybrid approach: Jump3

- Model for gene expression

- Tree-based method for network reconstruction

Results

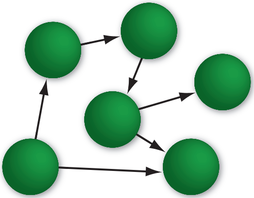


Hybrid approach: Jump3

- Model for gene expression

- Tree-based method for network reconstruction

Results

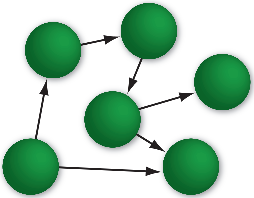


Hybrid approach: Jump3

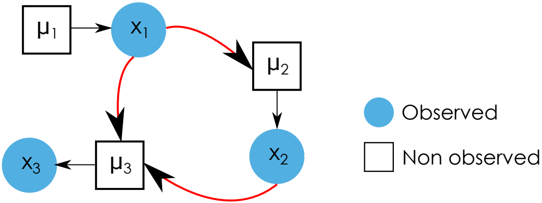
- Model for gene expression

- Tree-based method for network reconstruction

Results



We use the on/off model of gene expression

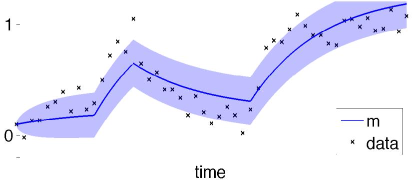


For each gene i:

$$\mathrm{d}x_i = (A_i \mu_i(t) + b_i - \lambda_i x_i)\mathrm{d}t + \sigma \mathrm{d}w(t)$$

$x_i(t)$: gene expression
$\mu_i(t)$: promoter activity state $(0/1)$
A_i, b_i, λ_i: kinetic parameters

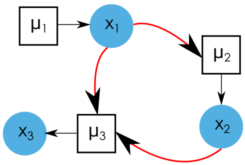
We model the expression x_i as a Gaussian process

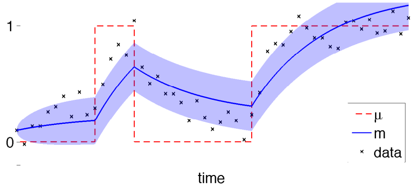


- x_i is completely described by its mean m_i and covariance K_i
- For every finite set of time points: $\mathbf{x}_i \sim \mathcal{N}(\mathbf{m}_i, K_i)$
- x_i is observed with i.i.d. Gaussian noise: $\hat{\mathbf{x}}_i \sim \mathcal{N}(\mathbf{m}_i, K_i + \sigma_{obs}^2 I)$
- We can compute the likelihood:

$$\log p(\hat{\mathbf{x}}_i) = -\frac{1}{2}(\hat{\mathbf{x}}_i - \mathbf{m}_i)^\top (K_i + \sigma_{obs}^2 I)^{-1} (\hat{\mathbf{x}}_i - \mathbf{m}_i) + c_i$$

The likelihood depends on the promoter state μ





Model: $$\mathrm{d}x_i = (A_i \mu_i(t) + b_i - \lambda_i x_i)\mathrm{d}t + \sigma \mathrm{d}w(t)$$

Likelihood:
$$\log p(\hat{\mathbf{x}}_i) = -\frac{1}{2}(\hat{\mathbf{x}}_i - \mathbf{m}_i)^\top (K_i + \sigma_{obs}^2 I)^{-1} (\hat{\mathbf{x}}_i - \mathbf{m}_i) + c_i$$

Goals (for each gene i):

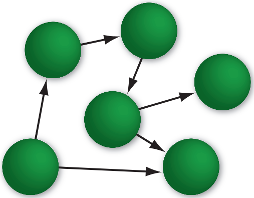
1. Find the trajectory μ_i that maximises the likelihood
2. Find the genes that influence μ_i (network reconstruction)

Hybrid approach: Jump3

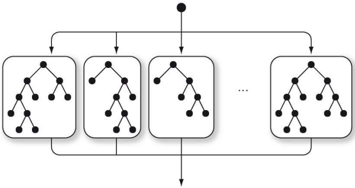
- Model for gene expression

- Tree-based method for network reconstruction

Results



Tree-based methods have several advantages



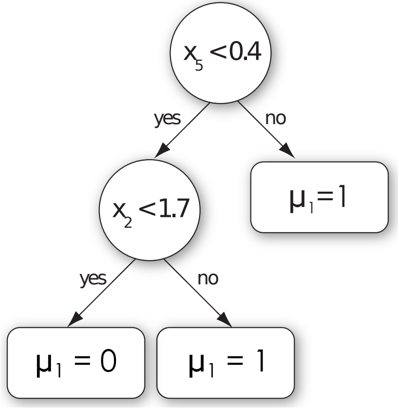
Bagging
Random Forests
Extra-Trees
...

