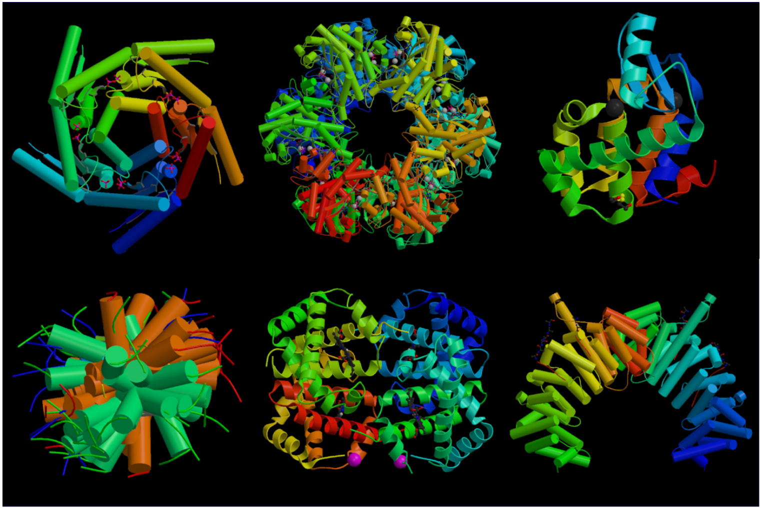
# Inference of missing edges in biological networks

Jean-Philippe Vert
Jean-Philippe.Vert@mines-paristech.fr

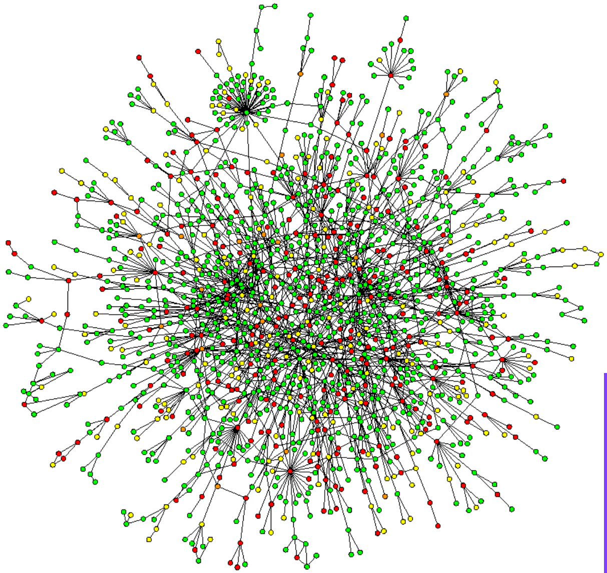
Mines ParisTech, Institut Curie, INSERM U900

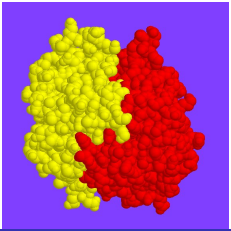
Telecom ParisTech, October 23, 2008.

## Proteins

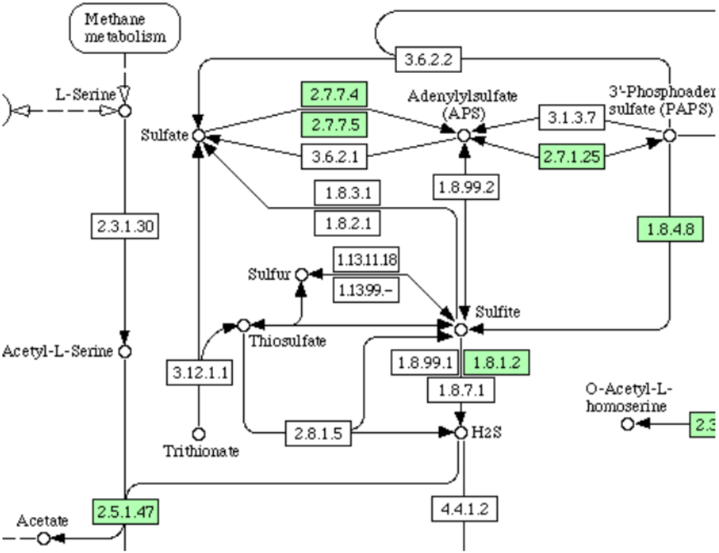


## Network 1: protein-protein interaction

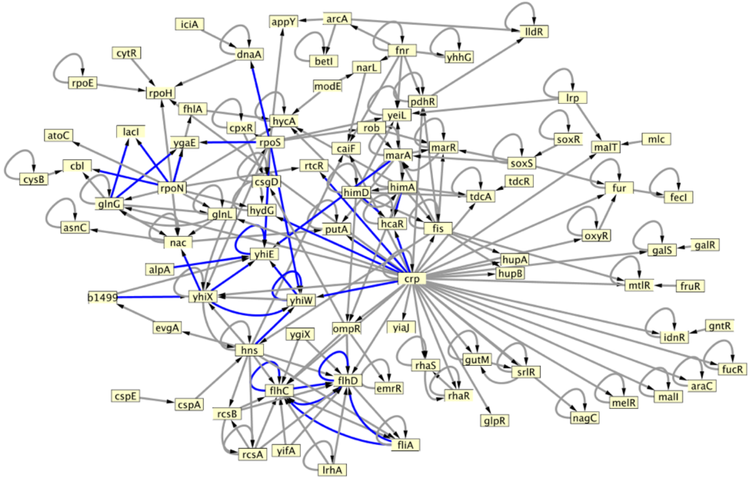




# Network 2: metabolic network



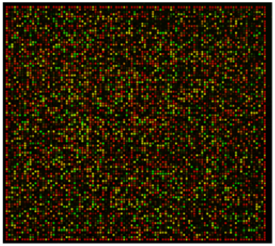
# Network 3: gene regulatory network

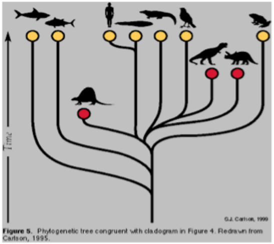


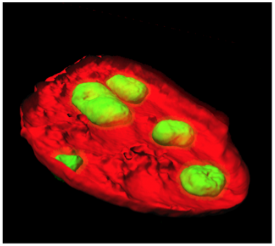
## Data available

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

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

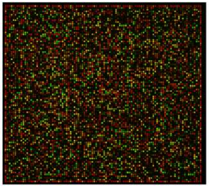
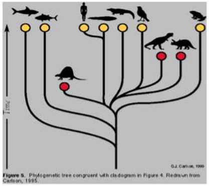


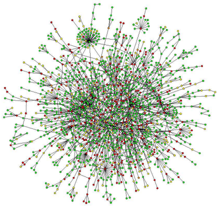
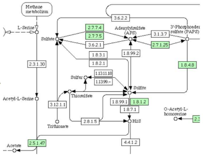
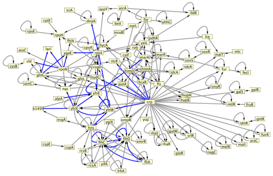

Time
G.J. Carlson, 1999
Figure 5. Phylogenetic tree congruent with cladogram in Figure 4. Redrawn from Carlson, 1995.



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

# Our goal



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



Methane metabolism
L-Serine
Sulfate
Adenylylsulfate (APS)
3'-Phosphoadenylyl sulfate (PAPS)
Sulfur
Thiosulfate
Sulfite
Acetyl-L-Serine
Trithionate
H2S
O-Acetyl-L-homoserine
Acetate

crp
rpoS
rpoN
hns
fis
arcA
fnr

## 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

## 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

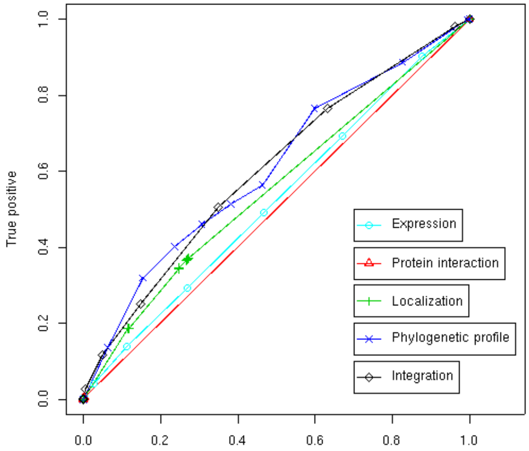
- Excellent approach if the model is correct and enough data are available
- Interpretability of the model
- Inclusion of prior knowledge

