Machine Learning in Computational Biology

Jean-Philippe Vert

Jean-Philippe.Vert@mines.org

Outline

Outline

Outline

Outline

Outline

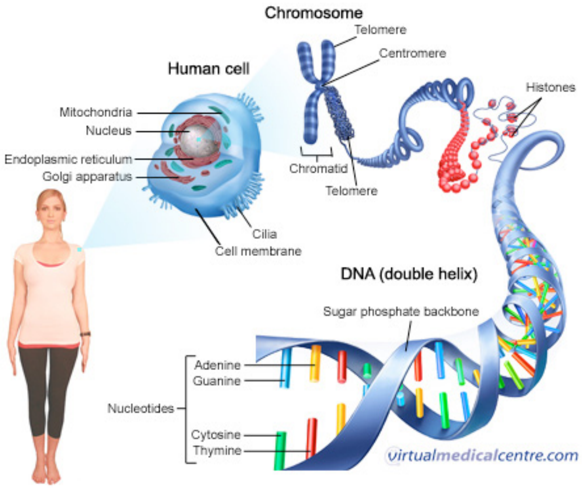
Outline

Outline

Outline

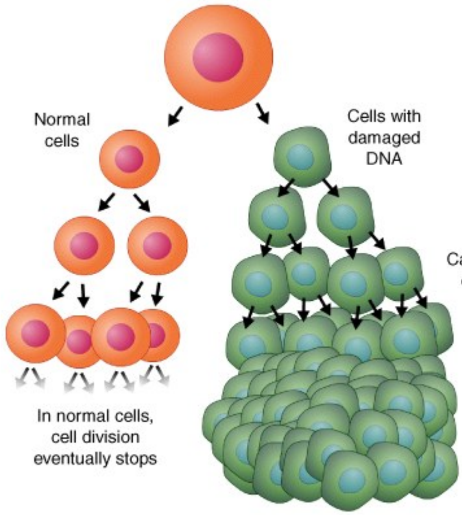
Outline

Cells, chromosomes, DNA



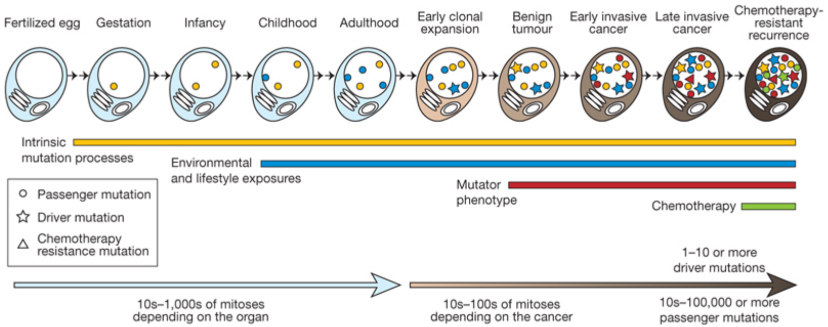
1 body $= 10^{14}$ human cells (and 100x more non-human cells)
1 cell $= 6 \times 10^9$ ACGT coding for $20,000+$ genes

Cancer



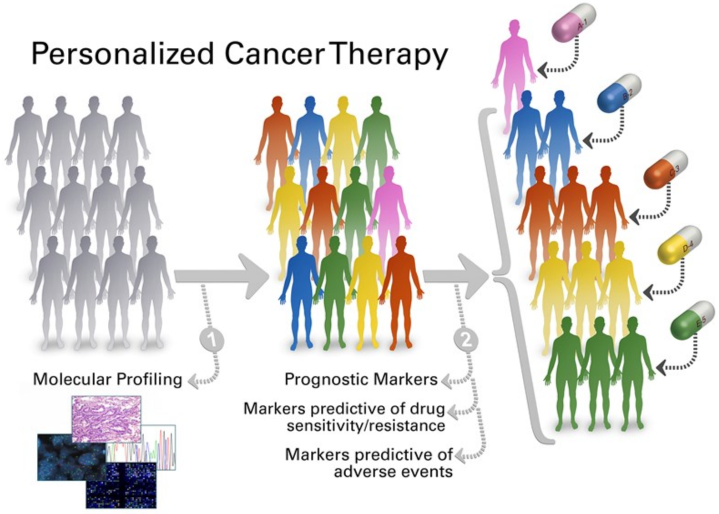
`http://rise.duke.edu/seek/pages/page.html?0205`

Each cancer has a unique history


Fertilized egg
Gestation
Infancy
Childhood
Adulthood
Early clonal expansion
Benign tumour
Early invasive cancer
Late invasive cancer
Chemotherapy-resistant recurrence
Intrinsic mutation processes
Environmental and lifestyle exposures
Mutator phenotype
Chemotherapy
○ Passenger mutation
☆ Driver mutation
△ Chemotherapy resistance mutation
1–10 or more driver mutations
10s–1,000s of mitoses depending on the organ
10s–100s of mitoses depending on the cancer
10s–100,000 or more passenger mutations

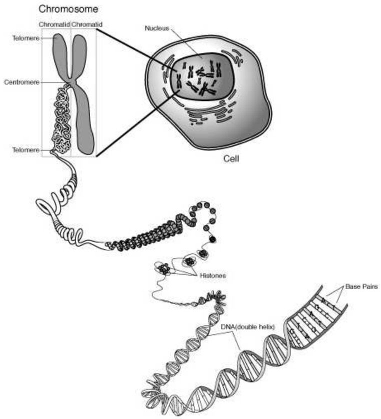
Stratton et al. [2009]

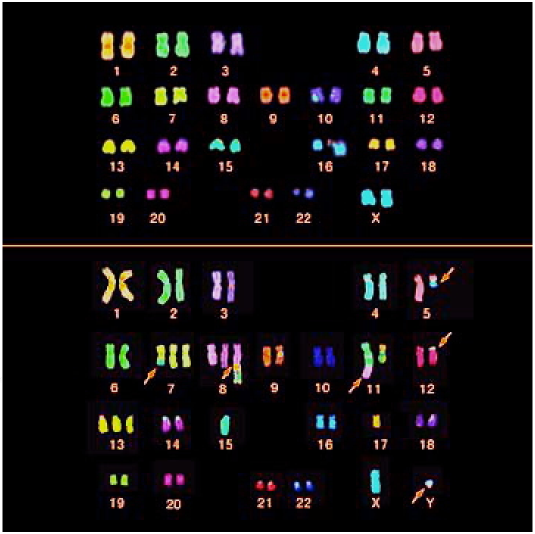
...and responds differently to different treatments