Can deal with interacting features

Non-parametric

Work well with high-dimensional datasets

Decision trees are used to predict promoter states

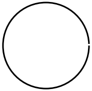


Each interior node tests the expression of a regulator.

Each leaf is a prediction of the promoter state of the target gene.

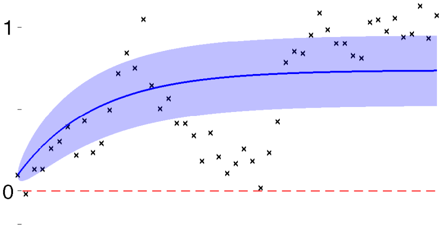
Promoter states are not observed.
$\rightarrow$ We can not use standard decision trees.

Jump trees are learned through maximisation of the likelihood

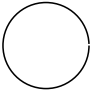


Start with $\mu_1(t) = 0, \forall t$

$\mathcal{L} = -2.56$

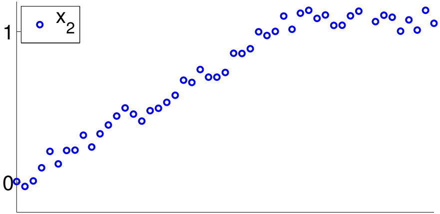


Jump trees are learned through maximisation of the likelihood

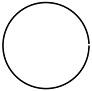


Candidate split:

$$\mu_1(t) = \begin{cases} 0, & \text{if } \hat{x}_2(t) < c \\ 1, & \text{if } \hat{x}_2(t) \geq c \end{cases}$$

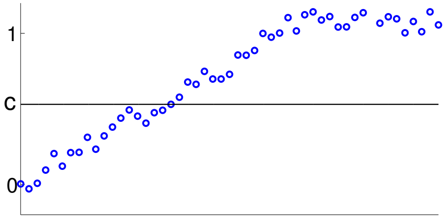


Jump trees are learned through maximisation of the likelihood

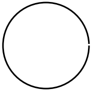


Candidate split:

$$\mu_1(t) = \begin{cases} 0, & \text{if } \hat{x}_2(t) < c \\ 1, & \text{if } \hat{x}_2(t) \geq c \end{cases}$$

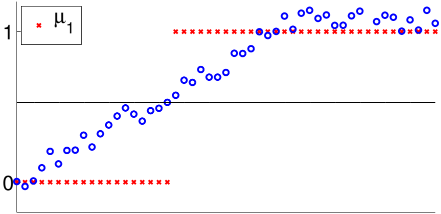


Jump trees are learned through maximisation of the likelihood

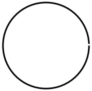


Candidate split:

$$\mu_1(t) = \begin{cases} 0, & \text{if } \hat{x}_2(t) < c \\ 1, & \text{if } \hat{x}_2(t) \geq c \end{cases}$$



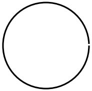
Jump trees are learned through maximisation of the likelihood



Candidate split:

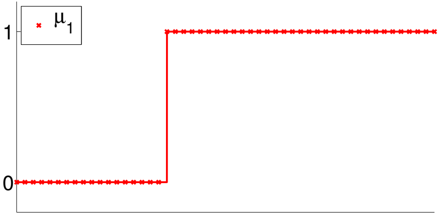
$$\mu_1(t) = \begin{cases} 0, & \text{if } \hat{x}_2(t) < c \\ 1, & \text{if } \hat{x}_2(t) \geq c \end{cases}$$

Jump trees are learned through maximisation of the likelihood

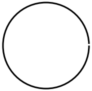


Candidate split:

$$\mu_1(t) = \begin{cases} 0, & \text{if } \hat{x}_2(t) < c \\ 1, & \text{if } \hat{x}_2(t) \geq c \end{cases}$$

