# Lecture 2:
# Inference of missing edges in biological networks

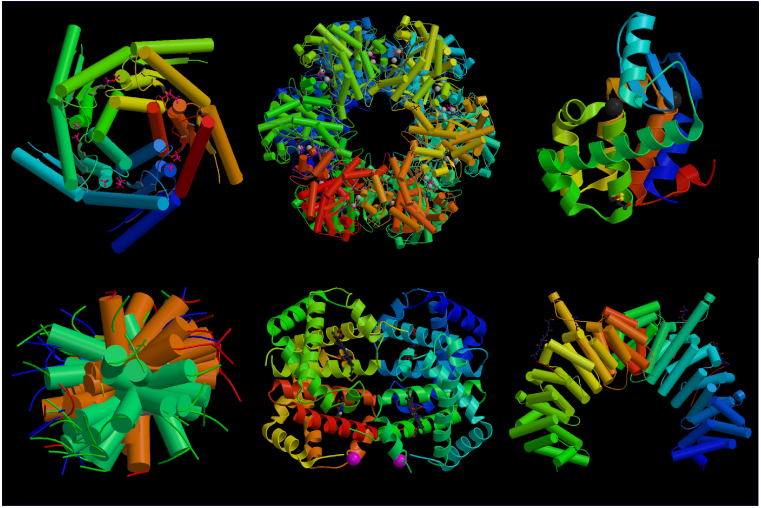
Jean-Philippe Vert

Mines ParisTech / Curie Institute / Inserm
Paris, France

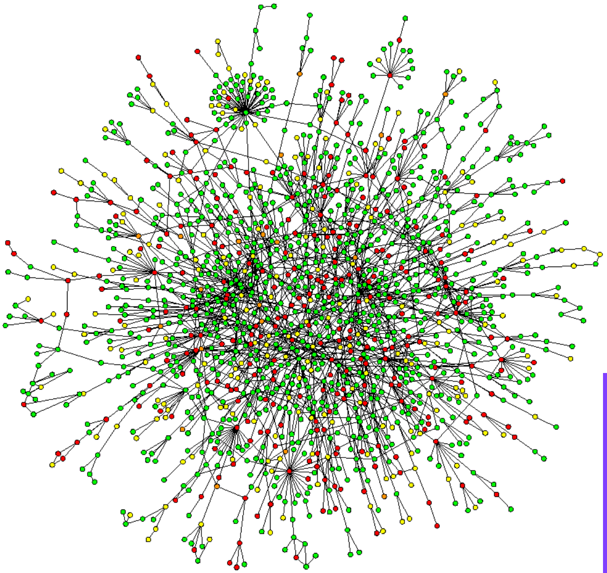
"Optimization, machine learning and bioinformatics" summer school, Erice, Sep 9-16, 2010.

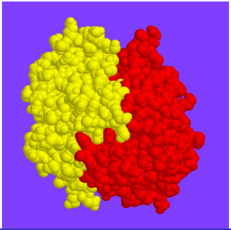
# Outline

# Proteins

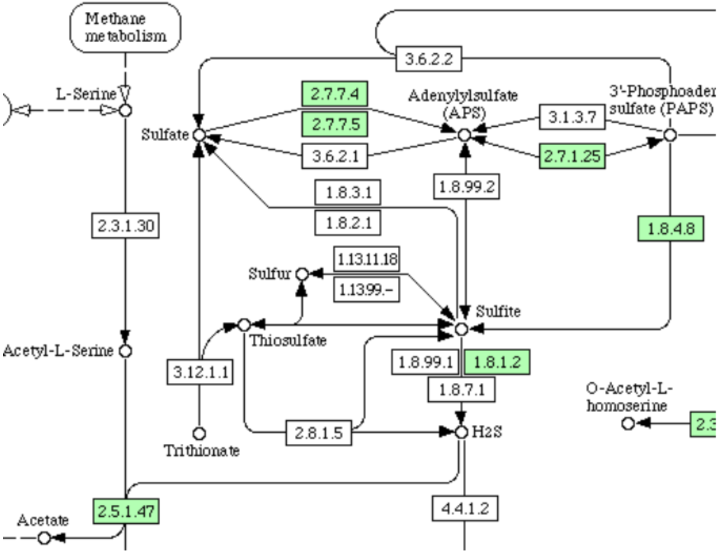


# Network 1: protein-protein interaction

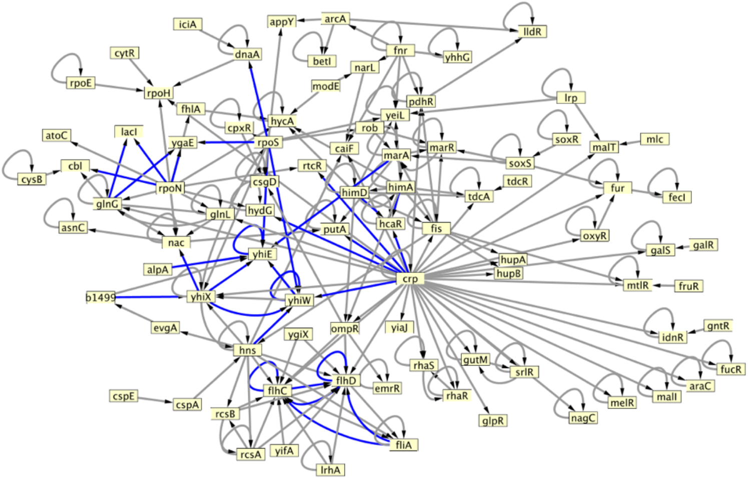




# Network 2: metabolic network



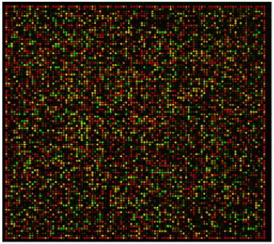
# Network 3: gene regulatory network



# Data available

*Biologists* have collected a lot of data about proteins. e.g.,

- Gene expression measurements
- Phylogenetic profiles
- Location of proteins/enzymes in the cell



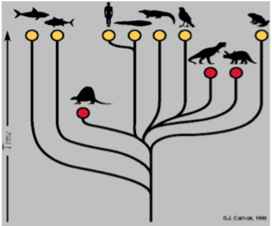
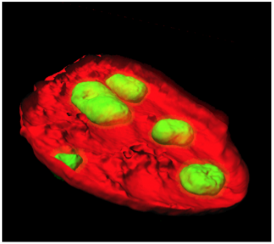
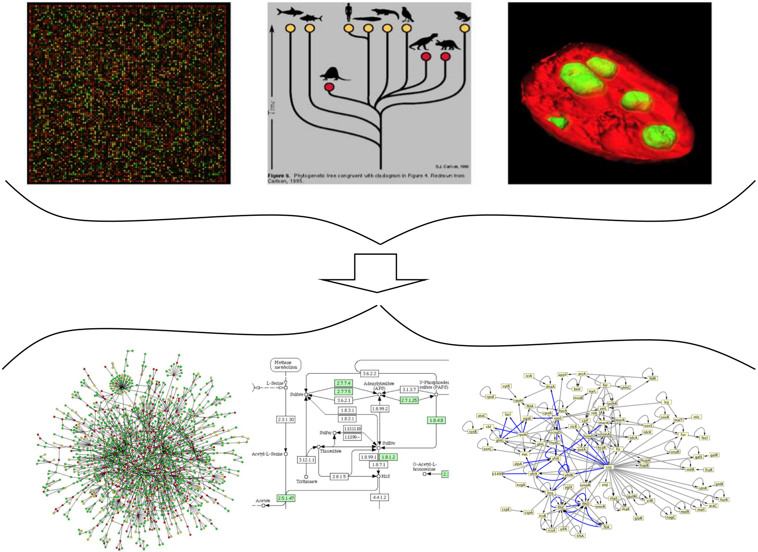



**Figure 5.** Phylogenetic tree congruent with cladogram in Figure 4. Redrawn from Carlson, 1995.



How to use this information “intelligently” to find a good function that predicts edges between nodes.

# Our goal


G.J. Carlson, 1999
Figure 5. Phylogenetic tree congruent with cladogram in Figure 4. Redrawn from Carlson, 1995.