Personalized Cancer Therapy
A-1
B-2
C-3
D-4
E-5
1
2
Molecular Profiling
Prognostic Markers
Markers predictive of drug sensitivity/resistance
Markers predictive of adverse events

https://pct.mdanderson.org

Chromosomic aberrations in cancer

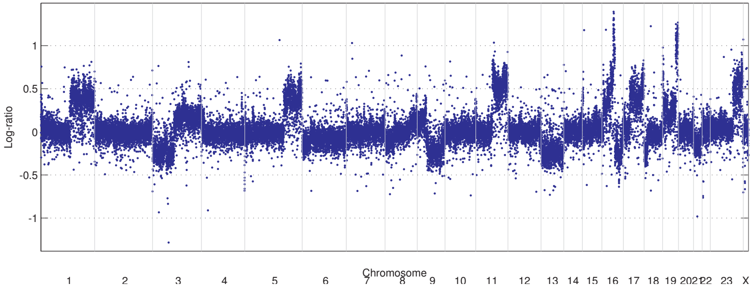




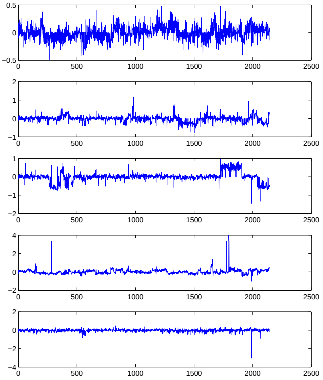
Comparative Genomic Hybridization (CGH)

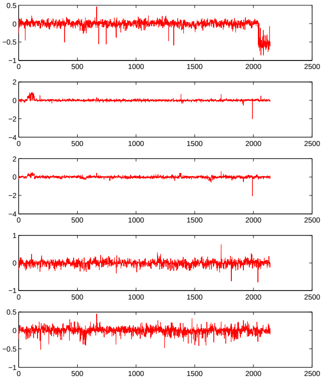
Motivation

- Comparative genomic hybridization (CGH) data measure the DNA copy number along the genome
- Very useful, in particular in cancer research to observe systematically variants in DNA content



Cancer prognosis: can we predict the future evolution?

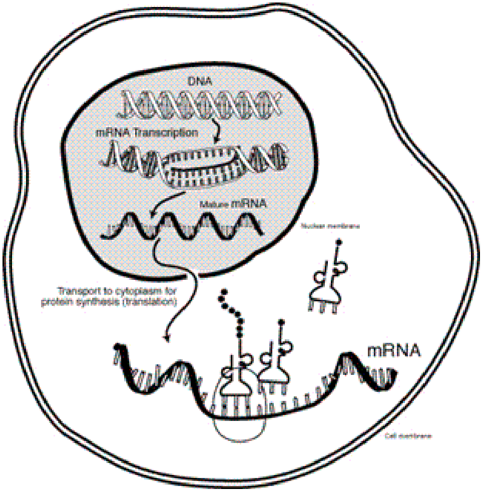


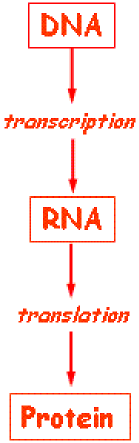


Problem 1

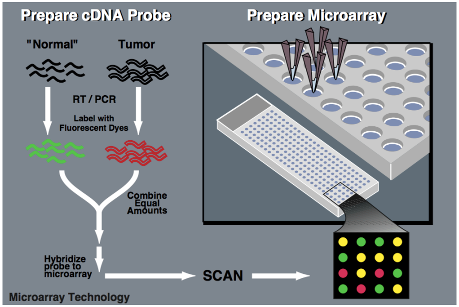
From a CGH profile, can we predict whether a melanoma will relapse (left) or not (right)?

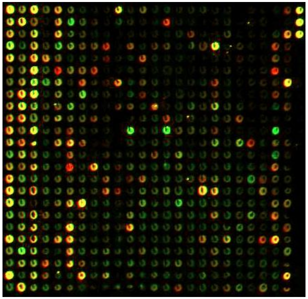
DNA → RNA → protein

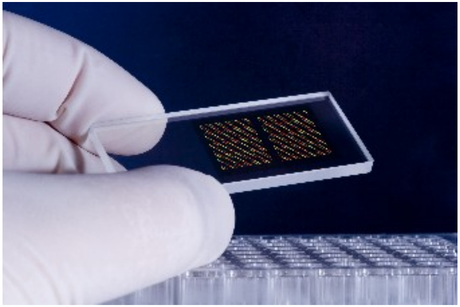




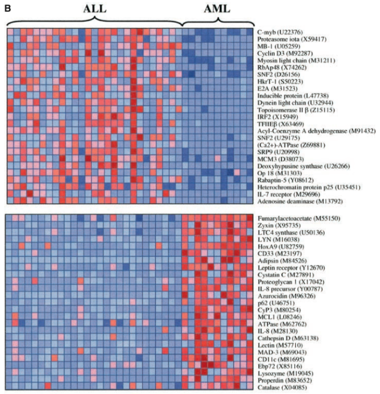
Tissue profiling with DNA chips







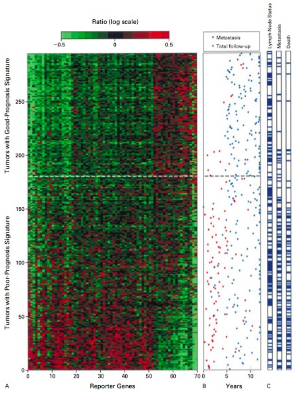
Use in diagnosis

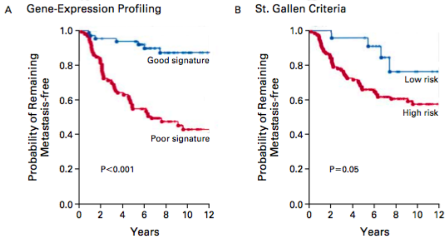


Problem 2

Given the expression profile of a leukemia, is it an acute lymphocytic or myeloid leukemia (ALL or AML)?

Use in prognosis





No. at Risk

	0	2	4	6	8	10	12
Good signature	60	57	54	45	31	22	12
Poor signature	91	72	55	41	26	17	9

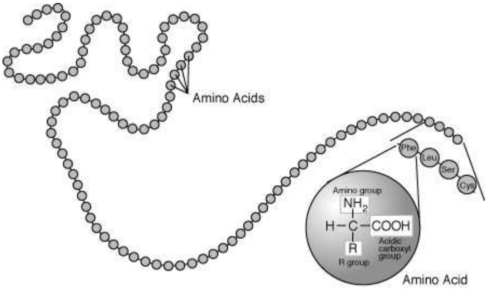
No. at Risk

	0	2	4	6	8	10	12
Low risk	22	22	21	17	9	5	2
High risk	129	107	88	69	48	34	19

Problem 3

Given the expression profile of a breast cancer, is the risk of relapse within 5 years high?

Proteins



A : Alanine	V : Valine	L : Leucine
F : Phenylalanine	P : Proline	M : Methionine
E : Acide glutamique	K : Lysine	R : Arginine
T : Threonine	C : Cysteine	N : Asparagine
H : Histidine	V : Thyrosine	W : Tryptophane
I : Isoleucine	S : Serine	Q : Glutamine
D : Acide aspartique	G : Glycine	

Protein annotation

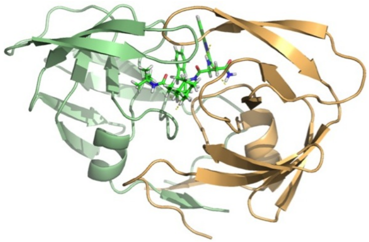
Data available

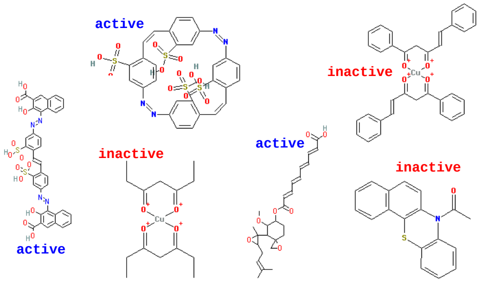
- Secreted proteins:
  MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNIEA...
  MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...
  MALHTVLIMLSLLPMLEAQNPEHANITIGEPITNETLGWL...
  ...
- Non-secreted proteins:
  MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVNLGVG...
  MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...
  MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..
  ...