### Cons

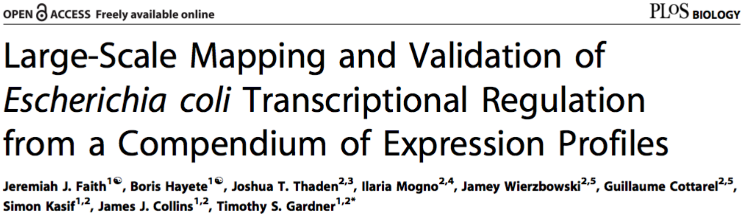
- Specific to particular data and networks
- Needs a correct model!
- Difficult integration of heterogeneous data
- Often needs a lot of data and long computation time

## 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).



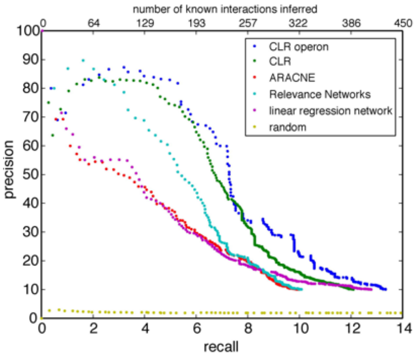
# Evaluation on regulatory network reconstruction



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*]



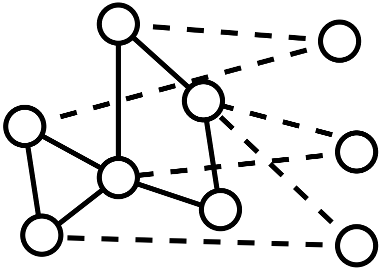
## Outline

## Supervised methods

### Motivation

In actual applications,

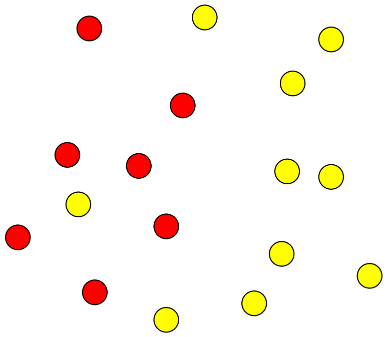
- we know in advance parts of the network to be inferred
- the problem is to add/remove nodes and edges using genomic data as side information



### Supervised method

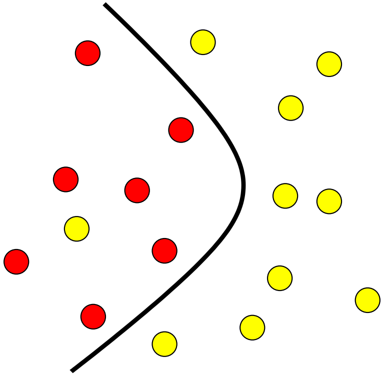
- Given genomic data and the currently known network...
- Infer missing edges between current nodes and additional nodes.

## Pattern recognition



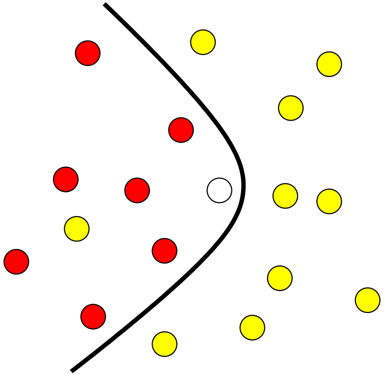
- Given a training set of patterns in two classes, learn to discriminate them
- Many algorithms (ANN, SVM, Decision tress, ...)

## Pattern recognition



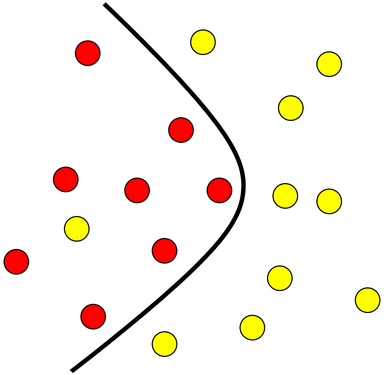
- Given a training set of patterns in two classes, learn to discriminate them
- Many algorithms (ANN, SVM, Decision tress, ...)

## 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 $(X_1, X_2)$

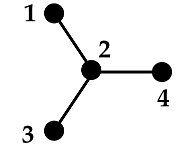
**Two solutions**

- 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

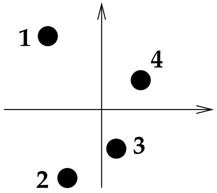
## Pattern recognition for pairs

### Formulation and 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!



Known graph

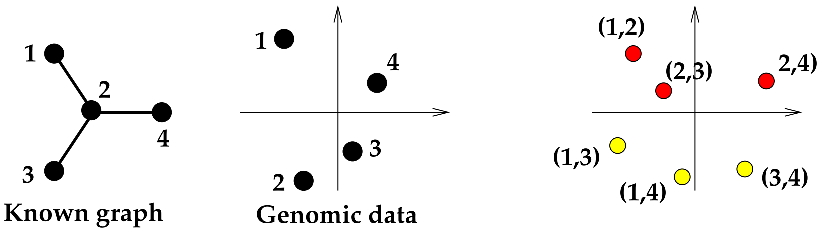


Genomic data

## Pattern recognition for pairs

### Formulation and basic issue

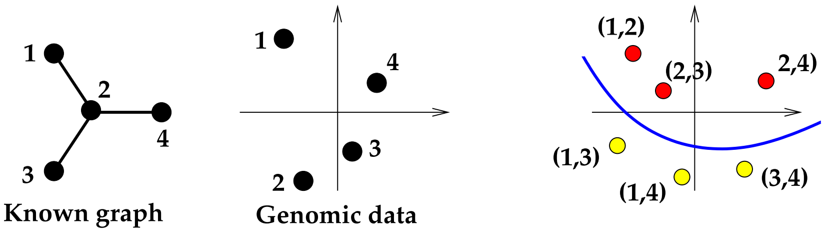
- A pair can be connected (1) or not connected (-1)
- From the known subgraph we can extract examples of connected and non-connected pairs
- However the genomic data characterize individual proteins; we need to work with pairs of proteins instead!



## Pattern recognition for pairs

### Formulation and 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

### Representing a pair as a vector

- Each individual protein is represented by a vector $v \in \mathbb{R}^p$
- 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 $v$?

## Representing a pair

### Direct sum

- 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...

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

## Representing a pair

### Direct sum

- 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...

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

## Representing a pair

### Direct product

- 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...)

# Representing a pair

## Direct product

- 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...)

## Representing a pair

### Direct product

- 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...)

## Other representations for pairs

### Symmetric tensor product (Ben-Hur and Noble, 2006)

$$\psi(u,v) = (u \otimes v) + (v \otimes u) \; .$$

Intuition: a pair $(A,B)$ is similar to a pair $(C,D)$ if:

- $A$ is similar to $C$ and $B$ is similar to $D$, or...
- $A$ is similar to $D$ and $B$ is similar to $C$

### Metric learning (V. et al, 2007)

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

Intuition: a pair $(A,B)$ is similar to a pair $(C,D)$ if:

- $A - B$ is similar to $C - D$, or...
- $A - B$ is similar to $D - C$.

## Other representations for pairs

### Symmetric tensor product (Ben-Hur and Noble, 2006)

$$\psi(u, v) = (u \otimes v) + (v \otimes u) \; .$$

Intuition: a pair $(A, B)$ is similar to a pair $(C, D)$ if:

- $A$ is similar to $C$ and $B$ is similar to $D$, or...
- $A$ is similar to $D$ and $B$ is similar to $C$

### Metric learning (V. et al, 2007)

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

Intuition: a pair $(A, B)$ is similar to a pair $(C, D)$ if:

- $A - B$ is similar to $C - D$, or...
- $A - B$ is similar to $D - C$.

## Link with metric learning

### Metric learning

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

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

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

### Metric learning

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

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

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

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

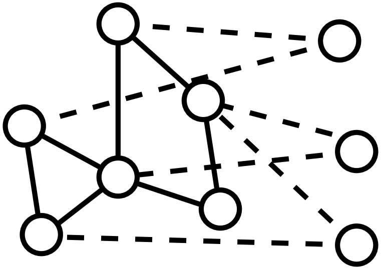
  solves this metric learning problem without the constraint $M \geq 0$.
- Equivalently, train the SVM over pairs with the metric learning pairwise kernel:

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

## Supervised inference with local models

### The idea (Bleakley et al., 2007)

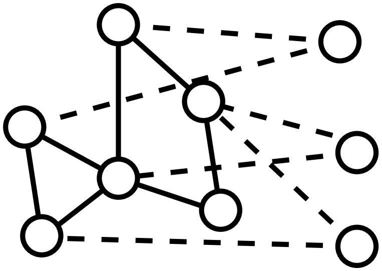
- Motivation: define specific models for each target node to discriminate between its neighbors and the others
- Treat each node independently from the other. Then combine predictions for ranking candidate edges.