# More precisely

## Formalization

- $\mathcal{V} = \{1, \ldots, N\}$ vertices (*e.g., genes, proteins*)
- $\mathcal{D} = (x_1, \ldots, x_N) \in \mathcal{H}^N$ data about the vertices ($\mathcal{H}$ *Hilbert space*)
- Goal: predict edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$.

## "De novo" inference

- Given data about individual genes and proteins $\mathcal{D}$, ...
- ... Infer the edges between genes and proteins $\mathcal{E}$

## "Supervised" inference

- Given data about individual genes and proteins $\mathcal{D}$, ...
- ... and given some known interactions $\mathcal{E}_{train} \subset \mathcal{E}$, ...
- ... infer unknown interactions $\mathcal{E}_{test} = \mathcal{E} \backslash \mathcal{E}_{train}$

# More precisely

## Formalization

- $\mathcal{V} = \{1, \ldots, N\}$ vertices (*e.g., genes, proteins*)
- $\mathcal{D} = (x_1, \ldots, x_N) \in \mathcal{H}^N$ data about the vertices ($\mathcal{H}$ *Hilbert space*)
- Goal: predict edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$.

## "De novo" inference

- Given data about individual genes and proteins $\mathcal{D}$, ...
- ... Infer the edges between genes and proteins $\mathcal{E}$

## "Supervised" inference

- Given data about individual genes and proteins $\mathcal{D}$, ...
- ... and given some known interactions $\mathcal{E}_{train} \subset \mathcal{E}$, ...
- ... infer unknown interactions $\mathcal{E}_{test} = \mathcal{E} \backslash \mathcal{E}_{train}$

# More precisely

## Formalization

- $\mathcal{V} = \{1, \ldots, N\}$ vertices (*e.g., genes, proteins*)
- $\mathcal{D} = (x_1, \ldots, x_N) \in \mathcal{H}^N$ data about the vertices ($\mathcal{H}$ *Hilbert space*)
- Goal: predict edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$.

## "De novo" inference

- Given data about individual genes and proteins $\mathcal{D}$, ...
- ... Infer the edges between genes and proteins $\mathcal{E}$

## "Supervised" inference

- Given data about individual genes and proteins $\mathcal{D}$, ...
- ... and given some known interactions $\mathcal{E}_{train} \subset \mathcal{E}$, ...
- ... infer unknown interactions $\mathcal{E}_{test} = \mathcal{E} \backslash \mathcal{E}_{train}$

# Outline

1. Introduction
2. De novo vs supervised methods
3. Supervised methods for pairs
4. Learning with local models
5. From local models to pairwise kernels
6. Experiments
7. Conclusion

# De novo methods

## Typical strategies

- Fit a dynamical system to time series (e.g., PDE, boolean networks, state-space models)
- Detect statistical conditional independence or dependency (Bayesian netwok, mutual information networks, co-expression)

## Pros

- Excellent approach if the model is correct and enough data are available
- Interpretability of the model
- Inclusion of prior knowledge

## Cons

- Specific to particular data and networks
- Needs a correct model!
- Difficult integration of heterogeneous data
- Often needs a lot of data and long computation time

# De novo methods

## Typical strategies

- Fit a dynamical system to time series (e.g., PDE, boolean networks, state-space models)
- Detect statistical conditional independence or dependency (Bayesian netwok, mutual information networks, co-expression)

## Pros

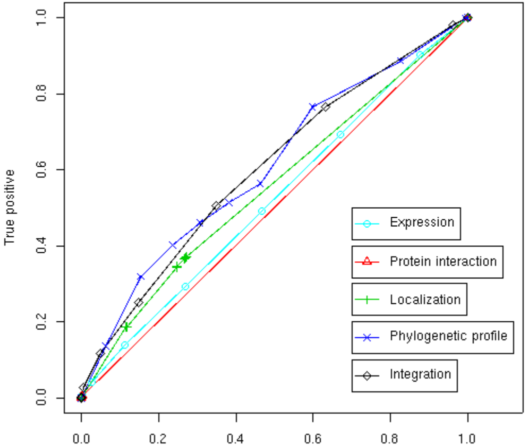
- Excellent approach if the model is correct and enough data are available
- Interpretability of the model
- Inclusion of prior knowledge

## Cons

- Specific to particular data and networks
- Needs a correct model!
- Difficult integration of heterogeneous data
- Often needs a lot of data and long computation time

# Evaluation on metabolic network reconstruction

- The known metabolic network of the yeast involves 769 proteins.
- Predict edges from distances between a variety of genomic data (expression, localization, phylogenetic profiles, interactions).

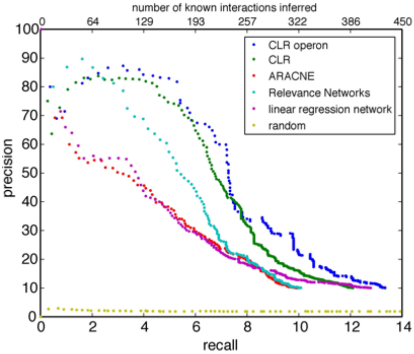


# Evaluation on regulatory network reconstruction

OPEN ACCESS Freely available online PLoS BIOLOGY

## Large-Scale Mapping and Validation of *Escherichia coli* Transcriptional Regulation from a Compendium of Expression Profiles

Jeremiah J. Faith[1☯], Boris Hayete[1☯], Joshua T. Thaden[2,3], Ilaria Mogno[2,4], Jamey Wierzbowski[2,5], Guillaume Cottarel[2,5], Simon Kasif[1,2], James J. Collins[1,2], Timothy S. Gardner[1,2*]

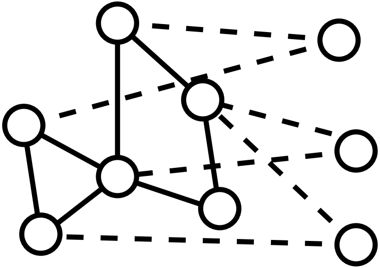


# Supervised methods

## Motivation

In actual applications,

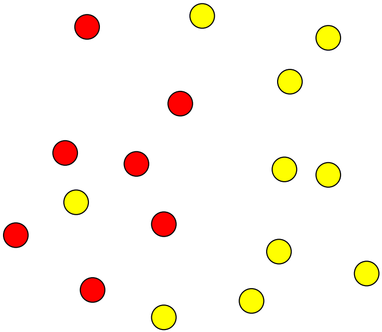
- we know in advance parts of the network to be inferred
- the problem is to add/remove nodes and edges using genomic data as side information



## Supervised method

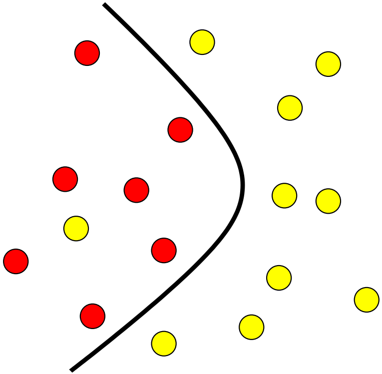
- Given genomic data and the currently known network...
- Infer missing edges between current nodes and additional nodes.

# Pattern recognition



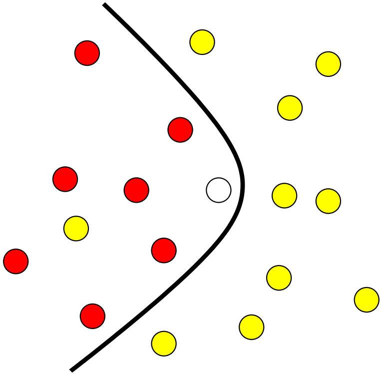
- Given a training set of patterns in two classes, learn to discriminate them
- Many algorithms (ANN, SVM, Decision tress, ...)

# Pattern recognition



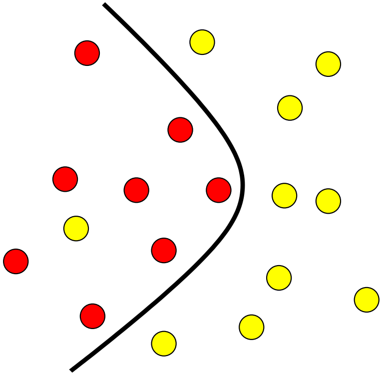
- Given a training set of patterns in two classes, learn to discriminate them
- Many algorithms (ANN, SVM, Decision tress, ...)