Problem 4

Given a newly sequenced protein, is it secreted or not?

Drug discovery

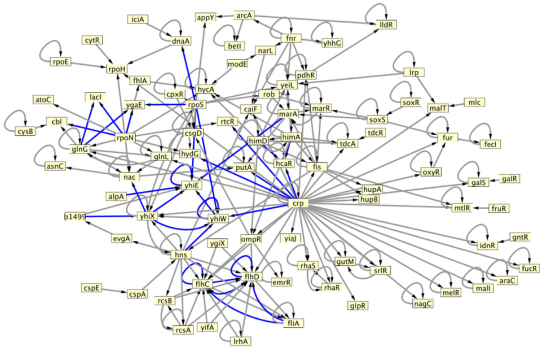


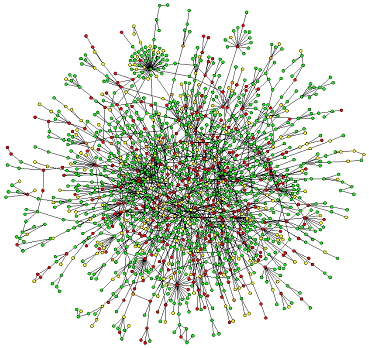


Problem 5

Given a new candidate molecule, is it likely to be active?

Gene network inference

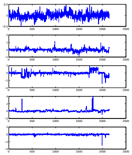
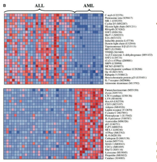


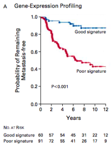


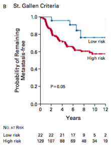
Problem 6

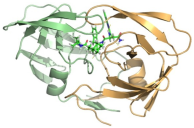
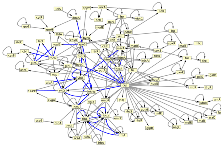
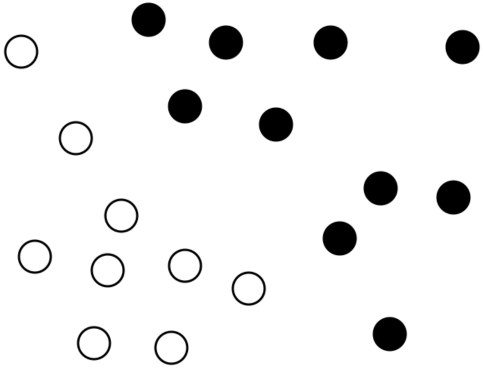
Given known interactions, can we infer new ones?

A common topic...

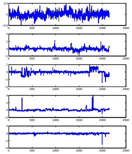



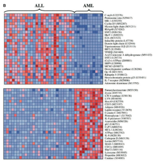


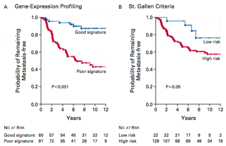


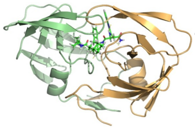




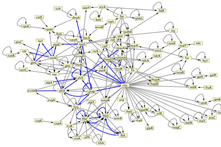
A common topic...

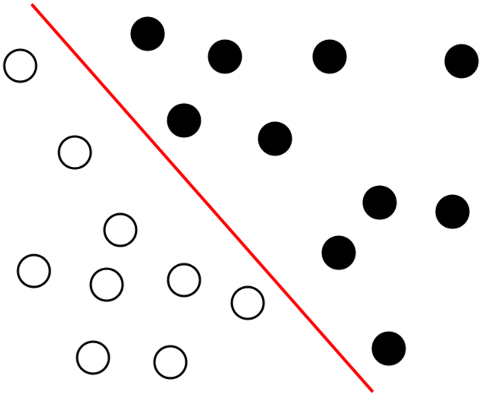




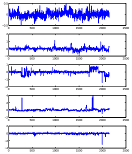


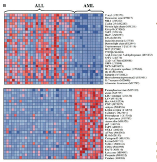


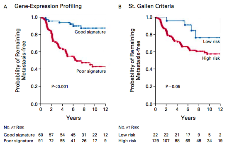


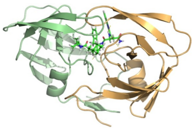


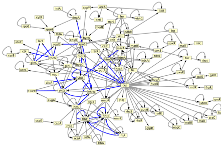
A common topic...

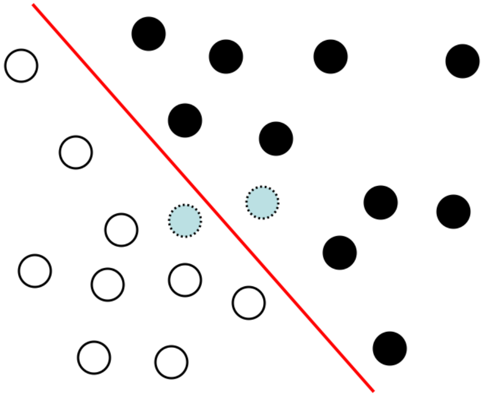




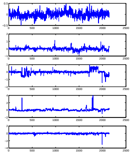
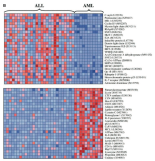
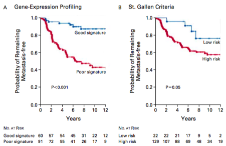
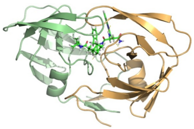
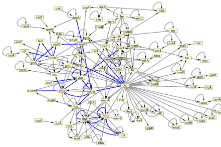
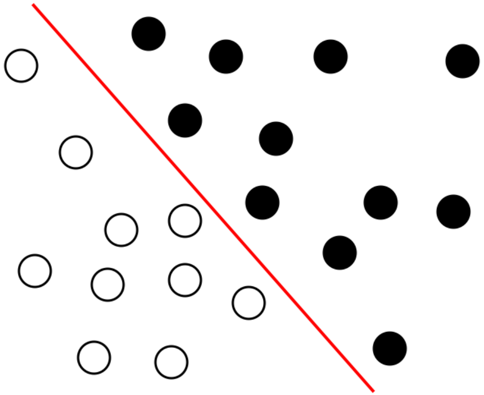




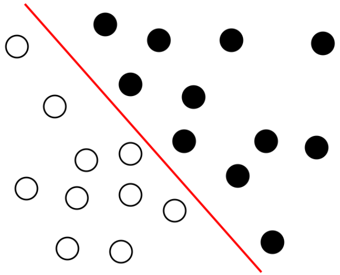




A common topic...