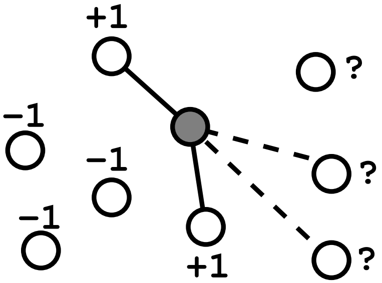


## Supervised inference with local models

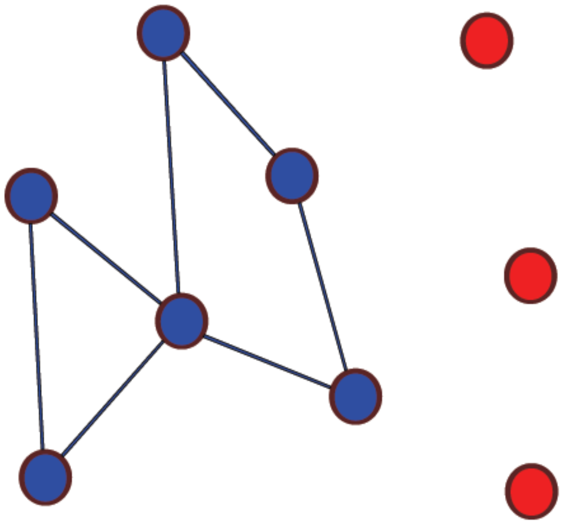
### The idea (Bleakley et al., 2007)

- 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.