# Pattern recognition



- Given a training set of patterns in two classes, learn to discriminate them
- Many algorithms (ANN, SVM, Decision tress, ...)

# Pattern recognition



- Given a training set of patterns in two classes, learn to discriminate them
- Many algorithms (ANN, SVM, Decision tress, ...)

# Pattern recognition and graph inference

## Pattern recognition

Associate a binary label $Y$ to each data $X$

## Graph inference

Associate a binary label $Y$ to each pair of data $(X_1, X_2)$

## Two solutions

- Consider each pair $(X_1, X_2)$ as a single data -> learning over pairs
- Reformulate the graph inference problem as a pattern recognition problem at the level of individual vertices -> local models

# Pattern recognition and graph inference

## Pattern recognition

Associate a binary label $Y$ to each data $X$

## Graph inference

Associate a binary label $Y$ to each pair of data $(X_1, X_2)$

## Two solutions

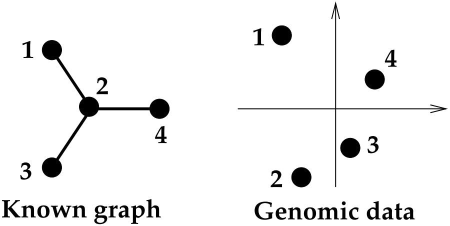
- Consider each pair $(X_1, X_2)$ as a single data -> learning over pairs
- Reformulate the graph inference problem as a pattern recognition problem at the level of individual vertices -> local models

# Outline

1. Introduction
2. De novo vs supervised methods
3. **Supervised methods for pairs**
4. Learning with local models
5. From local models to pairwise kernels
6. Experiments
7. Conclusion

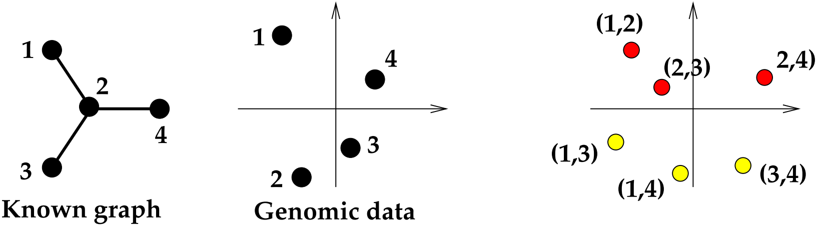
# Pattern recognition for pairs: basic issue

- A pair can be connected (1) or not connected (-1)
- From the known subgraph we can extract examples of connected and non-connected pairs
- However the genomic data characterize individual proteins; we need to work with pairs of proteins instead!



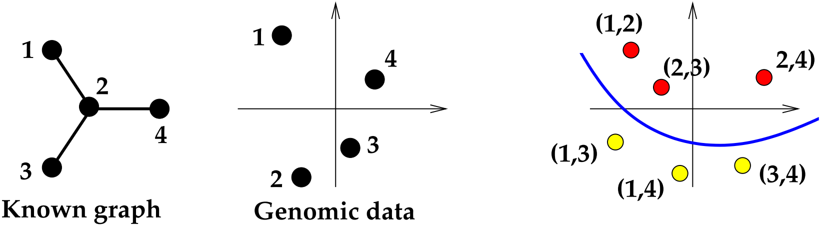
# Pattern recognition for pairs: basic issue

- A pair can be connected (1) or not connected (-1)
- From the known subgraph we can extract examples of connected and non-connected pairs
- However the genomic data characterize individual proteins; we need to work with pairs of proteins instead!



# Pattern recognition for pairs: basic issue

- A pair can be connected (1) or not connected (-1)
- From the known subgraph we can extract examples of connected and non-connected pairs
- However the genomic data characterize individual proteins; we need to work with pairs of proteins instead!



# Representing a pair as a vector

- Each individual protein is represented by a vector $v \in \mathbb{R}^p$
- Depending on the network, we are interested in ordered or unordered pairs of proteins.
- We must represent a pair of proteins $(u, v)$ by a vector $\psi(u, v) \in \mathbb{R}^q$ in order to estimate a linear classifier
- Question: how build $\psi(u, v)$ from $u$ and $v$, in the ordered and unordered cases?

# Direct sum for ordered pairs?

- A simple idea is to concatenate the vectors $u$ and $v$ to obtain a $2p$-dimensional vector of $(u, v)$:

$$\psi(u, v) = u \oplus v = \begin{pmatrix} u \\ v \end{pmatrix}.$$

- Problem: a linear function then becomes additive...

$$f(u, v) = w^\top \psi(u, v) = w_1^\top u + w^\top v.$$

# Direct sum for ordered pairs?

- A simple idea is to concatenate the vectors $u$ and $v$ to obtain a $2p$-dimensional vector of $(u, v)$:

$$\psi(u, v) = u \oplus v = \begin{pmatrix} u \\ v \end{pmatrix} .$$

- Problem: a linear function then becomes additive...

$$f(u, v) = w^\top \psi(u, v) = w_1^\top u + w^\top v .$$

# Direct product for ordered pairs

- Alternatively, make the direct product, i.e., the $p^2$-dimensional vector whose entries are all products of entries of $u$ by entries of $v$:

$$\psi(u, v) = u \otimes v$$

- Problem: can get really large-dimensional...
- Good news: inner product factorizes:

$$(u_1 \otimes v_1)^\top (u_2 \otimes v_2) = \left(u_1^\top u_2\right) \times \left(v_1^\top v_2\right) ,$$

which is good for algorithms that use only inner products (SVM...):

$$K_P((u_1, v_1), (u_2, v_2)) = \psi(u_1, v_1)^\top \psi(u_2, v_2) = K(u_1, u_2)K(v_1, v_2)$$

# Direct product for ordered pairs

- Alternatively, make the direct product, i.e., the $p^2$-dimensional vector whose entries are all products of entries of $u$ by entries of $v$:

$$\psi(u, v) = u \otimes v$$

- Problem: can get really large-dimensional...
- Good news: inner product factorizes:

$$(u_1 \otimes v_1)^\top (u_2 \otimes v_2) = \left(u_1^\top u_2\right) \times \left(v_1^\top v_2\right) ,$$

which is good for algorithms that use only inner products (SVM...):

$$K_P\left((u_1, v_1), (u_2, v_2)\right) = \psi(u_1, v_1)^\top \psi(u_2, v_2) = K(u_1, u_2)K(v_1, v_2)$$

# Direct product for ordered pairs

- Alternatively, make the direct product, i.e., the $p^2$-dimensional vector whose entries are all products of entries of $u$ by entries of $v$:

$$\psi(u, v) = u \otimes v$$

- Problem: can get really large-dimensional...
- Good news: inner product factorizes:

$$(u_1 \otimes v_1)^\top (u_2 \otimes v_2) = \left(u_1^\top u_2\right) \times \left(v_1^\top v_2\right) ,$$

which is good for algorithms that use only inner products (SVM...):

$$K_P((u_1, v_1), (u_2, v_2)) = \psi(u_1, v_1)^\top \psi(u_2, v_2) = K(u_1, u_2) K(v_1, v_2)$$

# Representing an unordered pair

- Often we want to work with unordered pairs, e.g., PPI network:

$$\{u, v\} = \{(u, v), (v, u)\}$$

- This suggest to symmetrize the representation of ordered pairs:

$$\psi_U(\{u, v\}) = \psi(u, v) + \psi(v, u)$$

- When $\psi(u, v) = u \otimes v$, this leads to the symmetric tensor product pairwise kernel (TPPK) (Ben-Hur and Noble, 2006):

$$K_{TPPK}\left(\{u_1, v_1\}, \{u_2, v_2\}\right) = K(u_1, u_2)K(v_1, v_2) + K(u_1, v_2)K(v_1, u_2)$$