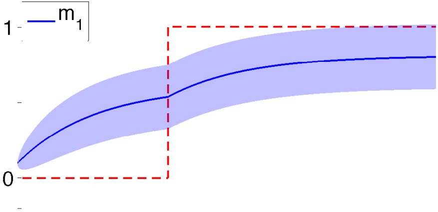


Jump trees are learned through maximisation of the likelihood

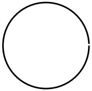


Candidate split:

$$\mu_1(t) = \begin{cases} 0, & \text{if } \hat{x}_2(t) < c \\ 1, & \text{if } \hat{x}_2(t) \geq c \end{cases}$$



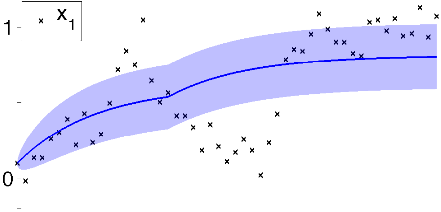
Jump trees are learned through maximisation of the likelihood



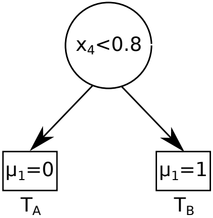
Candidate split:

$$\mu_1(t) = \begin{cases} 0, & \text{if } \hat{x}_2(t) < c \\ 1, & \text{if } \hat{x}_2(t) \geq c \end{cases}$$