Pattern recognition, *aka* supervised classification

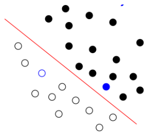


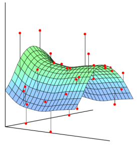
Challenges

- High dimension
- Few samples
- Structured data
- Heterogeneous data
- Prior knowledge
- Fast and scalable implementations
- Interpretable models

Outline

More formally





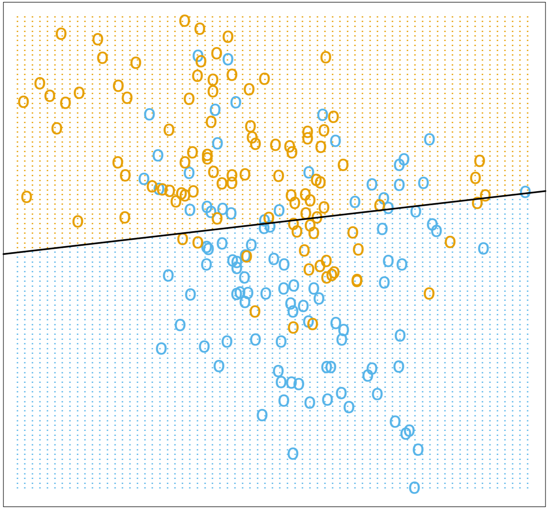
Input

- $\mathcal{X}$ the space of patterns or data (typically, $\mathcal{X} = \mathbb{R}^p$)
- $\mathcal{Y}$ the space of response or labels
    - Binary classification or pattern recognition : $\mathcal{Y} = \{-1, 1\}$
    - Regression : $\mathcal{Y} = \mathbb{R}$
    - Structured output: $\mathcal{Y}$ general
- $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ a training set in $(\mathcal{X} \times \mathcal{Y})^n$

Output

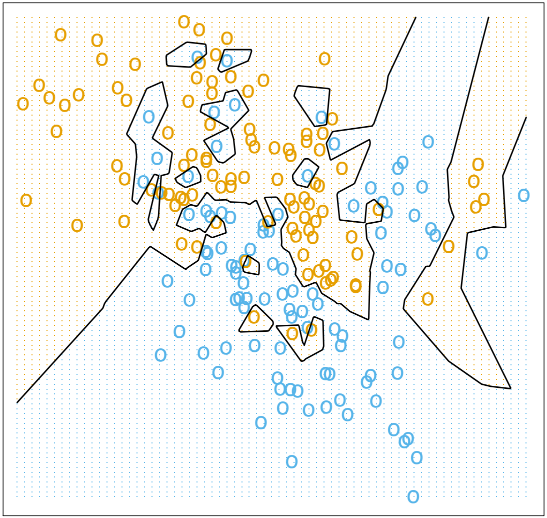
- A function $f : \mathcal{X} \to \mathcal{Y}$ to predict the output associated to any new pattern $x \in \mathcal{X}$ by $f(x)$

Simple example 1 : ordinary least squares (OLS)



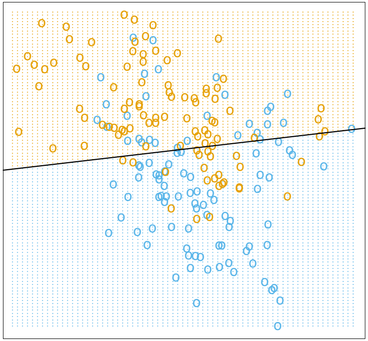
(Hastie et al. The elements of statistical learning. Springer, 2001.)

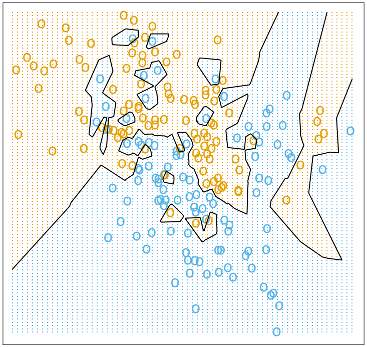
Simple example 2 : 1-nearest neighbor (1-NN)



(Hastie et al. The elements of statistical learning. Springer, 2001.)

What’s wrong?





- OLS: the linear separation is not appropriate = "large bias"
- 1-NN: the classifier seems too unstable = "large variance"

The fundamental "bias-variance" trade-off

- Assume $Y = f(X) + \epsilon$, where ϵ is some noise
- From the training set $\mathcal{S}$ we estimate the predictor $\hat{f}$
- On a new point x_0, we predict $\hat{f}(x_0)$ but the "true" observation will be $Y_0 = f(x_0) + \epsilon$
- On average, we make an error of:

$$
\begin{aligned}
&E_{\epsilon,\mathcal{S}}\left(Y_0 - \hat{f}(x_0)\right)^2 \\
&\quad = E_{\epsilon,\mathcal{S}}\left(f(x_0) + \epsilon - \hat{f}(x_0)\right)^2 \\
&\quad = E\epsilon^2 + E_{\mathcal{S}}\left(f(x_0) - \hat{f}(x_0)\right)^2 \\
&\quad = E\epsilon^2 + \left(f(x_0) - E_{\mathcal{S}}\hat{f}(x_0)\right)^2 + E_{\mathcal{S}}\left(\hat{f}(x_0) - E_{\mathcal{S}}\hat{f}(x_0)\right)^2 \\
&\quad = \mathit{noise} + \mathit{bias}^2 + \mathit{variance}
\end{aligned}
$$

Back to OLS

- Parametric model for $\beta \in \mathbb{R}^{p+1}$:

$$f_\beta(X) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i = X^\top \beta$$

- Estimate $\hat{\beta}$ from training data to minimize

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f_\beta(x_i))^2 = (\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta)$$

- Solution if $\mathbf{X}^\top \mathbf{X}$ is non-singular:

$$\hat{\beta} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}$$

Optimality of OLS

Gauss-Markov theorem

- Assume $\mathbf{Y} = \mathbf{X}\beta + \epsilon$, where $E\epsilon = 0$ and $E\epsilon\epsilon^\top = \sigma^2 I$.
- Then the least squares estimator $\hat{\beta}$ is BLUE (best linear unbiased estimator), i.e., for any other estimator $\tilde{\beta} = C\mathbf{Y}$ with $E\tilde{\beta} = \beta$,

$$Var(\hat{\beta}) \leq Var(\tilde{\beta})$$

Optimality of OLS

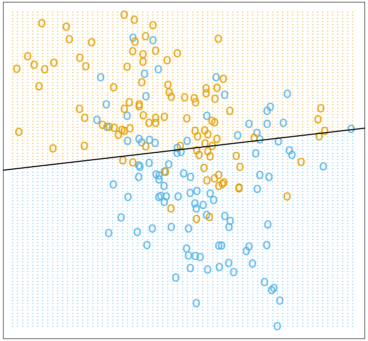
Gauss-Markov theorem

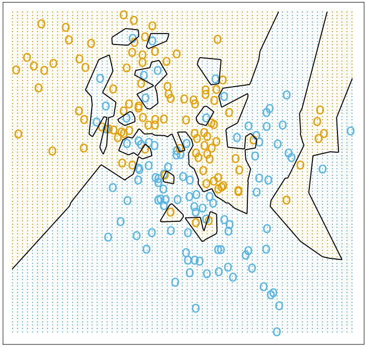
- Assume $\mathbf{Y} = \mathbf{X}\beta + \epsilon$, where $E\epsilon = 0$ and $E\epsilon\epsilon^\top = \sigma^2 I$.
- Then the least squares estimator $\hat{\beta}$ is BLUE (best linear unbiased estimator), i.e., for any other estimator $\tilde{\beta} = C\mathbf{Y}$ with $E\tilde{\beta} = \beta$,

$$Var(\hat{\beta}) \leq Var(\tilde{\beta})$$

Nevertheless, if variance may be very large, we may have smaller total risk by increasing bias to decrease variance

The curse of dimensionality


Small dimension


Large dimension

In high dimensions, variance dominates, even for simple linear estimators.
BLUE estimators are useless.