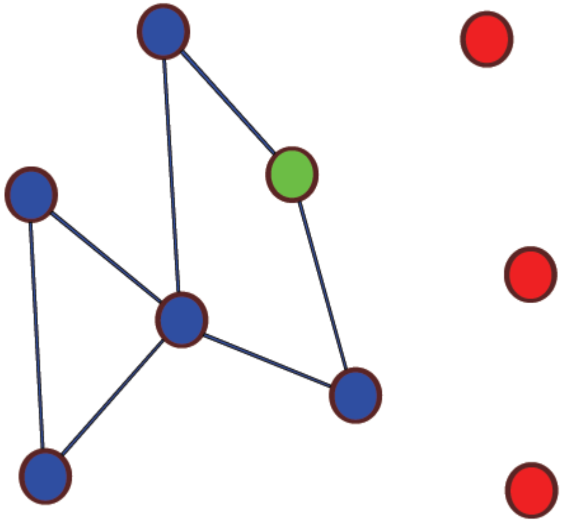




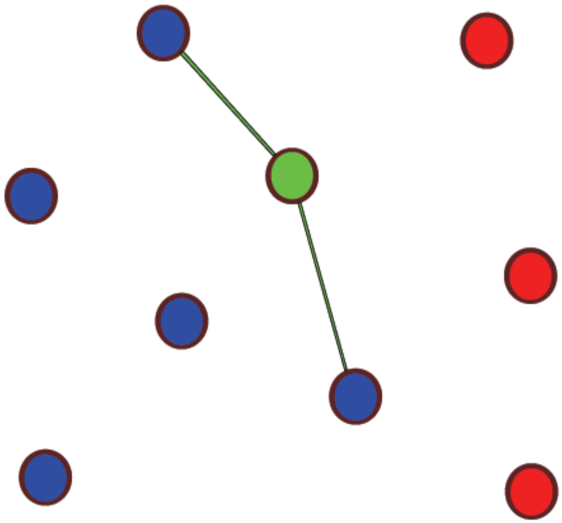
## The LOCAL model



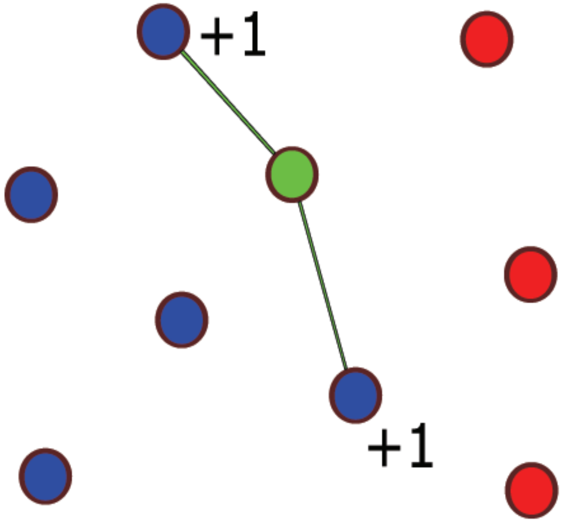
## The LOCAL model



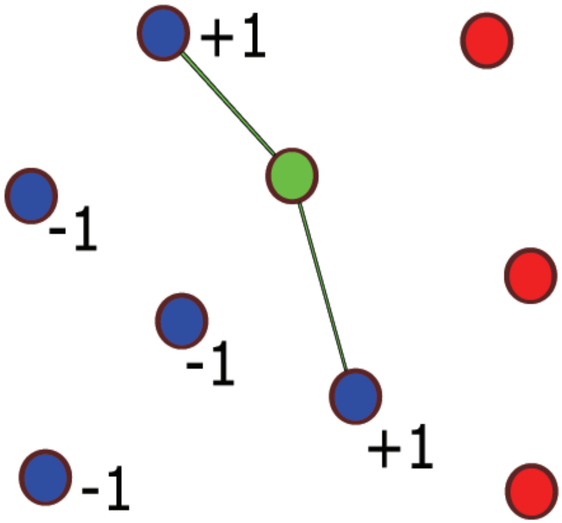
## The LOCAL model



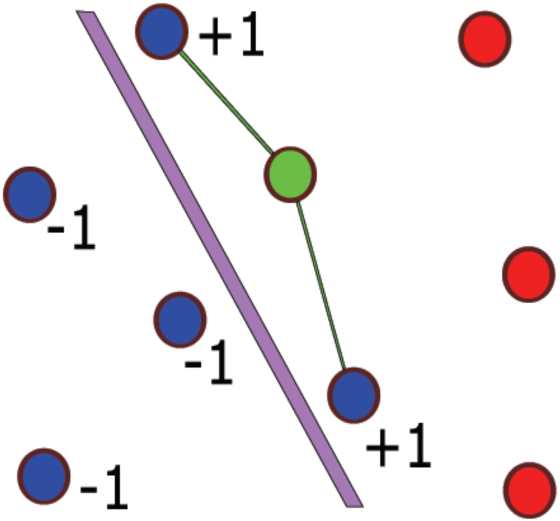
## The LOCAL model



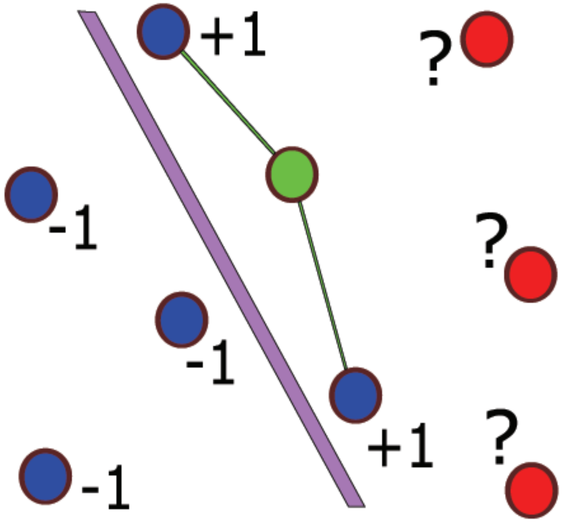
## The LOCAL model



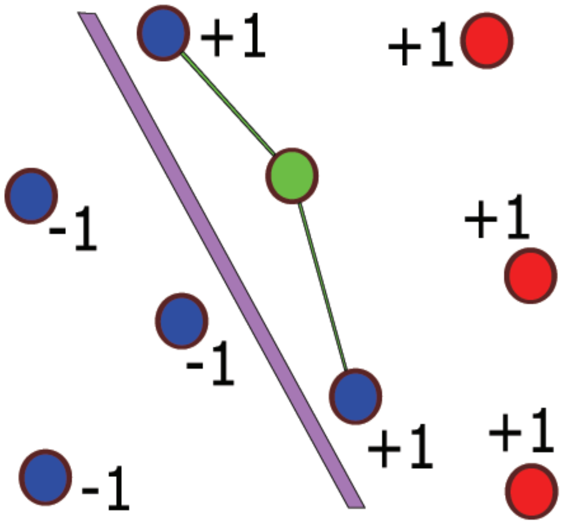
# The LOCAL model



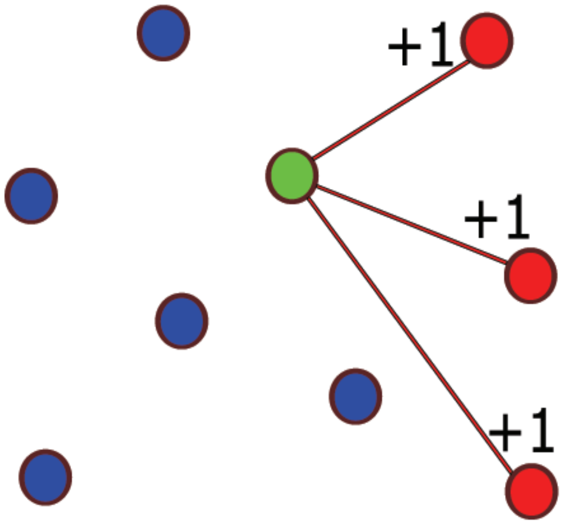
## The LOCAL model



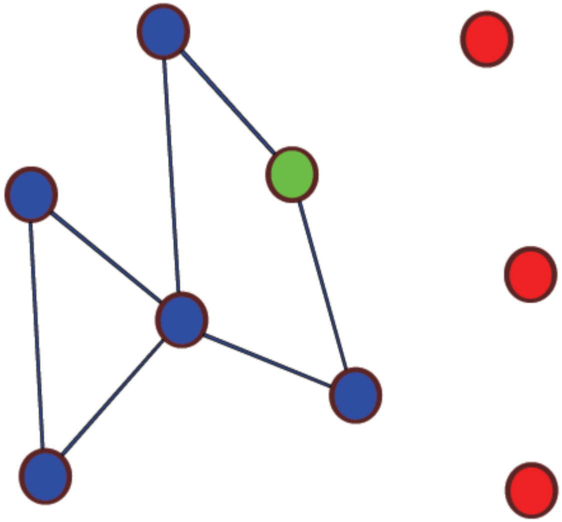
# The LOCAL model



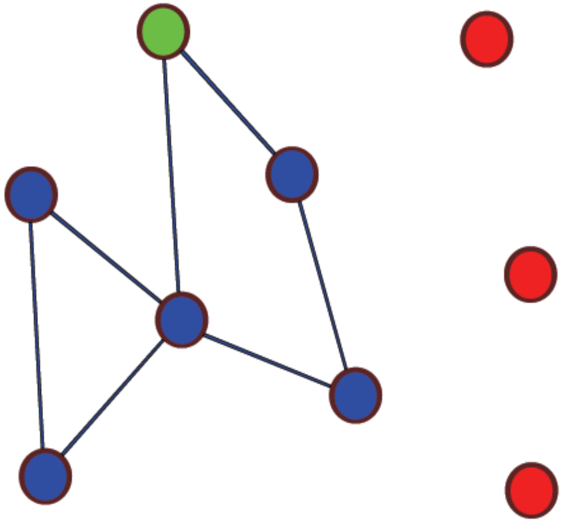
# The LOCAL model



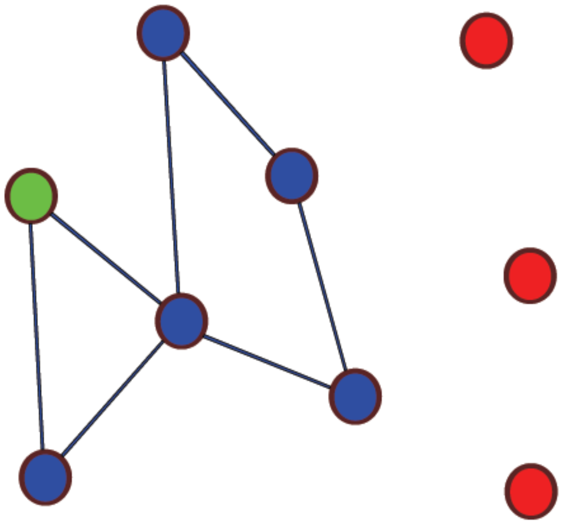
## The LOCAL model



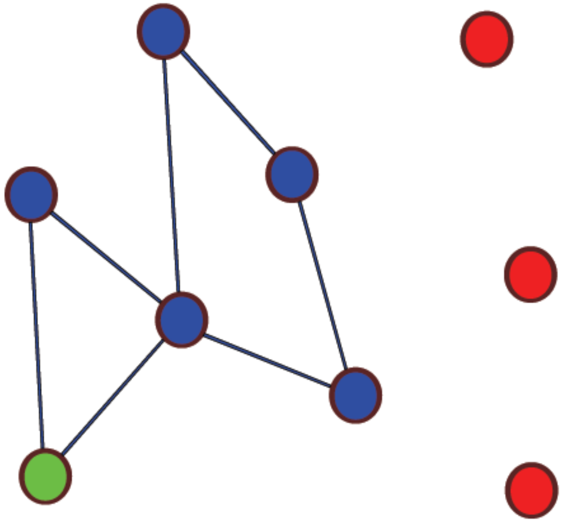
## The LOCAL model



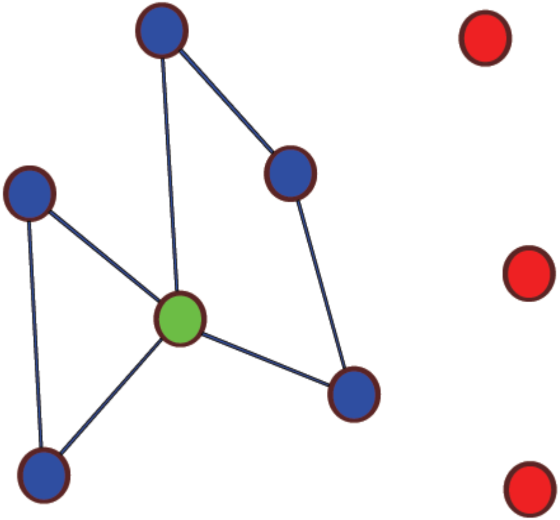
## The LOCAL model



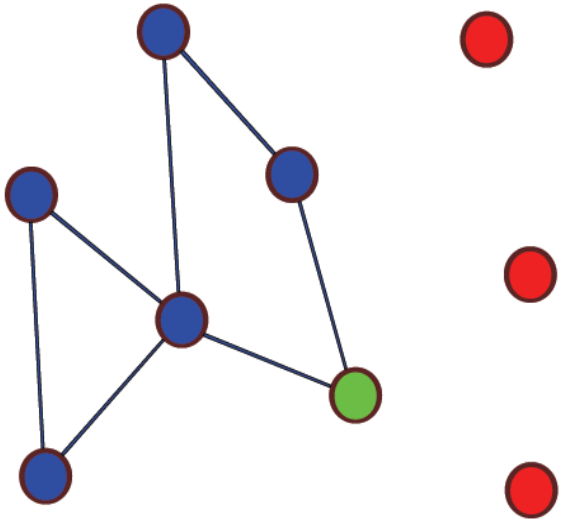
## The LOCAL model



## The LOCAL model



## The LOCAL model



## A few remarks

- 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

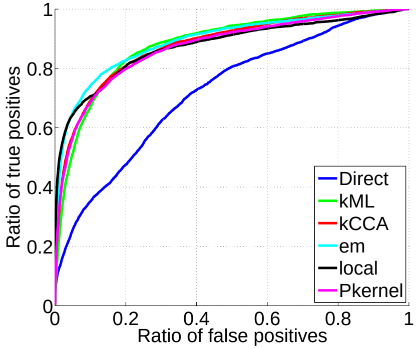
# A few remarks

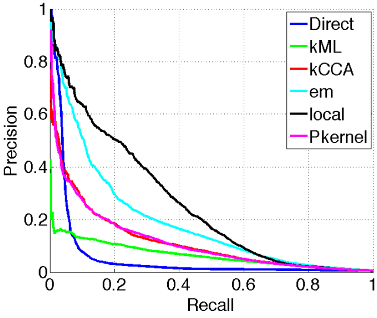
- Weak hypothesis:
  - if A is connected to B,
  - if C is similar to B,
  - then A is likely to be connected to C.
- Computationally: much faster to train $N$ local models with $N$ training points each, than to train 1 model with $N^2$ training points.
- Caveats:
  - each local model may have very few training points
  - no sharing of information between different local models