# Representing an unordered pair

- Often we want to work with unordered pairs, e.g., PPI network:

$$\{u, v\} = \{(u, v), (v, u)\}$$

- This suggest to symmetrize the representation of ordered pairs:

$$\psi_U(\{u, v\}) = \psi(u, v) + \psi(v, u)$$

- When $\psi(u, v) = u \otimes v$, this leads to the symmetric tensor product pairwise kernel (TPPK) (Ben-Hur and Noble, 2006):

$$K_{TPPK}(\{u_1, v_1\}, \{u_2, v_2\}) = K(u_1, u_2)K(v_1, v_2) + K(u_1, v_2)K(v_1, u_2)$$

# Representing an unordered pair

- Often we want to work with unordered pairs, e.g., PPI network:

$$\{u, v\} = \{(u, v), (v, u)\}$$

- This suggest to symmetrize the representation of ordered pairs:

$$\psi_U(\{u, v\}) = \psi(u, v) + \psi(v, u)$$

- When $\psi(u, v) = u \otimes v$, this leads to the symmetric tensor product pairwise kernel (TPPK) (Ben-Hur and Noble, 2006):

$$K_{TPPK}(\{u_1, v_1\}, \{u_2, v_2\}) = K(u_1, u_2)K(v_1, v_2) + K(u_1, v_2)K(v_1, u_2)$$

# Another idea: metric learning

- For two vectors $u, v \in \mathcal{H}$ let the metric:

$$d_M(u, v) = (u - v)^\top M (u - v) \, .$$

- Can we learn the metric $M$ such that, in the new metric, connected points are near each other, and non-connected points are far from each other?
- We consider the problem:

$$\min_{M \geq 0} \sum_i l(u_i, v_i, y_i) + \lambda ||M||^2_{Frobenius} \, ,$$

where $l$ is a *hinge loss* to enforce:

$$d_M(u_i, v_i) \begin{cases} \leq 1 - \gamma & \text{if}(u_i, v_i)\text{is connected} \, , \\ \geq 1 + \gamma & \text{otherwise.} \end{cases}$$

# Another idea: metric learning

- For two vectors $u, v \in \mathcal{H}$ let the metric:

$$d_M(u, v) = (u - v)^\top M (u - v) \, .$$

- Can we learn the metric $M$ such that, in the new metric, connected points are near each other, and non-connected points are far from each other?
- We consider the problem:

$$\min_{M \geq 0} \sum_i l(u_i, v_i, y_i) + \lambda ||M||^2_{Frobenius} \, ,$$

where $l$ is a *hinge loss* to enforce:

$$d_M(u_i, v_i) \begin{cases} \leq 1 - \gamma & \text{if}(u_i, v_i)\text{is connected} \, , \\ \geq 1 + \gamma & \text{otherwise.} \end{cases}$$

# Another idea: metric learning

- For two vectors $u, v \in \mathcal{H}$ let the metric:

$$d_M(u, v) = (u - v)^\top M(u - v)\,.$$

- Can we learn the metric $M$ such that, in the new metric, connected points are near each other, and non-connected points are far from each other?
- We consider the problem:

$$\min_{M \geq 0} \sum_i l(u_i, v_i, y_i) + \lambda ||M||^2_{Frobenius}\,,$$

where $l$ is a *hinge loss* to enforce:

$$d_M(u_i, v_i) \begin{cases} \leq 1 - \gamma & \text{if}(u_i, v_i)\text{is connected}\,, \\ \geq 1 + \gamma & \text{otherwise.} \end{cases}$$

# Link with metric learning

## Theorem (V. et al., 2007)

- A SVM with the representation

$$\psi(\{u, v\}) = (u - v)^{\otimes 2}$$

trained to discriminate connected from non-connected pairs, solves this metric learning problem without the constraint $M \geq 0$.

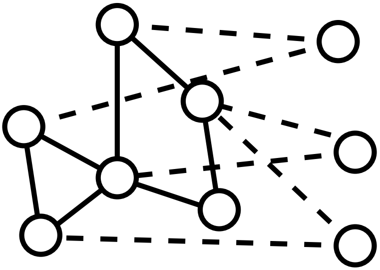
- Equivalently, train the SVM over pairs with the metric learning pairwise kernel:

$$\begin{aligned} K_{MLPK}\left(\{u_1, v_1\}, \{u_2, v_2\}\right) &= \psi(\{u_1, v_1\})^\top \psi(\{u_2, v_2\}) \\ &= \left[K(u_1, u_2) - K(u_1, v_2) - K(v_1, u_2) + K(u_2, v_2)\right]^2 . \end{aligned}$$

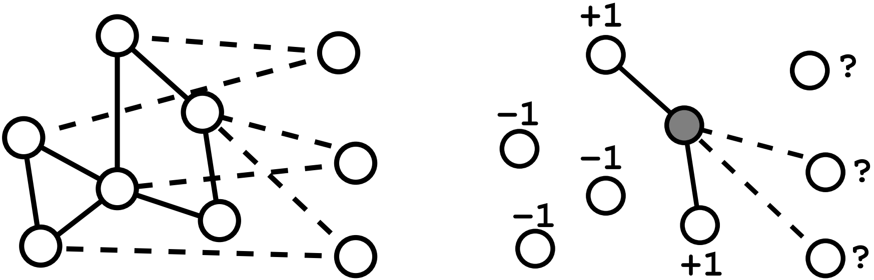
# Outline

# The idea (Bleakley et al., 2007)

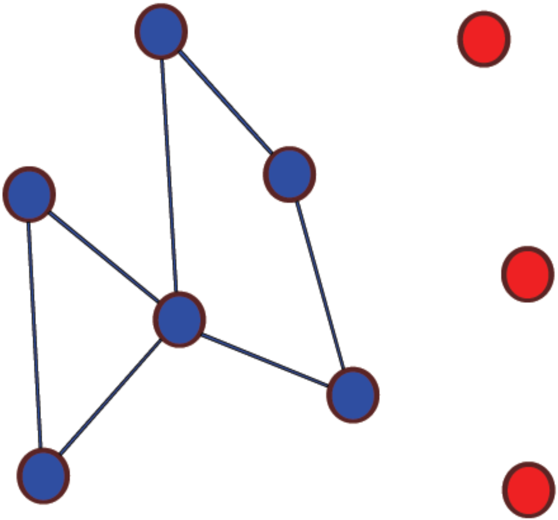
- Motivation: define specific models for each target node to discriminate between its neighbors and the others
- Treat each node independently from the other. Then combine predictions for ranking candidate edges.



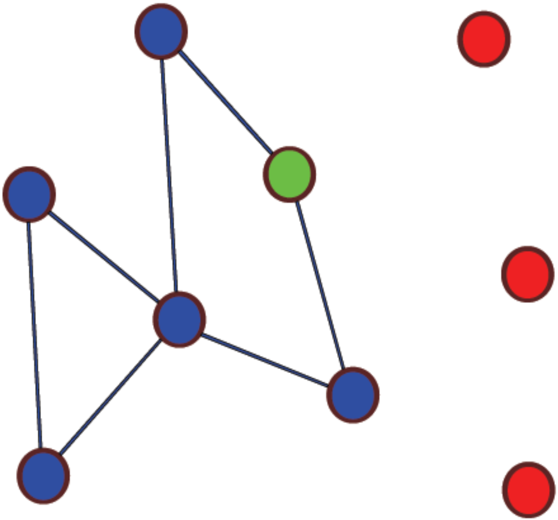
- Motivation: define specific models for each target node to discriminate between its neighbors and the others
- Treat each node independently from the other. Then combine predictions for ranking candidate edges.