A solution: shrinkage estimators

1. Define a large family of "candidate classifiers", e.g., linear predictors:

$$f_\beta(x) = \beta^\top x \quad \text{for } x \in \mathbb{R}^p$$

A solution: shrinkage estimators

1. Define a large family of "candidate classifiers", e.g., linear predictors:

$$f_\beta(x) = \beta^\top x \quad \text{for } x \in \mathbb{R}^p$$

2. For any candidate classifier f_β, quantify how "good" it is on the training set with some empirical risk, e.g.:

$$R(\beta) = \frac{1}{n}\sum_{i=1}^{n}(f_\beta(x_i) - y_i)^2 \,.$$

A solution: shrinkage estimators

1. Define a large family of "candidate classifiers", e.g., linear predictors:

$$f_\beta(x) = \beta^\top x \quad \text{for } x \in \mathbb{R}^p$$

2. For any candidate classifier f_β, quantify how "good" it is on the training set with some empirical risk, e.g.:

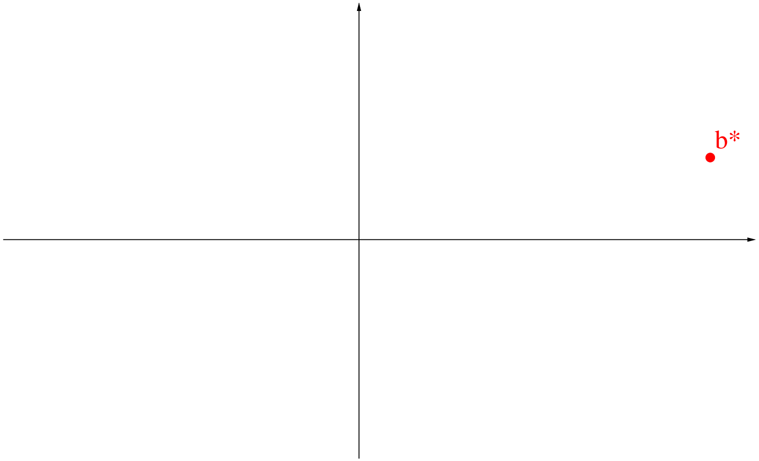
$$R(\beta) = \frac{1}{n}\sum_{i=1}^{n}(f_\beta(x_i) - y_i)^2 \,.$$

3. Choose β that achieves the minimium empirical risk, subject to some constraint:

$$\min_\beta R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C \,.$$

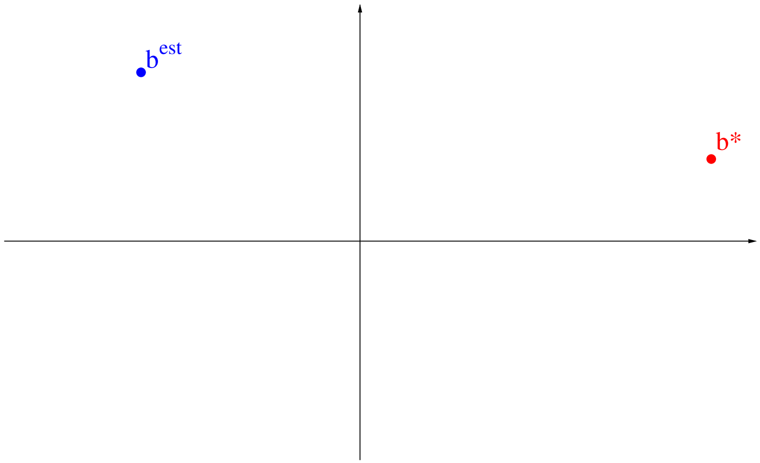
Why skrinkage classifiers?

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



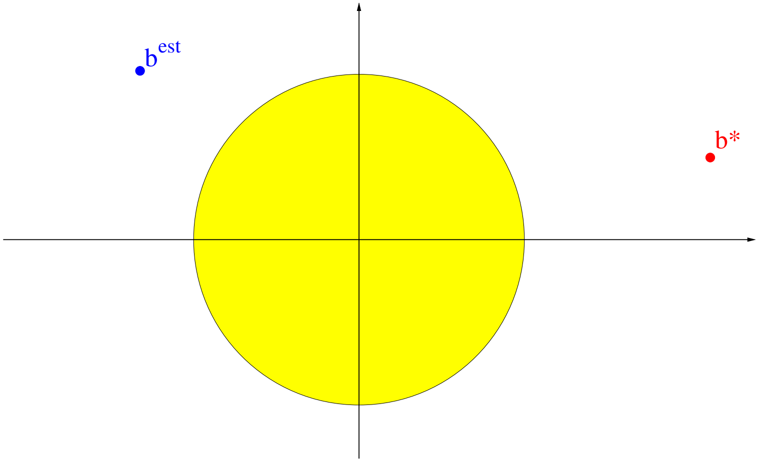
Why skrinkage classifiers?

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



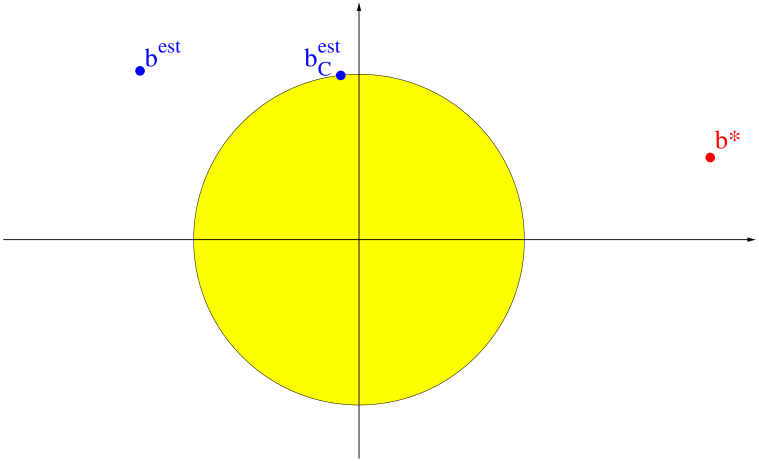
Why skrinkage classifiers?