## A few remarks

- 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

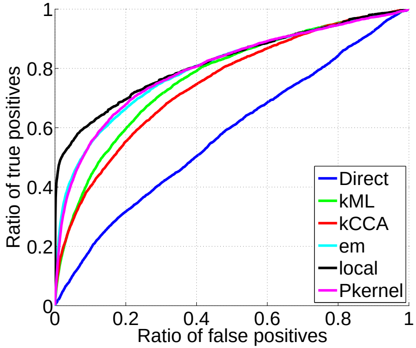
# Results: protein-protein interaction (yeast)

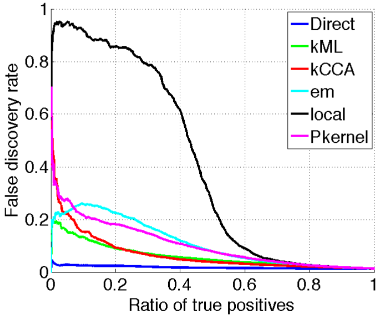




(from Bleakley et al., 2007)

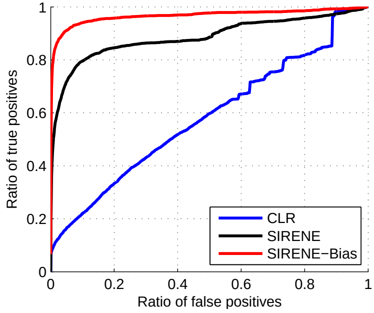
## Results: metabolic gene network (yeast)

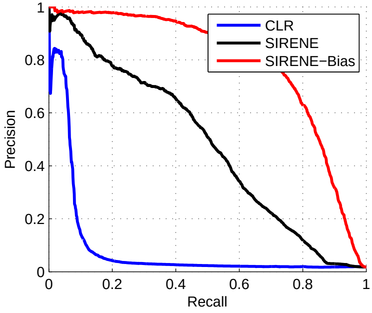




(from Bleakley et al., 2007)

## 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)*

# Applications: missing enzyme prediction

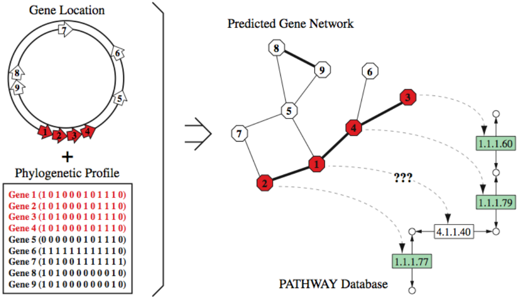
the FEBS Journal

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

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

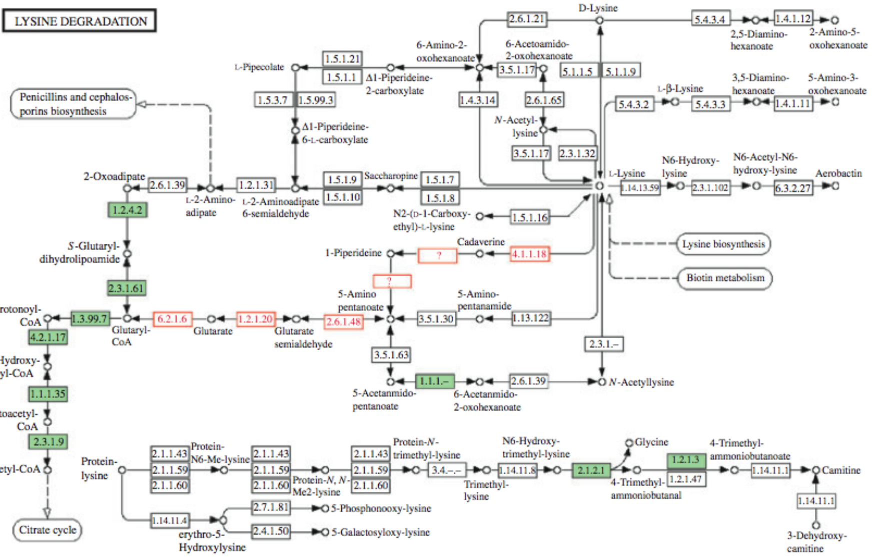
Yoshihiro Yamanishi[1], Hisaaki Mihara[2], Motoharu Osaki[2], Hisashi Muramatsu[3], Nobuyoshi Esaki[2], Tetsuya Sato[1], Yoshiyuki Hizukuri[1], Susumu Goto[1] and Minoru Kanehisa[1]

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

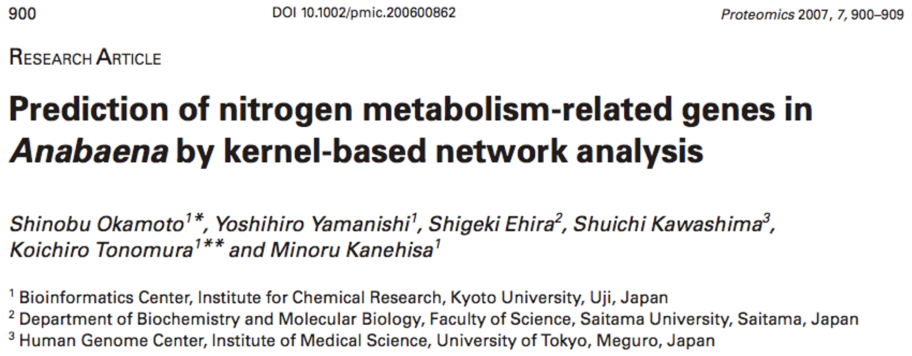


Gene 1 (1 0 1 0 0 0 1 0 1 1 1 0)
Gene 2 (1 0 1 0 0 0 1 0 1 1 1 0)
Gene 3 (1 0 1 0 0 0 1 0 1 1 1 0)
Gene 4 (1 0 1 0 0 0 1 0 1 1 1 0)
Gene 5 (0 0 0 0 0 0 1 0 1 1 1 0)
Gene 6 (1 1 1 1 1 1 1 1 1 1 1 0)
Gene 7 (1 0 1 0 0 1 1 1 1 1 1 1)
Gene 8 (1 0 1 0 0 0 0 0 0 0 1 0)
Gene 9 (1 0 1 0 0 0 0 0 0 0 1 0)

# 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]*

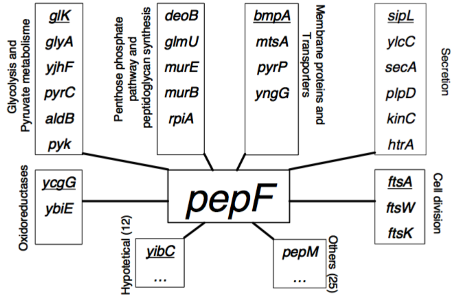
[1] Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Japan
[2] Department of Biochemistry and Molecular Biology, Faculty of Science, Saitama University, Saitama, Japan
[3] Human Genome Center, Institute of Medical Science, University of Tokyo, Meguro, Japan

## Applications: function annotation

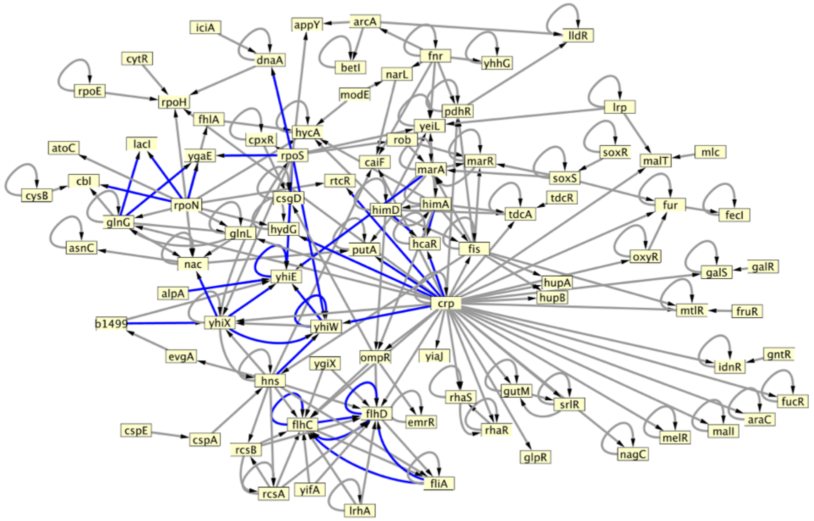
Determination of the role of the bacterial peptidase PepF by statistical inference and further experimental validation

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.