$$\mathcal{L} = -2.35$$

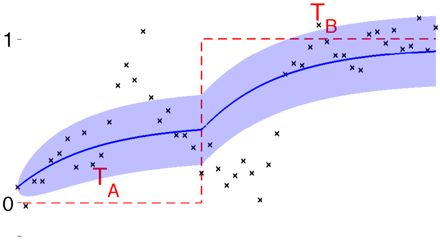


Jump trees are learned through maximisation of the likelihood

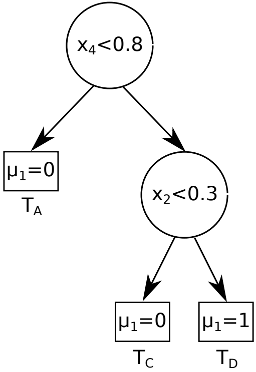


Select the split with the highest likelihood

$$\mathcal{L} = -1.39$$

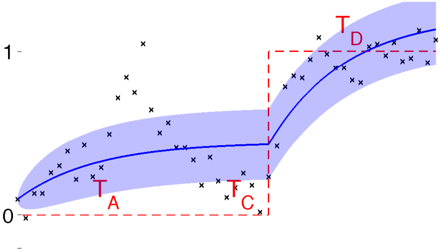


Jump trees are learned through maximisation of the likelihood

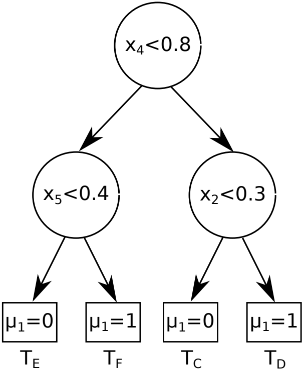


Repeat the procedure for each child node

$$\mathcal{L} = 0.47$$

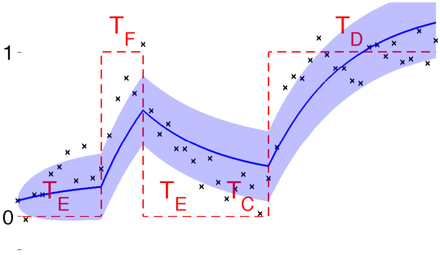


Jump trees are learned through maximisation of the likelihood

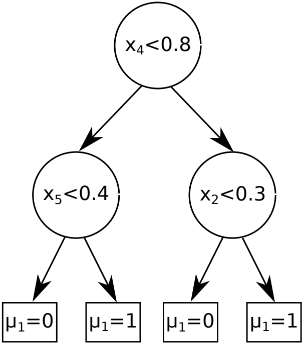


Repeat the procedure for each child node

$\mathcal{L} = 2.14$

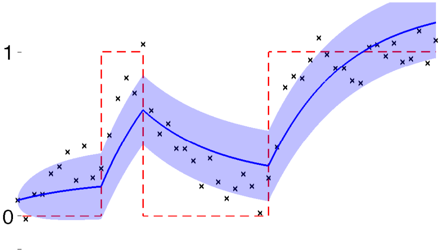


Jump trees are learned through maximisation of the likelihood

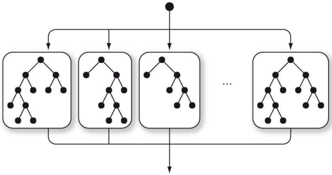


Stop when the likelihood can not be increased

$$\mathcal{L} = 2.14$$



An ensemble of randomised trees is constructed



Randomise x_i and c

$$\mu_1(t) = \begin{cases} 0, & \text{if } \hat{x}_i(t) < c \\ 1, & \text{if } \hat{x}_i(t) \geq c \end{cases}$$

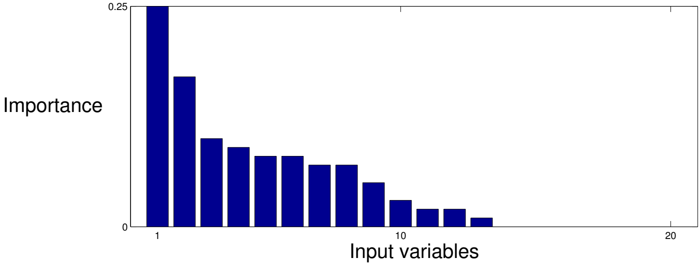
Extra-Trees (Geurts *et al.*, *Machine Learning*, 2006):

- At each node, the best split is chosen among K random splits.

- The prediction of $\mu(t)$ is averaged over the trees.

The tree-based model is informative

The learned model can be used to find the most relevant inputs.



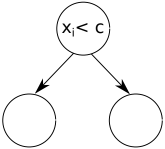
The variable importance is based on likelihood increase

At each tree node $\mathcal{N}$:

$$I(\mathcal{N}) = \mathcal{L}_{\text{after}} - \mathcal{L}_{\text{before}}$$

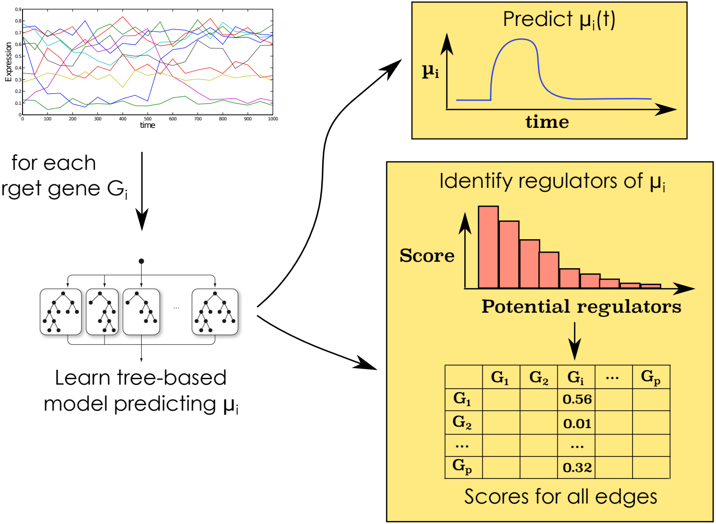
$\mathcal{L}_{\text{after}}$: likelihood after the split
$\mathcal{L}_{\text{before}}$: likelihood before the split

Importance of regulator x_i:
sum of I values over the nodes where x_i appears



Weight of edge gene $i \rightarrow$ gene j:
importance of x_i in the model predicting μ_j

Jump3 predicts the states and the network topology

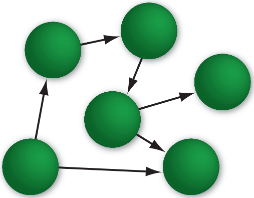


Hybrid approach: Jump3

- Model for gene expression

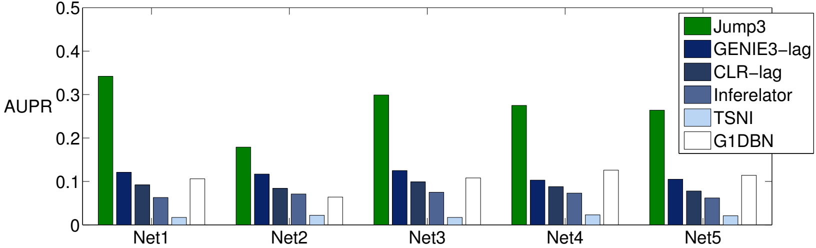
- Tree-based method for network reconstruction