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



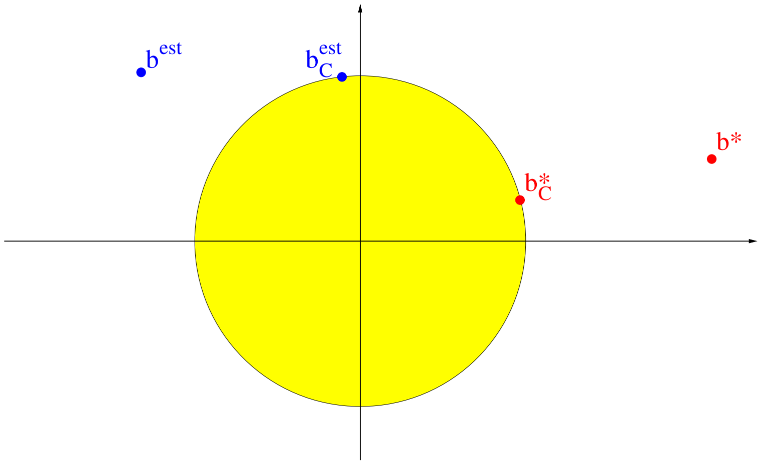
Why skrinkage classifiers?

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \le C\,.$$



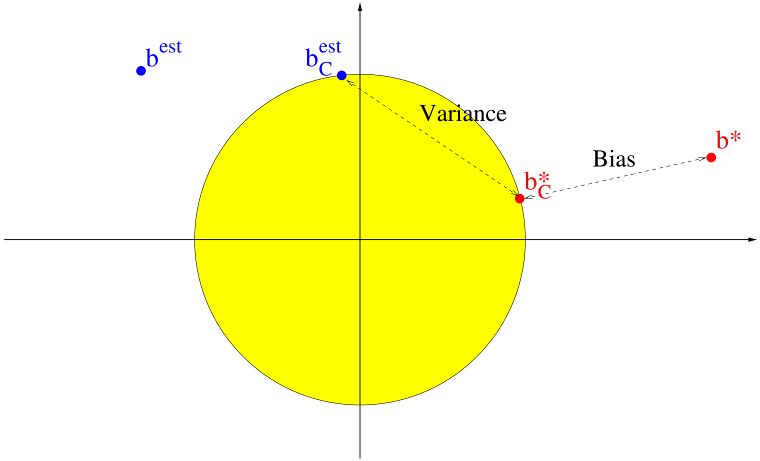
Why skrinkage classifiers?

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



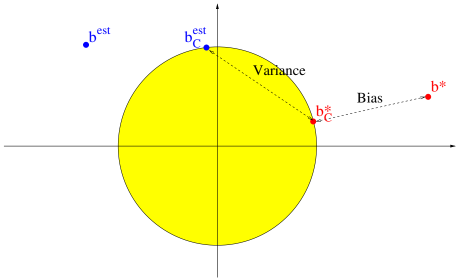
Why skrinkage classifiers?

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



Why skrinkage classifiers?

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C \,.$$

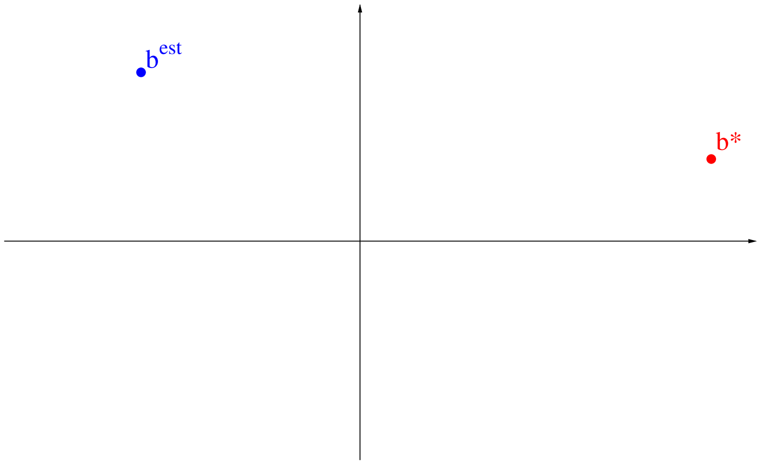


- "Increases bias and decreases variance"
- Equivalent formulation:

$$\min_{\beta} R(\beta) + \lambda\Omega(\beta) \,.$$

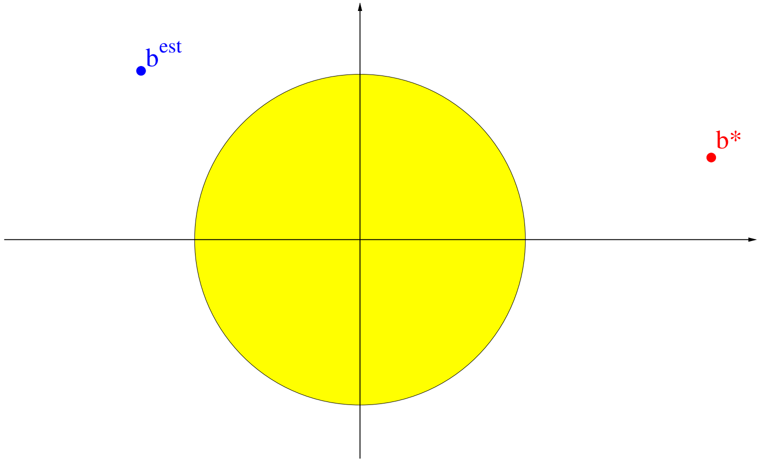
Choice of Ω can decrease the bias

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



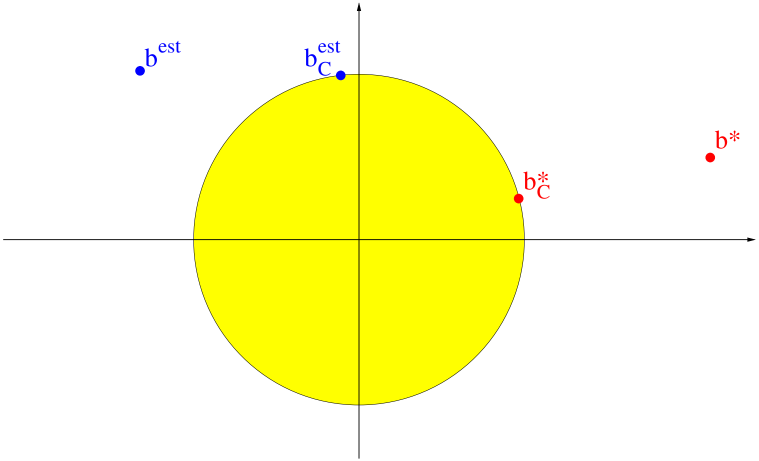
Choice of Ω can decrease the bias

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



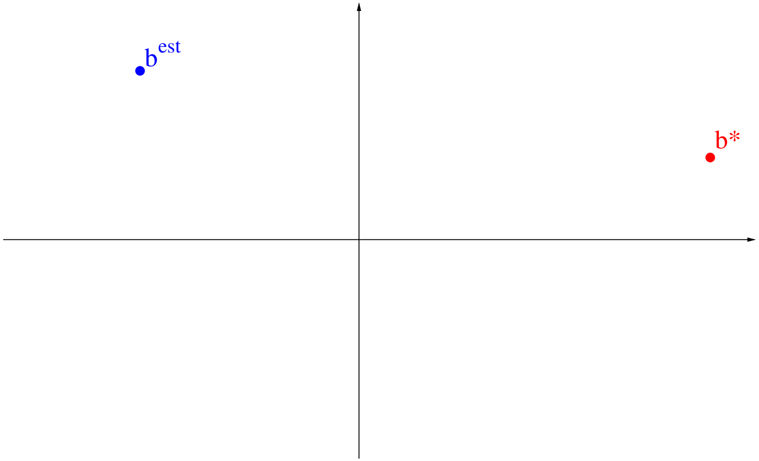
Choice of Ω can decrease the bias

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



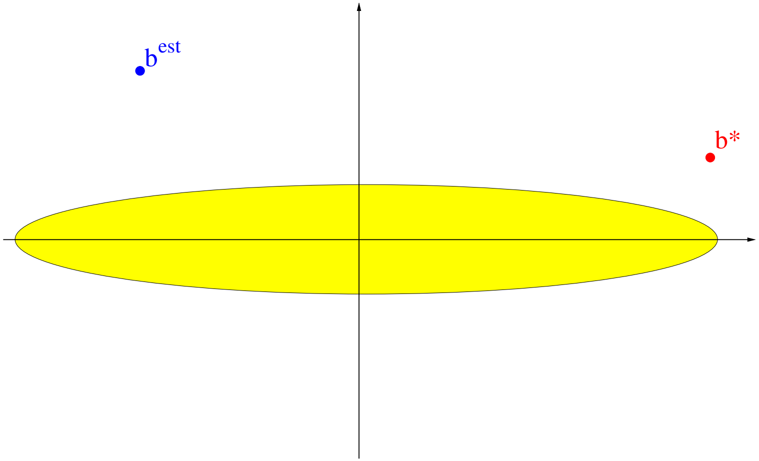
Choice of Ω can decrease the bias

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



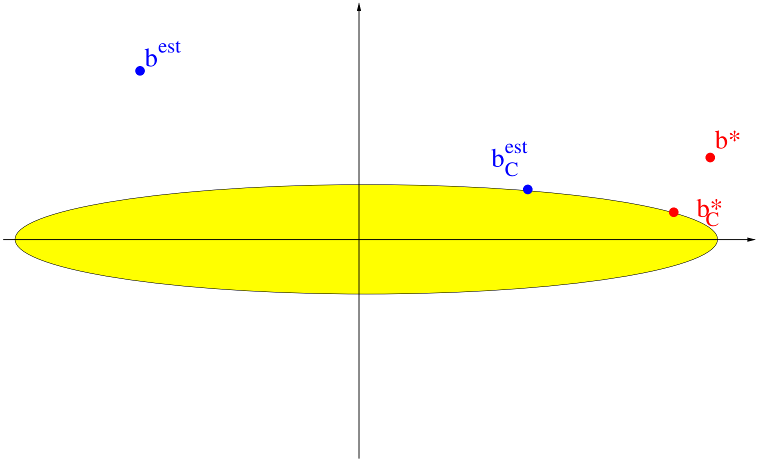
Choice of Ω can decrease the bias

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$



Choice of Ω can decrease the bias

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C\,.$$

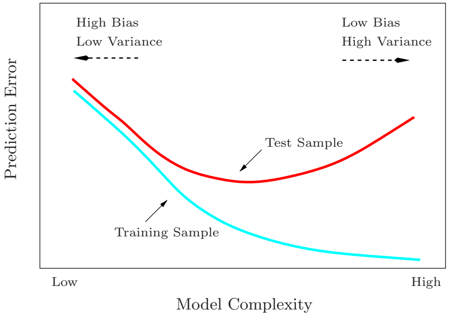


Choice of C or λ: structured regression and model selection

- Define a family of function classes $\mathcal{F}_\lambda$, where λ controls the "complexity"
- For each λ, define

$$\hat{f}_\lambda = \underset{\mathcal{F}_\lambda}{\operatorname{argmin}}\ EPE(f)$$

- Select $\hat{f} = \hat{f}_{\hat{\lambda}}$ to minimize the bias-variance tradeoff.



Cross-validation

A simple and systematic procedure to estimate the risk (and to optimize the model's parameters)

1. Randomly divide the training set (of size n) into K (almost) equal portions, each of size K/n
2. For each portion, fit the model with different parameters on the $K-1$ other groups and test its performance on the left-out group
3. Average performance over the K groups, and take the parameter with the smallest average performance.

Taking $K = 5$ or 10 is recommended as a good default choice.

Summary

1. Many problems in computational biology and medicine can be formulated as high-dimensional classification or regression tasks
2. The total error of a learning system is the sum of a bias and a variance error
3. In high dimension, the variance term often dominates
4. Shrinkage methods allow to control the bias/variance trade-off
5. The choice of the penalty is where we can put prior knowledge to decrease bias

Choosing or designing a penalty...

$$\min_{\beta} R(\beta) \quad \text{subject to} \quad \Omega(\beta) \leq C \,.$$

We will only focus on convex penalties, which lead to efficient algorithms. We will touch upon two important families of penalties:

1. Smooth convex penalty: ridge regression, SVM, kernels...
2. Nonsmooth convex penalty: lasso, group lasso, fused lasso,...

How to Make Paper Lanterns

Looking for instructions on how to make paper lanterns? My husband designed an easy template for making paper lanterns in a cute round shape. They look a bit oriental, don't you think?

Outline

Outline

Ridge regression [Hoerl and Kennard, 1970]

1. Consider the set of linear predictors:

$$\forall \beta \in \mathbb{R}^p , \quad f_\beta(x) = \beta^\top x \quad \text{for } x \in \mathbb{R}^p .$$

Ridge regression [Hoerl and Kennard, 1970]

1. Consider the set of linear predictors:

$$\forall \beta \in \mathbb{R}^p, \quad f_\beta(x) = \beta^\top x \quad \text{for } x \in \mathbb{R}^p.$$

2. Consider the mean square error (MSE) as empirical risk:

$$R(\beta) = \frac{1}{n}\sum_{i=1}^{n}(f_\beta(x_i) - y_i)^2.$$

Ridge regression [Hoerl and Kennard, 1970]

1. Consider the set of linear predictors:

$$\forall \beta \in \mathbb{R}^p, \quad f_\beta(x) = \beta^\top x \quad \text{for } x \in \mathbb{R}^p .$$

2. Consider the mean square error (MSE) as empirical risk:

$$R(\beta) = \frac{1}{n} \sum_{i=1}^{n} (f_\beta(x_i) - y_i)^2 .$$

3. Consider the Euclidean norm as a penalty:

$$\Omega(\beta) = \| \beta \|_2^2 = \sum_{i=1}^{p} \beta_i^2 .$$

Solution

- Let $X = (x_1, \ldots, x_n)$ the $n \times p$ data matrix, and $Y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^p$ the response vector.