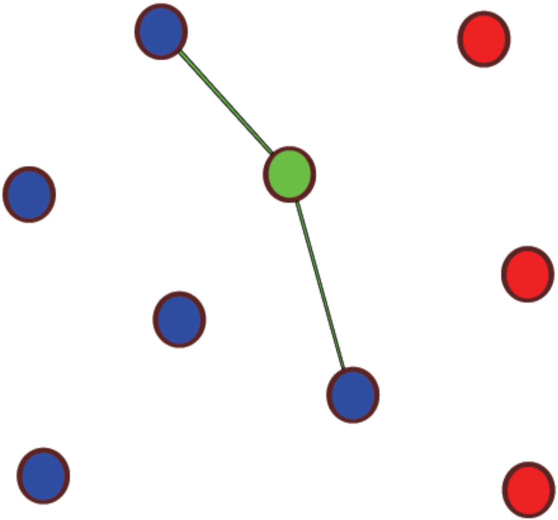
# The LOCAL model



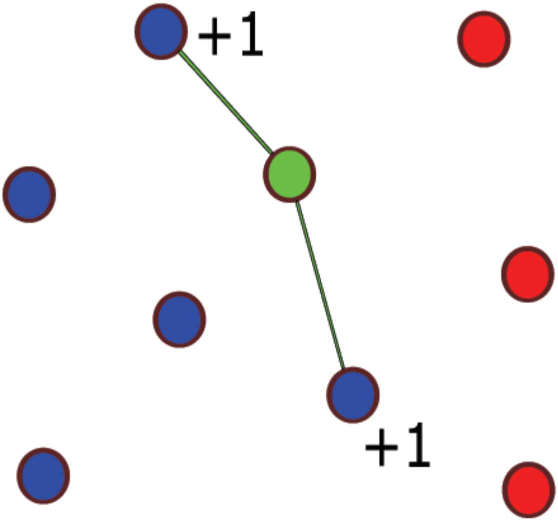
# The LOCAL model



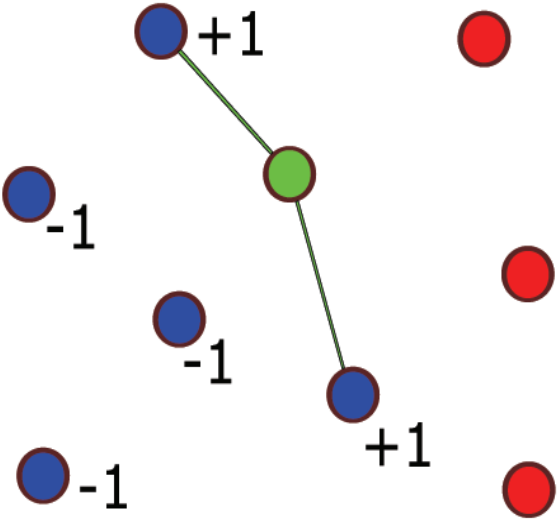
# The LOCAL model



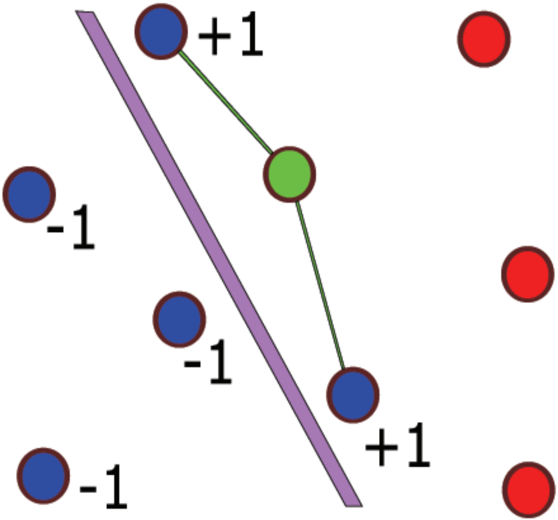
# The LOCAL model



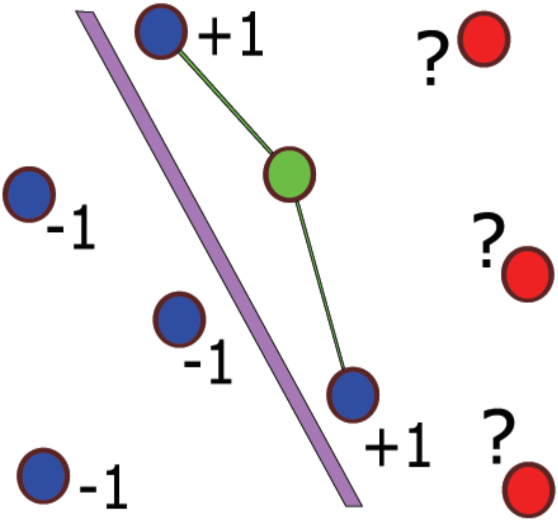
# The LOCAL model



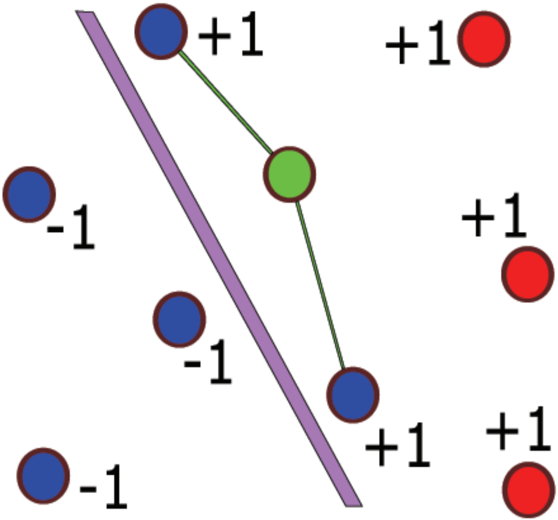
# The LOCAL model



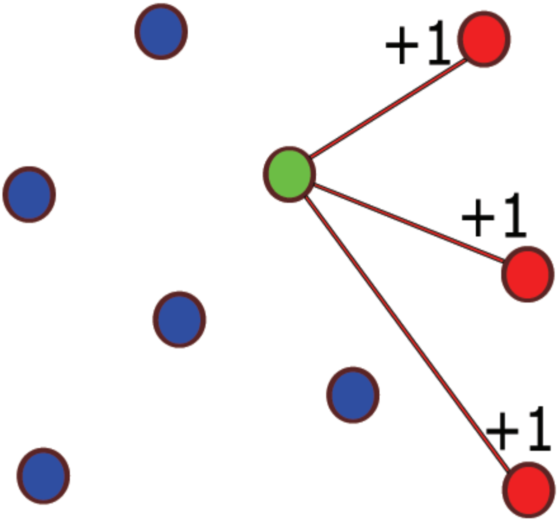
# The LOCAL model



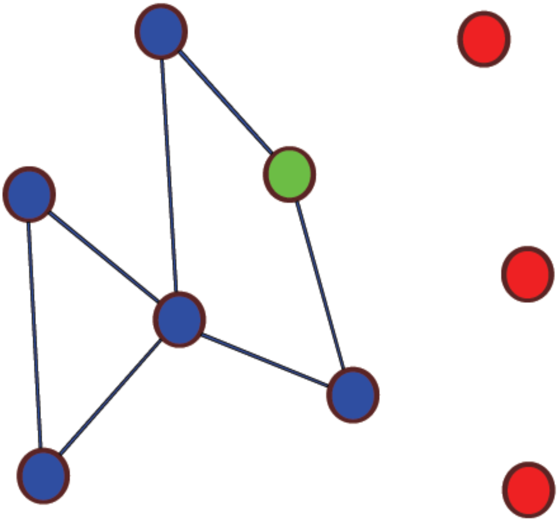
# The LOCAL model



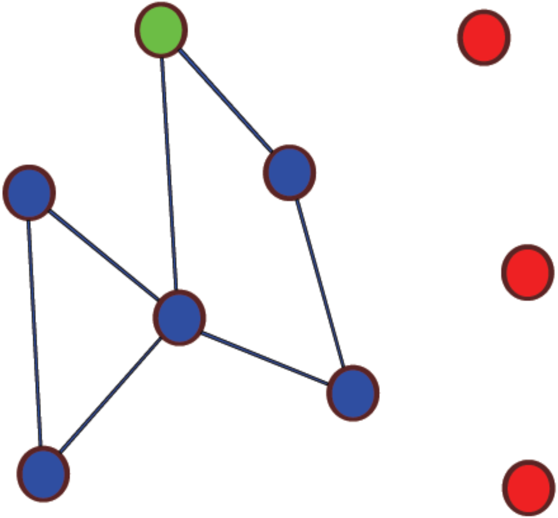
# The LOCAL model



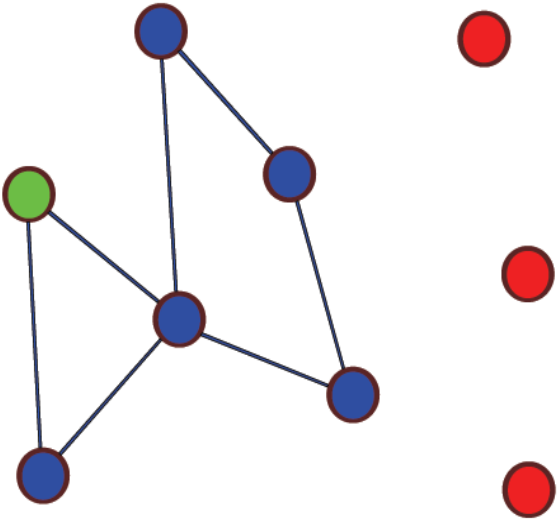
# The LOCAL model



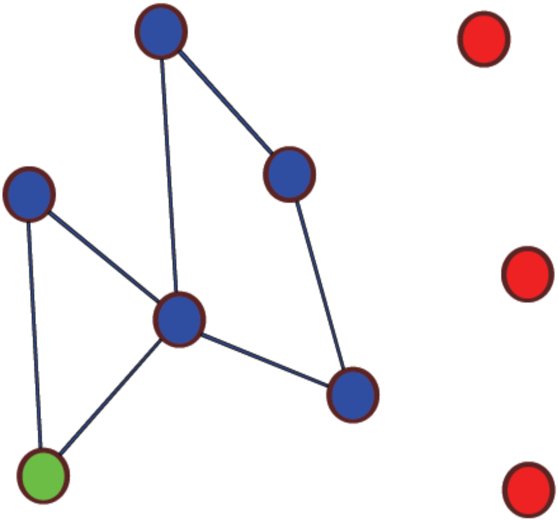
# The LOCAL model



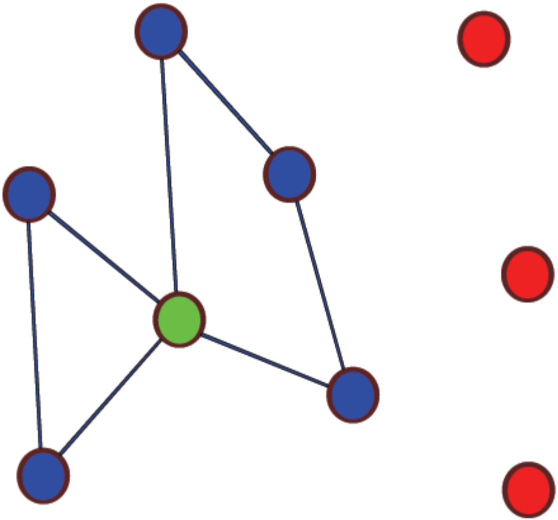
# The LOCAL model



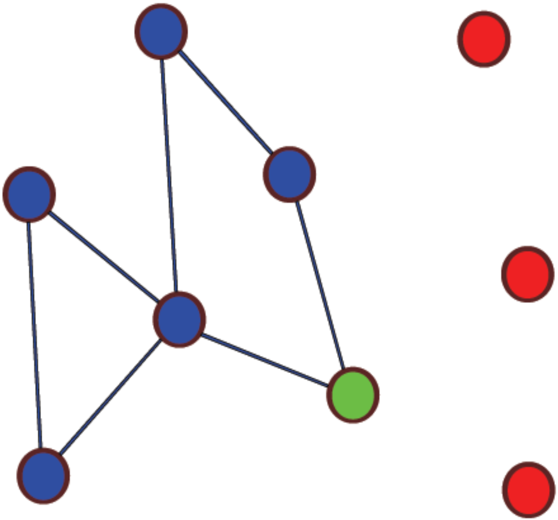
# The LOCAL model



# The LOCAL model



# The LOCAL model



# A few remarks

- In the case of unordered interactions, we need to symmetrize the prediction, typically by averaging the predictive scores of $A \to B$ and $B \to A$ to predict the interaction $\{A, B\}$
- Weak hypothesis:
  - if A is connected to B,
  - if C is similar to B,
  - then A is likely to be connected to C.
- Computationally: much faster to train $N$ local models with $N$ training points each, than to train 1 model with $N^2$ training points.
- Caveats:
  - each local model may have very few training points
  - no sharing of information between different local models

# A few remarks

- In the case of unordered interactions, we need to symmetrize the prediction, typically by averaging the predictive scores of $A \rightarrow B$ and $B \rightarrow A$ to predict the interaction $\{A, B\}$
- Weak hypothesis:
  - if A is connected to B,
  - if C is similar to B,
  - then A is likely to be connected to C.
- Computationally: much faster to train $N$ local models with $N$ training points each, than to train 1 model with $N^2$ training points.
- Caveats:
  - each local model may have very few training points
  - no sharing of information between different local models

# A few remarks

- In the case of unordered interactions, we need to symmetrize the prediction, typically by averaging the predictive scores of $A \to B$ and $B \to A$ to predict the interaction $\{A, B\}$
- Weak hypothesis:
  - if A is connected to B,
  - if C is similar to B,
  - then A is likely to be connected to C.
- Computationally: much faster to train $N$ local models with $N$ training points each, than to train 1 model with $N^2$ training points.
- Caveats:
  - each local model may have very few training points
  - no sharing of information between different local models