Results

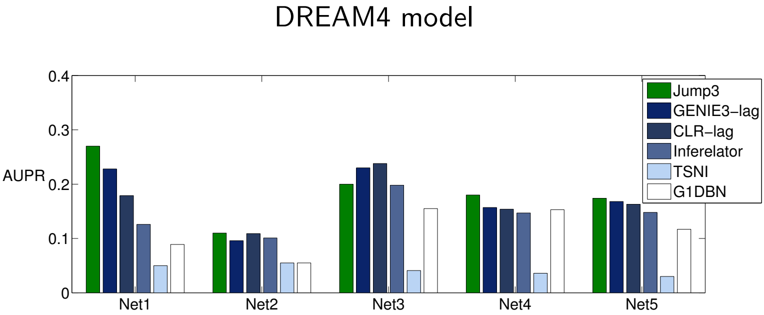


Jump3 is competitive with existing methods

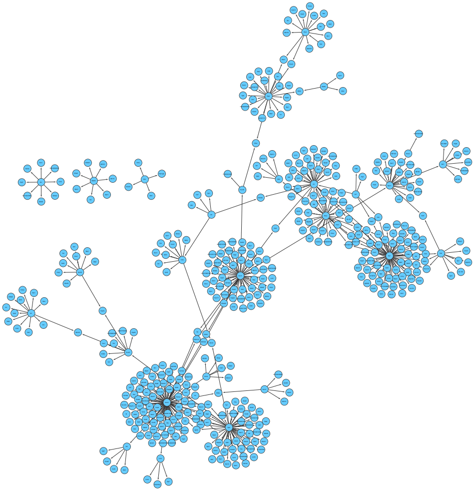
On/off model



Jump3 is competitive with existing methods



We used Jump3 to infer the IFNγ network



Hubs TFs contain interferon genes, one gene associated with virus infection, and cancer-associated genes.

Summary and future work

Summary

Jump3: Semi-parametric model-based method
for network inference and modelling

Can be applied to large-scale networks

Yields good performances on artificial data

Can generate biologically meaningful hypotheses

Future work

Incorporation of model-based prior knowledge (i.e. dynamical parametric model) within tree-based model.

References

V. A. Huynh-Thu and G. Sanguinetti.
Combining tree-based and dynamical systems for the inference of gene regulatory networks.
Bioinformatics 31, 2015.

Software:
http://www.montefiore.ulg.ac.be/~huynh-thu/software.html

Mean and variance of the Gaussian process

SDE:

$$\mathrm{d}x = (A\mu(t) + b - \lambda x)\mathrm{d}t + \sigma \mathrm{d}w(t)$$

Solution:

$$x(t) = x(0)e^{-\lambda t} + A\int_0^t e^{-\lambda(t-\tau)}\mu(\tau)d\tau + \frac{b}{\lambda}(1 - e^{-\lambda t}) + \sigma\int_0^t e^{-\lambda(t-\tau)}dw(\tau)$$

Mean:

$$m(t) = x(0)e^{-\lambda t} + A\int_0^t e^{-\lambda(t-\tau)}\mu(\tau)d\tau + \frac{b}{\lambda}(1 - e^{-\lambda t})$$

Covariance:

$$\mathrm{Cov}(x(t), x(t')) = \frac{\sigma^2}{2\lambda}(e^{-\lambda|t-t'|} - e^{-\lambda(t+t')})$$

Normalisation

For a single tree:

$$\sum_{i \neq j} w_{i \to j} = \mathcal{L}_{\text{fin}} - \mathcal{L}_{\text{init}}$$

$w_{i \to j}$: importance of gene i for the prediction of gene j
$\mathcal{L}_{\text{init}}$: likelihood when $\mu_j(t) = 0, \forall t$
$\mathcal{L}_{\text{fin}}$: likelihood with learned $\mu_j(t)$

$$\downarrow$$

Positive bias for edges towards genes for which $\mathcal{L}_{\text{fin}} - \mathcal{L}_{\text{init}}$ is high

$$\downarrow$$

Normalisation:

$$\frac{w_{i \to j}}{\mathcal{L}_{\text{fin}} - \mathcal{L}_{\text{init}}}$$