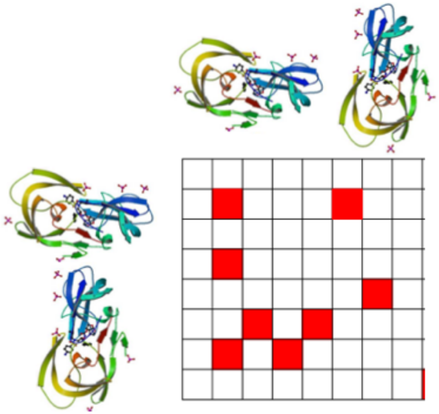
## Application: predicted regulatory network (E. coli)



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

## Outline

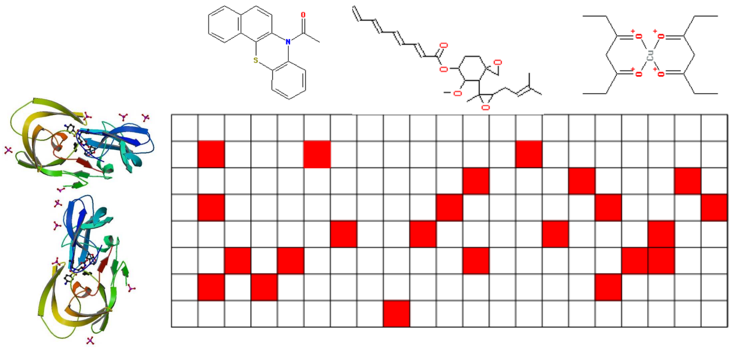
## Graph learning = learning over pairs of vertices



# Extension (not symmetric)

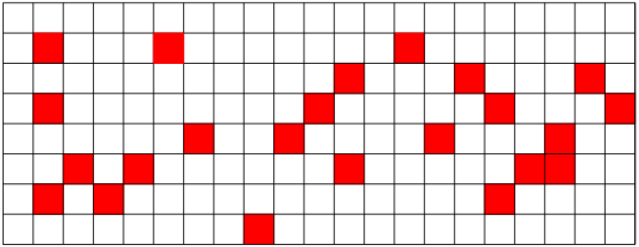
## Chemogenomics

- Given a family of proteins of therapeutic interest (e.g., GPCR's)
- Given all known small molecules that bind to these proteins
- Can we predict unknown interactions?



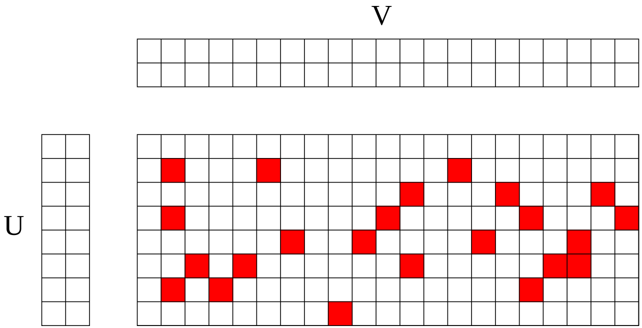
## Collaborative Filtering (CF)

- Given a set of $n_{\mathcal{X}}$ "movies" $\mathbf{x} \in \mathcal{X}$ and a set of $n_{\mathcal{Y}}$ "customers" $\mathbf{y} \in \mathcal{Y}$,
- predict the "rating" $z(\mathbf{x}, \mathbf{y}) \in \mathcal{Z}$ of customer $\mathbf{y}$ for movie $\mathbf{x}$
- Training data: large $n_{\mathcal{X}} \times n_{\mathcal{Y}}$ incomplete matrix $Z$ that describes the known ratings of some customers for some movies
- Goal: complete the matrix.



# CF by low-rank matrix approximation

- A common strategy for CF
- $Z$ has rank less than $k \Leftrightarrow \boxed{Z = UV^\top}$ $U \in \mathbb{R}^{n_\mathcal{X} \times k}$, $V \in \mathbb{R}^{n_\mathcal{Y} \times k}$
- Examples: PLSA (Hoffmann, 2001), MMMF (Srebro et al, 2004)
- Numerical and statistical efficiency



## CF by low-rank matrix approximation example

### Fitting low-rank models (Srebro et al, 2004)

- Relax the (non-convex) rank of $Z$ into the (convex) trace norm of $Z$: if $\sigma_i(Z)$ are the singular values of $Z$,

$$\mathrm{rank} Z = \sum_i 1_{\sigma_i(Z)>0} \qquad \|Z\|_* = \sum_i \sigma_i(Z)\,.$$

- $n$ observations $z_u$ corresponding to $\mathbf{x}_{i(u)}$ and $\mathbf{y}_{j(u)}$, $u = 1, \ldots, n$:

$$\min_{Z \in \mathbb{R}^{n_\mathcal{X} \times n_\mathcal{Y}}} \sum_{u=1}^{n} \ell(z_u, Z_{i(u),j(u)}) + \lambda \|Z\|_*\,,$$

  where $\ell(z, z')$ is a convex loss function.
- This is an SDP if $\ell$ is SDP-representable

## CF by low-rank matrix approximation example

### Fitting low-rank models (Srebro et al, 2004)

- Relax the (non-convex) rank of $Z$ into the (convex) trace norm of $Z$: if $\sigma_i(Z)$ are the singular values of $Z$,

$$\mathrm{rank} Z = \sum_i 1_{\sigma_i(Z)>0} \qquad \|Z\|_* = \sum_i \sigma_i(Z)\,.$$

- $n$ observations $z_u$ corresponding to $\mathbf{x}_{i(u)}$ and $\mathbf{y}_{j(u)}$, $u = 1, \ldots, n$:

$$\min_{Z \in \mathbb{R}^{n_\mathcal{X} \times n_\mathcal{Y}}} \sum_{u=1}^{n} \ell(z_u, Z_{i(u),j(u)}) + \lambda \|Z\|_*\,,$$

  where $\ell(z, z')$ is a convex loss function.
- This is an SDP if $\ell$ is SDP-representable

## Remark

### Basic facts

- $n_{\mathcal{X}}$ movies and $n_{\mathcal{Y}}$ customers
- The known rating $z(\mathbf{x}_i, \mathbf{y}_j)$ of customer $\mathbf{y}_j$ for movie $\mathbf{x}_i$ is stored in the $(i, j)$-th entry of a matrix $M$ (of size $n_{\mathcal{X}} \times n_{\mathcal{Y}}$).
- $M$ represents a linear application / bilinear form:

$$M : \mathbb{R}^{n_{\mathcal{Y}}} \to \mathbb{R}^{n_{\mathcal{X}}}$$

  defined by:

$$e_i^\top M f_j = M_{i,j}$$

- Rank / trace norm are spectral properties of the linear application

## Reformulation

- Represent the $i$-th movie $\mathbf{x}_i \in \mathcal{X}$ (resp. $j$-th customer $\mathbf{y}_j \in \mathcal{Y}$) by the $i$-th basis vector $e_i \in \mathbb{R}^{n_\mathcal{X}}$ (resp. $f_j \in \mathbb{R}^{n_\mathcal{Y}}$):

$$\phi_X(x_i) = e_i \,, \quad \phi_Y(y_j) = f_j \,.$$

- Approximate the rating function by a bilinear form:

$$\forall (\mathbf{x}_i, \mathbf{y}_j) \in \mathcal{X} \times \mathcal{Y} \,, \quad G_M(\mathbf{x}_i, \mathbf{y}_j) = \phi_X(\mathbf{x}_i)^\top M \phi_Y(\mathbf{y}_j) \,,$$

  by constraining a spectral property of $M : \mathbb{R}^{n_\mathcal{X}} \mapsto \mathbb{R}^{n_\mathcal{X}}$.

**An idea**

If we have additional attributes about movies / customer, why not include them in $\phi_X(\mathbf{x})$ and $\phi_Y(\mathbf{y})$?