# A few remarks

- In the case of unordered interactions, we need to symmetrize the prediction, typically by averaging the predictive scores of $A \rightarrow B$ and $B \rightarrow A$ to predict the interaction $\{A, B\}$
- Weak hypothesis:
  - if A is connected to B,
  - if C is similar to B,
  - then A is likely to be connected to C.
- Computationally: much faster to train $N$ local models with $N$ training points each, than to train 1 model with $N^2$ training points.
- Caveats:
  - each local model may have very few training points
  - no sharing of information between different local models

# Outline

# Motivation

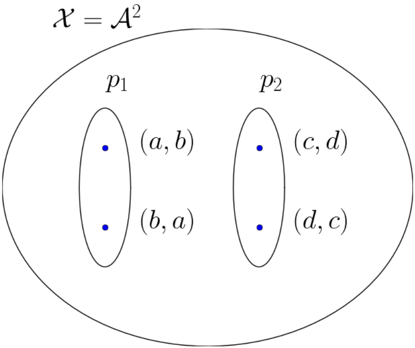
In the case of unordered pairs $\{A, B\}$, pairwise kernels such as the TPPK and local models look very different:

- Local models seem to over-emphasize the asymmetry of the relationships, but symmetrize the prediction *a posteriori*
- Pairwise kernels symmetrize the data *a priori* and learn in the space or unordered pairs

Can be clarify the links between these approaches, and perhaps interpolate between them?

# Notations

- $\mathcal{A}$ the set of individual proteins, endowed with a kernel $K_{\mathcal{A}}$
- $\mathcal{X} = \mathcal{A}^2$ the set of ordered pairs of the form $x = (a, b)$ endowed with a kernel $K_{\mathcal{X}}$ (usually deduced from $K_{\mathcal{A}}$)
- $\mathcal{P}$ the set of unordered pairs of the form $p = \{(a, b), (b, a)\}$
- We want to learn over $\mathcal{P}$ from a set of labeled training pairs $(p_1, y_1), \ldots, (p_n, y_n) \in \mathcal{P} \times \{-1, 1\}$



# Two strategies to learn over $\mathcal{P}$

## Strategy 1: Inference over $\mathcal{P}$ with a pair kernel

1. Define a kernel $K_{\mathcal{P}}$ over $\mathcal{P}$ by convolution of $K_{\mathcal{X}}$:

$$K_{\mathcal{P}}(p, p') = \frac{1}{|p| \cdot |p'|} \sum_{x \in p, x' \in p'} K_{\mathcal{X}}(x, x') \,.$$

2. Train a classifier over $\mathcal{P}$ e.g., a SVM, using the kernel $K_{\mathcal{P}}$

## Strategy 2: Inference over $\mathcal{X}$ with a pair duplication

1. Duplicate each training pair $p = \{a, b\}$ into 2 ordered paired
2. Train a classifier over $\mathcal{X}$, e.g., a SVM, using the kernel $K_{\mathcal{X}}$
3. The classifier over $\mathcal{P}$ is then the *a posteriori* average:

$$f_{\mathcal{P}}(p) = \frac{1}{|p|} \sum_{x \in p} f_{\mathcal{X}}(x)$$

# Two strategies to learn over $\mathcal{P}$

## Strategy 1: Inference over $\mathcal{P}$ with a pair kernel

1. Define a kernel $K_{\mathcal{P}}$ over $\mathcal{P}$ by convolution of $K_{\mathcal{X}}$:

$$K_{\mathcal{P}}(p, p') = \frac{1}{|p| \cdot |p'|} \sum_{x \in p, x' \in p'} K_{\mathcal{X}}(x, x') .$$