## Reformulation

- Represent the $i$-th movie $\mathbf{x}_i \in \mathcal{X}$ (resp. $j$-th customer $\mathbf{y}_j \in \mathcal{Y}$) by the $i$-th basis vector $e_i \in \mathbb{R}^{n_\mathcal{X}}$ (resp. $f_j \in \mathbb{R}^{n_\mathcal{Y}}$):

$$\phi_X(x_i) = e_i \,, \quad \phi_Y(y_j) = f_j \,.$$

- Approximate the rating function by a bilinear form:

$$\forall (\mathbf{x}_i, \mathbf{y}_j) \in \mathcal{X} \times \mathcal{Y} \,, \quad G_M(\mathbf{x}_i, \mathbf{y}_j) = \phi_X(\mathbf{x}_i)^\top M \phi_Y(\mathbf{y}_j) \,,$$

  by constraining a spectral property of $M : \mathbb{R}^{n_\mathcal{X}} \mapsto \mathbb{R}^{n_\mathcal{X}}$.

### An idea

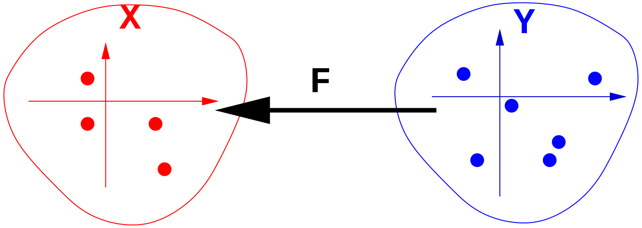
If we have additional attributes about movies / customer, why not include them in $\phi_X(\mathbf{x})$ and $\phi_Y(\mathbf{y})$?

## Setting

- Movies: points in a Hilbert space $\mathcal{X}$
- Customers: points in a Hilbert space $\mathcal{Y}$
- We model the preference of customer **y** for a movie **x** by a bilinear form:

$$f(\mathbf{x},\mathbf{y}) = \langle \mathbf{x}, F\mathbf{y}\rangle_{\mathcal{X}} \ ,$$

where $F \in \mathcal{B}_0\left(\mathcal{Y},\mathcal{X}\right)$ is a compact linear operator (i.e., a "matrix").



## Spectra of compact operators

### Classical results

- For $(\mathbf{x},\mathbf{y})$ in $\mathcal{X} \times \mathcal{Y}$ the tensor product $\mathbf{x} \otimes \mathbf{y}$ is the operator

$$\forall \mathbf{h} \in \mathcal{Y}, \quad (\mathbf{x} \otimes \mathbf{y})\,\mathbf{h} = \langle \mathbf{y}, \mathbf{h} \rangle_{\mathcal{Y}}\, \mathbf{x}\,.$$

- Any compact operator $F : \mathcal{Y} \to \mathcal{X}$ admits a spectral decomposition:

$$F = \sum_{i=1}^{\infty} \sigma_i \mathbf{u}_i \otimes \mathbf{v}_i\,.$$

where the $\sigma_i \geq 0$ are the singular values and $(\mathbf{u}_i)_{i\in\mathbb{N}}$ and $(\mathbf{v}_i)_{i\in\mathbb{N}}$ are orthonormal families in $\mathcal{X}$ and $\mathcal{Y}$.

- The spectrum of $F$ is the set of singular values sorted in decreasing order: $\sigma_1(F) \geq \sigma_2(F) \geq \ldots \geq 0$.
- This is the natural generalization of singular values for matrices.

## Spectra of compact operators

### Classical results

- For $(\mathbf{x}, \mathbf{y})$ in $\mathcal{X} \times \mathcal{Y}$ the tensor product $\mathbf{x} \otimes \mathbf{y}$ is the operator

$$\forall \mathbf{h} \in \mathcal{Y}, \quad (\mathbf{x} \otimes \mathbf{y})\, \mathbf{h} = \langle \mathbf{y}, \mathbf{h} \rangle_{\mathcal{Y}}\, \mathbf{x}\,.$$

- Any compact operator $F : \mathcal{Y} \to \mathcal{X}$ admits a spectral decomposition:

$$F = \sum_{i=1}^{\infty} \sigma_i \mathbf{u}_i \otimes \mathbf{v}_i\,.$$

where the $\sigma_i \geq 0$ are the singular values and $(\mathbf{u}_i)_{i \in \mathbb{N}}$ and $(\mathbf{v}_i)_{i \in \mathbb{N}}$ are orthonormal families in $\mathcal{X}$ and $\mathcal{Y}$.

- The spectrum of $F$ is the set of singular values sorted in decreasing order: $\sigma_1(F) \geq \sigma_2(F) \geq \ldots \geq 0$.
- This is the natural generalization of singular values for matrices.

## Spectra of compact operators

### Classical results

- For $(\mathbf{x}, \mathbf{y})$ in $\mathcal{X} \times \mathcal{Y}$ the tensor product $\mathbf{x} \otimes \mathbf{y}$ is the operator

$$\forall \mathbf{h} \in \mathcal{Y}, \quad (\mathbf{x} \otimes \mathbf{y})\, \mathbf{h} = \langle \mathbf{y}, \mathbf{h} \rangle_{\mathcal{Y}} \, \mathbf{x} \, .$$

- Any compact operator $F : \mathcal{Y} \to \mathcal{X}$ admits a spectral decomposition:

$$F = \sum_{i=1}^{\infty} \sigma_i \mathbf{u}_i \otimes \mathbf{v}_i \, .$$

  where the $\sigma_i \geq 0$ are the singular values and $(\mathbf{u}_i)_{i \in \mathbb{N}}$ and $(\mathbf{v}_i)_{i \in \mathbb{N}}$ are orthonormal families in $\mathcal{X}$ and $\mathcal{Y}$.
- The spectrum of $F$ is the set of singular values sorted in decreasing order: $\sigma_1(F) \geq \sigma_2(F) \geq \ldots \geq 0$.
- This is the natural generalization of singular values for matrices.

## Useful classes for operators

### Operators of finite rank

- The rank of an operator is the number of strictly positive singular values.
- Hence operators of rank smaller or equal to $k$ are characterized by:

$$\sigma_{k+1}(F) = 0 \,.$$

### Trace-class operators

The trace-class operators are the compact operators $F$ that satisfy:

$$\| F \|_* := \sum_{i=1}^{\infty} \sigma_i(F) < \infty \,.$$

$\| F \|_*$ is a norm over the trace-class operators, called the trace norm.

## Useful classes for operators

### Operators of finite rank

- The rank of an operator is the number of strictly positive singular values.
- Hence operators of rank smaller or equal to $k$ are characterized by:

$$\sigma_{k+1}(F) = 0 \,.$$

### Trace-class operators

The trace-class operators are the compact operators $F$ that satisfy:

$$\| F \|_* := \sum_{i=1}^{\infty} \sigma_i(F) < \infty \,.$$

$\| F \|_*$ is a norm over the trace-class operators, called the trace norm.

## Useful classes for operators (cont.)

### Hilbert-Schmidt operators

- The Hilbert-Schmidt operators are compact operators $F$ that satisfy:

$$\| F \|_{Fro}^2 := \sum_{i=1}^{\infty} \sigma_i(F)^2 < \infty \,.$$

- They form a Hilbert space with inner product:

$$\left\langle \mathbf{x} \otimes \mathbf{y}, \mathbf{x}' \otimes \mathbf{y}' \right\rangle_{\mathcal{X} \otimes \mathcal{Y}} = \left\langle \mathbf{x}, \mathbf{x}' \right\rangle_{\mathcal{X}} \left\langle \mathbf{y}, \mathbf{y}' \right\rangle_{\mathcal{Y}} \,.$$

## Spectral penalty function

**Definition**

A function $\Omega : \mathcal{B}_0(\mathcal{Y}, \mathcal{X}) \mapsto \mathbb{R} \cup \{+\infty\}$ is called a spectral penalty function if it can be written as:

$$\Omega(F) = \sum_{i=1}^{\infty} s_i\left(\sigma_i(F)\right) ,$$

where for any $i \geq 1$, $s_i : \mathbb{R}^+ \mapsto \mathbb{R}^+ \cup \{+\infty\}$ is a non-decreasing penalty function satisfying $s_i(0) = 0$.

## Spectral penalty function

### Examples

- Rank constraint: take $s_{k+1}(0) = 0$ and $s_{k+1}(u) = +\infty$ for $u > 0$, and $s_i = 0$ for $i \geq k$. Then

$$\Omega(F) = \begin{cases} 0 & \text{if } rank(F) \leq k \,, \\ +\infty & \text{if } rank(F) > k \,. \end{cases}$$

- Trace norm: take $s_i(u) = u$ for all $i$, then:

$$\Omega(F) = \| F \|_* \,.$$

- Hilbert-Schmidt norm: take $s_i(u) = u^2$ for all $i$, then

$$\Omega(F) = \| F \|_{Fro}^2 \,.$$

# Spectral penalty function

## Examples

- Rank constraint: take $s_{k+1}(0) = 0$ and $s_{k+1}(u) = +\infty$ for $u > 0$, and $s_i = 0$ for $i \geq k$. Then

$$\Omega(F) = \begin{cases} 0 & \text{if } rank(F) \leq k\,, \\ +\infty & \text{if } rank(F) > k\,. \end{cases}$$

- Trace norm: take $s_i(u) = u$ for all $i$, then:

$$\Omega(F) = \| F \|_* \,.$$

- Hilbert-Schmidt norm: take $s_i(u) = u^2$ for all $i$, then

$$\Omega(F) = \| F \|_{Fro}^2 \,.$$

## Spectral penalty function

### Examples

- Rank constraint: take $s_{k+1}(0) = 0$ and $s_{k+1}(u) = +\infty$ for $u > 0$, and $s_i = 0$ for $i \geq k$. Then

$$\Omega(F) = \begin{cases} 0 & \text{if } rank(F) \leq k\,, \\ +\infty & \text{if } rank(F) > k\,. \end{cases}$$

- Trace norm: take $s_i(u) = u$ for all $i$, then:

$$\Omega(F) = \| F \|_* \,.$$

- Hilbert-Schmidt norm: take $s_i(u) = u^2$ for all $i$, then

$$\Omega(F) = \| F \|_{Fro}^2 \,.$$

## Learning operator with spectral regularization

### Setting

- Training set: $(\mathbf{x}_i, \mathbf{y}_i, t_i)_{i=1,\ldots,N}$ a set of (movie,customer,preference).
- Loss function $l(t, t')$ : cost of predicting preference $t$ instead of $t'$.
- Empirical risk of an operator $F$:

$$R_N(F) = \frac{1}{N} \sum_{i=1}^{N} l\left( \langle \mathbf{x}_i, F\mathbf{y}_i \rangle_{\mathcal{X}}, t_i \right) .$$

### Learning an operator

$$\min_{F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X}),\ \Omega(F)<\infty} \left\{ R_N(F) + \lambda \Omega(F) \right\} .$$

## Learning operator with spectral regularization

### Setting

- Training set: $(\mathbf{x}_i, \mathbf{y}_i, t_i)_{i=1,\ldots,N}$ a set of (movie,customer,preference).
- Loss function $l(t, t')$ : cost of predicting preference $t$ instead of $t'$.
- Empirical risk of an operator $F$:

$$R_N(F) = \frac{1}{N}\sum_{i=1}^{N} l\left(\langle \mathbf{x}_i, F\mathbf{y}_i\rangle_{\mathcal{X}}, t_i\right) .$$

### Learning an operator

$$\min_{F\in\mathcal{B}_0(\mathcal{Y},\mathcal{X}),\ \Omega(F)<\infty} \left\{R_N(F) + \lambda\Omega(F)\right\} .$$

## Questions

### Theory

Is it a "good" algorithm in theory?

- To be investigated...
- See Srebro et al. (2004), Bach (2007) for preliminary results with the trace norm

### Practice

Can we implement it? Does it work on real data?

- Optimization problem in the space of compact operators... but we show later that it boils down to a finite-dimensional optimization problem
- Promising results on real data

## Questions

### Theory

Is it a "good" algorithm in theory?

- To be investigated...
- See Srebro et al. (2004), Bach (2007) for preliminary results with the trace norm

### Practice

Can we implement it? Does it work on real data?

- Optimization problem in the space of compact operators... but we show later that it boils down to a finite-dimensional optimization problem
- Promising results on real data

## Questions

### Theory

Is it a "good" algorithm in theory?

- To be investigated...
- See Srebro et al. (2004), Bach (2007) for preliminary results with the trace norm

### Practice

Can we implement it? Does it work on real data?

- Optimization problem in the space of compact operators... but we show later that it boils down to a finite-dimensional optimization problem
- Promising results on real data

## Questions

### Theory

Is it a "good" algorithm in theory?

- To be investigated...
- See Srebro et al. (2004), Bach (2007) for preliminary results with the trace norm

### Practice

Can we implement it? Does it work on real data?

- Optimization problem in the space of compact operators... but we show later that it boils down to a finite-dimensional optimization problem
- Promising results on real data

## A generalized representer theorem

**Theorem**

For any spectral penalty function $\Omega : \mathcal{B}_0(\mathcal{Y}, \mathcal{X}) \mapsto \mathbb{R}$, let the optimization problem:

$$\min_{F \in \mathcal{B}_0(\mathcal{Y},\mathcal{X}), \Omega(F) < \infty} \left\{ R_N(F) + \lambda \Omega(F) \right\} .$$

If the set of solutions is not empty, then there is a solution $F$ in $\mathcal{X}_N \otimes \mathcal{Y}_N$, i.e., there exists $\alpha \in \mathbb{R}^{m_\mathcal{X} \times m_\mathcal{Y}}$ such that:

$$F = \sum_{i=1}^{m_\mathcal{X}} \sum_{j=1}^{m_\mathcal{Y}} \alpha_{ij} \mathbf{u}_i \otimes \mathbf{v}_j ,$$

where $(\mathbf{u}_1, \ldots, \mathbf{u}_{m_\mathcal{X}})$ and $\left(\mathbf{v}_1, \ldots, \mathbf{v}_{m_\mathcal{Y}}\right)$ form orthonormal bases of $\mathcal{X}_N$ and $\mathcal{Y}_N$, respectively.

## Summary

We obtain various algorithms by choosing:

1. A loss function (depends on the application)
2. A spectral regularization (that is amenable to optimization)
3. Two Gram matrices (aka kernel matrices)

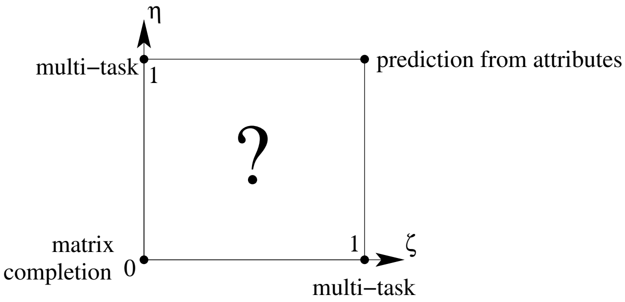
Both kernels and spectral regularization can be used to constrain the solution

## A family of kernels

Taken $K_{\otimes} = K \times G$ with

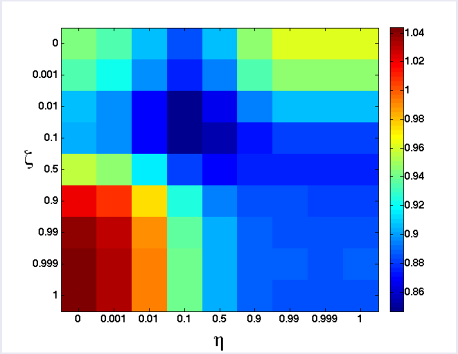
$$\begin{cases} K = \eta K^{x}_{Attribute} + (1-\eta) K^{x}_{Dirac}, \\ G = \zeta K^{y}_{Attribute} + (1-\zeta) K^{y}_{Dirac}, \end{cases}$$

for $0 \leq \eta \leq 1$ and $0 \leq \zeta \leq 1$



## Movies

- MovieLens 100k database, ratings with attributes
- Experiments with 943 movies and 1,642 customers, 100,000 rankings in $\{1, \ldots, 5\}$
- Train on a subset of the ratings, test on the rest
- error measured with MSE (best constant prediction: 1.26)



## Outline

## Take-home messages

- When the network is known in part, supervised methods can be 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
- Link with collaborative filtering with attributes
- Current research: infer edges simultaneously with global constraints on the graph?

## People I need to thank

- Yoshihiro Yamanishi, Minoru Kanehisa (Univ. Kyoto): kCCA, kML
- Jian Qian, Bill Noble (Univ. Washington): pairwise SVM
- Kevin Bleakley, Gerard Biau (Univ. Montpellier), Fantine Mordelet (ParisTech/Curie): local SVM
- Francis Bach (INRIA), Jake Abernethy (UC Berkeley), Theos Evgeniou (INSEAD): CF