2. Train a classifier over $\mathcal{P}$ e.g., a SVM, using the kernel $K_{\mathcal{P}}$

## Strategy 2: Inference over $\mathcal{X}$ with a pair duplication

1. Duplicate each training pair $p = \{a, b\}$ into 2 ordered paired
2. Train a classifier over $\mathcal{X}$, e.g., a SVM, using the kernel $K_{\mathcal{X}}$
3. The classifier over $\mathcal{P}$ is then the *a posteriori* average:

$$f_{\mathcal{P}}(p) = \frac{1}{|p|} \sum_{x \in p} f_{\mathcal{X}}(x)$$

# The TPPK kernel

$$K_{TPPK}\left(\{a,b\},\{c,d\}\right) = K_{\mathcal{A}}(a,c)K_{\mathcal{A}}(b,d) + K_{\mathcal{A}}(a,d)K_{\mathcal{A}}(b,c)\,.$$

## Theorem

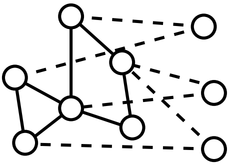
Let $\mathcal{X} = \mathcal{A}^2$ be endowed with the p.d. kernel:

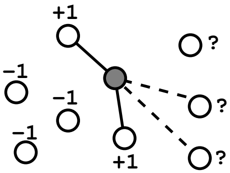
$$K_{\mathcal{X}}\left((a,b),(c,d)\right) = 2K_{\mathcal{A}}(a,c)K_{\mathcal{A}}(b,d)\,. \quad (1)$$

Then the TPPK approach is equivalent to both Strategy 1 and Strategy 2.

*Remarks: Equivalence with Strategy 1 is obvious, equivalence with Strategy 2 is not, see proof in Hue and V. (ICML 2010).*

# The local models






## Theorem

Let $\mathcal{X} = \mathcal{A}^2$ be endowed with the p.d. kernel:

$$K_{\mathcal{X}}\left((a, b), (c, d)\right) = \delta(a, c) K_{\mathcal{A}}(b, d)\,,$$

where $\delta$ is the Kronecker kernel ($\delta(a, c) = 1$ if $a = c$, 0 otherwise). Then the local approach is equivalent to Strategy 2.

*Remarks: Strategies 1 and 2 are not equivalent with this kernel. In general, they are equivalent up to a modification in the loss function of the learning algorithm, see details in Hue and V. (ICML 2010)..*

# Interpolation between local model and TPPK

| | Strategy 1: pair kernel | Strategy 2: duplication |
|---|---|---|
| $K_{\mathcal{X}} = K_{\mathcal{A}} \otimes K_{\mathcal{A}}$ | TPPK | TPPK |
| $K_{\mathcal{X}} = \delta \otimes K_{\mathcal{A}}$ | new | Local model |

Interpolation:

$$K_{\mathcal{X}} = ((1-\lambda)K_{\mathcal{A}} + \lambda\delta) \otimes K_{\mathcal{A}}$$

for $\lambda \in [0,1]$

# Interpolation between local model and TPPK

| | Strategy 1: pair kernel | Strategy 2: duplication |
|---|---|---|
| $K_{\mathcal{X}} = K_{\mathcal{A}} \otimes K_{\mathcal{A}}$ | TPPK | TPPK |
| $K_{\mathcal{X}} = \delta \otimes K_{\mathcal{A}}$ | new | Local model |

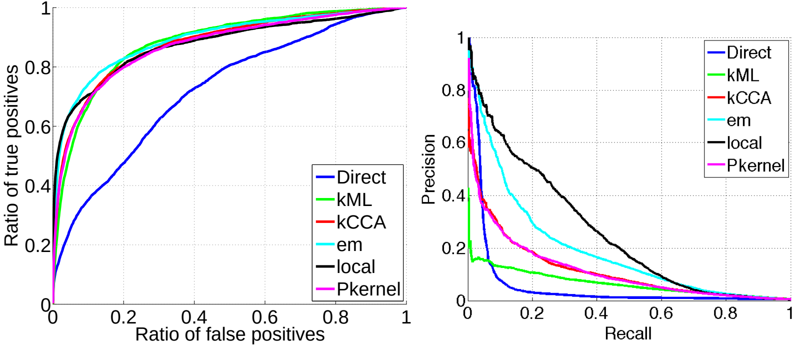
Interpolation:

$$K_{\mathcal{X}} = ((1 - \lambda)K_{\mathcal{A}} + \lambda\delta) \otimes K_{\mathcal{A}}$$

for $\lambda \in [0, 1]$

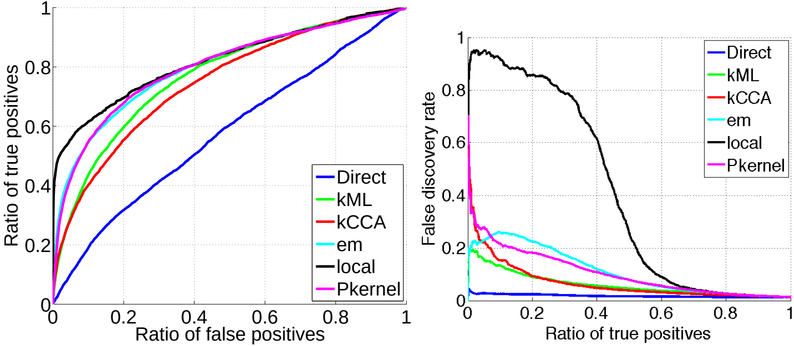
# Outline

# Results: protein-protein interaction (yeast)



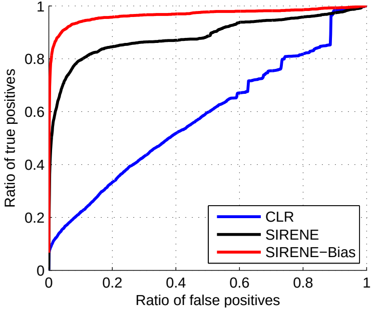

(from Bleakley et al., 2007)

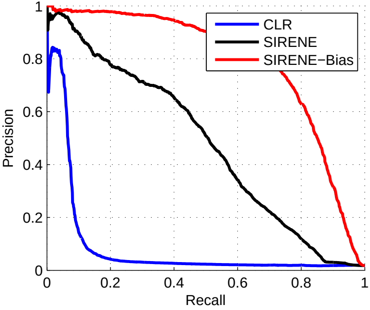
# Results: metabolic gene network (yeast)




(from Bleakley et al., 2007)

# Results: regulatory network (E. coli)




| Method | Recall at 60% | Recall at 80% |
|---|---|---|
| SIRENE | **44.5%** | **17.6%** |
| CLR | 7.5% | 5.5% |
| Relevance networks | 4.7% | 3.3% |
| ARACNe | 1% | 0% |
| Bayesian network | 1% | 0% |

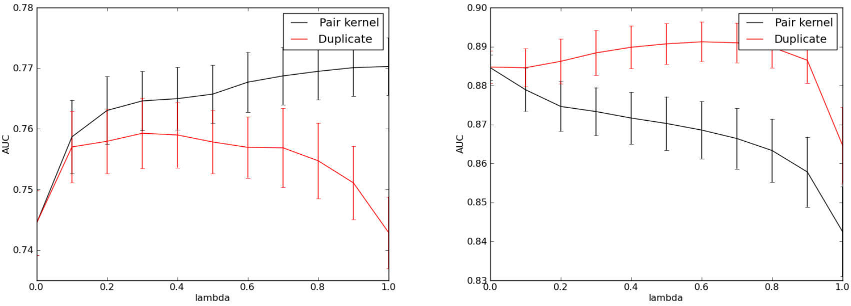
*SIRENE = Supervised Inference of REgulatory NEtworks (Mordelet and V., 2008)*

# Interpolation kernel

Table: Strategy and kernel realizing the maximum mean AUC for nine metabolic and protein-protein interaction networks experiments, with the kernel $K^\lambda$ for $\lambda \in [0, 1]$.

| benchmark | best kernel |
|---|---|
| interaction, exp | Duplicate, $\lambda = 0.7$ |
| interaction, loc | Pair kernel, $\lambda = 0.6$ |
| interaction, phy | Duplicate, $\lambda = 0.8$ |
| interaction, y2h | Duplicate / Pair kernel, $\lambda = 0$ |
| interaction, integrated | Duplicate / Pair kernel, $\lambda = 0$ |
| metabolic, exp | Pair kernel, $\lambda = 0.6$ |
| metabolic, loc | Pair kernel, $\lambda = 1$ |
| metabolic, phy | Pair kernel, $\lambda = 0.6$ |
| metabolic, integrated | Duplicate / Pair kernel, $\lambda = 0$ |

# Interpolation kernel




*Metabolic networks with localization data (left); PPI network with expression data (right)*

# Applications: missing enzyme prediction

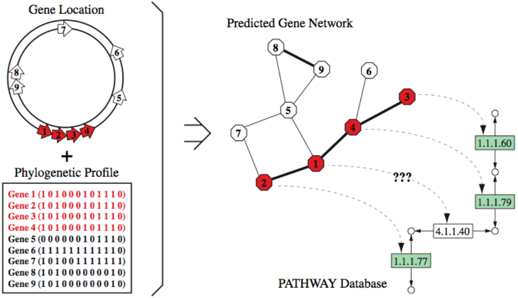
the FEBS Journal

## Prediction of missing enzyme genes in a bacterial metabolic network

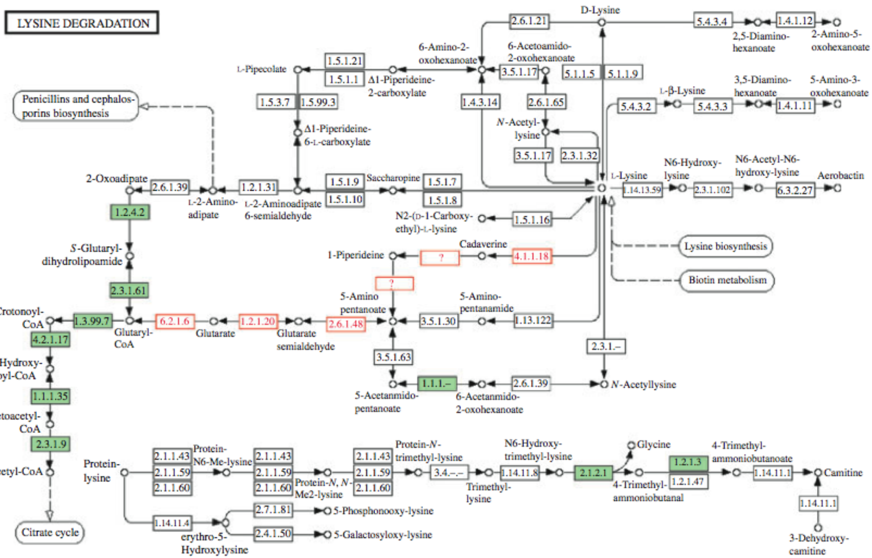
### Reconstruction of the lysine-degradation pathway of *Pseudomonas aeruginosa*

Yoshihiro Yamanishi[1], Hisaaki Mihara[2], Motoharu Osaki[2], Hisashi Muramatsu[3], Nobuyoshi Esaki[2], Tetsuya Sato[1], Yoshiyuki Hizukuri[1], Susumu Goto[1] and Minoru Kanehisa[1]

1 Bioinformatics Center, Institute for Chemical Research, Kyoto University, Japan
2 Division of Environmental Chemistry, Institute for Chemical Research, Kyoto University, Japan
3 Department of Biology, Graduate School of Science, Osaka University, Japan



# Applications: missing enzyme prediction



# Applications: missing enzyme prediction

900 DOI 10.1002/pmic.200600862 *Proteomics* 2007, *7*, 900–909

RESEARCH ARTICLE

## Prediction of nitrogen metabolism-related genes in *Anabaena* by kernel-based network analysis

*Shinobu Okamoto[1]*, Yoshihiro Yamanishi[1], Shigeki Ehira[2], Shuichi Kawashima[3], Koichiro Tonomura[1]** and Minoru Kanehisa[1]*

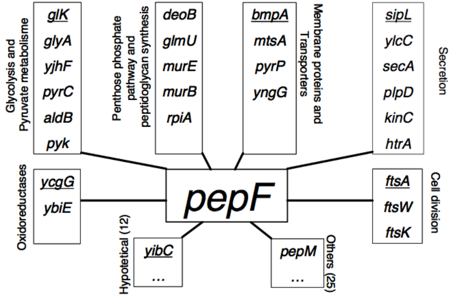
[1] Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Japan
[2] Department of Biochemistry and Molecular Biology, Faculty of Science, Saitama University, Saitama, Japan
[3] Human Genome Center, Institute of Medical Science, University of Tokyo, Meguro, Japan

# Applications: function annotation

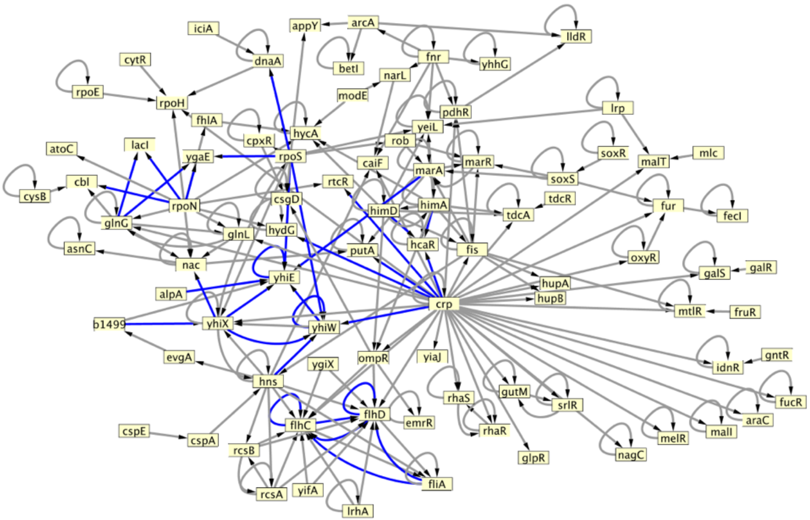
## Determination of the role of the bacterial peptidase PepF by statistical inference and further experimental validation

Liliana LOPEZ KLEINE[1,2] , Alain TRUBUIL[1], Véronique MONNET[2]

[1]Unité de Mathématiques et Informatiques Appliquées. INRA Jouy en Josas 78352, France.
[2]Unité de Biochimie Bactérienne. INRA Jouy en Josas 78352, France.



# Application: predicted regulatory network (E. coli)



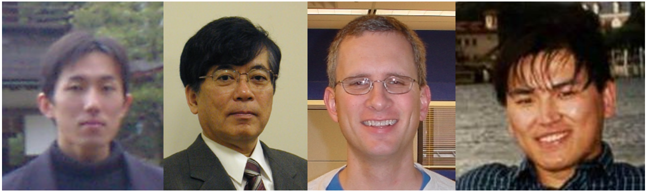

*Prediction at 60% precision, restricted to transcription factors (from Mordelet and V., 2008).*

# Outline

# Conclusion

- When the network is known in part, supervised methods are more adapted than unsupervised ones.
- A variety of methods have been investigated recently (metric learning, matrix completion, pattern recognition).
  - work for any network
  - work with any data
  - can integrate heterogeneous data, which strongly improves performance
- Promising topic: infer edges simultaneously with global constraints on the graph?

# People I need to thank



Yoshihiro Yamanishi, Minoru Kanehisa (Univ. Kyoto) Jian Qian, Bill Noble (Univ. Washington), Kevin Bleakley, Gerard Biau (Univ. Montpellier), Fantine Mordelet, Martial Hue (ParisTech)