Solution

- Let $X = (x_1, \ldots, x_n)$ the $n \times p$ data matrix, and $Y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^p$ the response vector.
- The penalized risk can be written in matrix form:

$$\begin{aligned} R(\beta) + \lambda\Omega(\beta) &= \frac{1}{n}\sum_{i=1}^{n}\left(f_\beta\left(x_i\right) - x_i\right)^2 + \lambda\sum_{i=1}^{p}\beta_i^2 \\ &= \frac{1}{n}\left(Y - X\beta\right)^\top\left(Y - X\beta\right) + \lambda\beta^\top\beta\,. \end{aligned}$$

Solution

- Let $X = (x_1, \dots, x_n)$ the $n \times p$ data matrix, and $Y = (y_1, \dots, y_n)^\top \in \mathbb{R}^p$ the response vector.
- The penalized risk can be written in matrix form:

$$\begin{aligned} R(\beta) + \lambda\Omega(\beta) &= \frac{1}{n}\sum_{i=1}^{n} \left(f_\beta\left(x_i\right) - x_i\right)^2 + \lambda \sum_{i=1}^{p} \beta_i^2 \\ &= \frac{1}{n}\left(Y - X\beta\right)^\top \left(Y - X\beta\right) + \lambda\beta^\top\beta\,. \end{aligned}$$

- Explicit minimizer:

$$\hat{\beta}_\lambda^{\text{ridge}} = \arg\min_{\beta\in\mathbb{R}^p} \left\{R(\beta) + \lambda\Omega(\beta)\right\} = \left(X^\top X + \lambda n I\right)^{-1} X^\top Y\,.$$

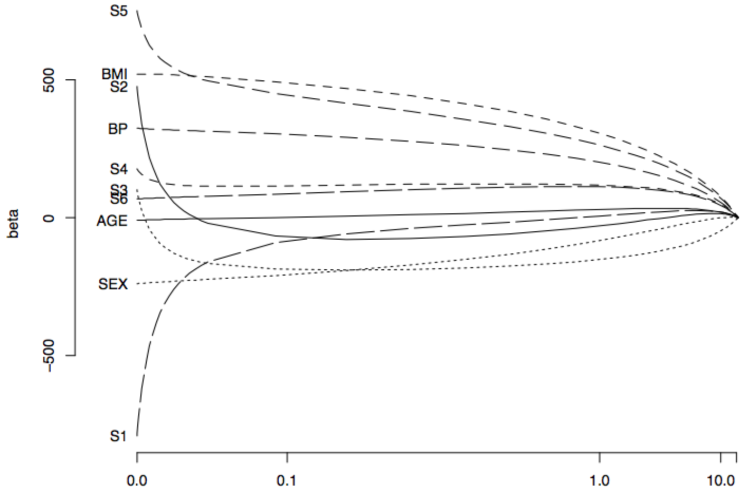
Limit cases

$$\hat{\beta}_{\lambda}^{\text{ridge}} = \left(X^{\top}X + \lambda n I\right)^{-1} X^{\top} Y$$

Corollary

- As $\lambda \to 0$, $\hat{\beta}_{\lambda}^{\text{ridge}} \to \hat{\beta}^{\text{OLS}}$ (low bias, high variance).
- As $\lambda \to +\infty$, $\hat{\beta}_{\lambda}^{\text{ridge}} \to 0$ (high bias, low variance).

Ridge regression example



[From Hastie et al., 2001]

Ridge regression with correlated features

Ridge regression is particularly useful in the presence of correlated features:

```
> library(MASS) # for the lm.ridge command
> x1 <- rnorm(20)
> x2 <- rnorm(20,mean=x1,sd=.01)
> y <- rnorm(20,mean=3+x1+x2)
> lm(y~x1+x2)$coef
(Intercept)          x1          x2
   3.070699   25.797872  -23.748019
> lm.ridge(y~x1+x2,lambda=1)
                x1       x2
3.066027 1.015862 0.956560
```

Generalization: ℓ_2-regularized learning

- A general ℓ_2-penalized estimator is of the form

$$\min_{\beta} \left\{ R(\beta) + \lambda \|\beta\|_2^2 \right\} , \qquad (1)$$

  where

$$R(\beta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_\beta(x_i), y_i)$$

  for some general loss functions ℓ.

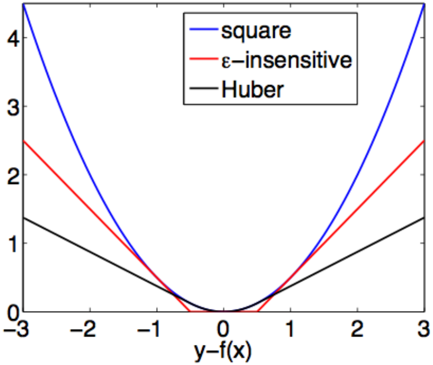
- Ridge regression corresponds to the particular loss

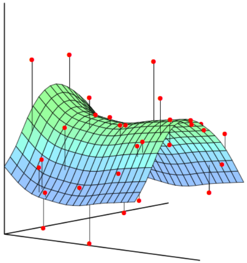
$$\ell(u, y) = (u - y)^2 \, .$$

- For general, convex losses, the problem (1) is strictly convex and has a unique global minimum, which can usually be found by numerical algorithms for convex optimization.

Losses for regression

- Square loss : $\ell(u, y) = (u - y)^2$
- ϵ-insensitive loss : $\ell(u, y) = (|\, u - y \,| - \epsilon)_+$
- Huber loss : mixed quadratic/linear



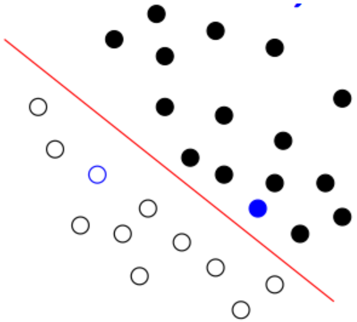


Outline

Binary classification

Setting

- $\mathcal{X} = \mathbb{R}^p$ set of inputs
- $\mathcal{Y} = \{-1, 1\}$ binary outputs
- $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ a training set in $(\mathcal{X} \times \mathcal{Y})^n$
- Goal: Estimate a function $f : \mathcal{X} \to \mathbb{R}$ to predict Y by $sign(f(X))$



The 0/1 loss

- The 0/1 loss measures if a prediction is correct or not:

$$\ell_{0/1}(f(\mathbf{x}), y)) = \mathbf{1}(yf(\mathbf{x}) < 0) = \begin{cases} 0 & \text{if } y = sign(f(\mathbf{x})) \\ 1 & \text{otherwise.} \end{cases}$$

- It is them tempting to learn $f_\beta(x) = \beta^\top x$ by solving:

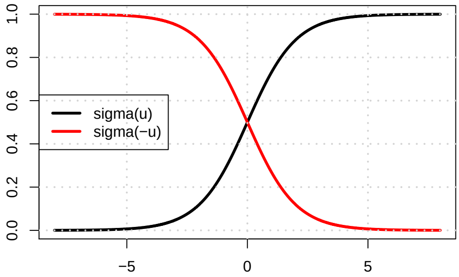
$$\min_{\beta \in \mathbb{R}^p} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell_{0/1}(f_\beta(\mathbf{x}_i), y_i)}_{\text{misclassification rate}} + \underbrace{\lambda \| \beta \|_2^2}_{\text{regularization}}$$

- However:
  - The problem is non-smooth, and typically NP-hard to solve
  - The regularization has no effect since the 0/1 loss is invariant by scaling of β
  - In fact, no function achieves the minimum when $\lambda > 0$ (*why?*)

The logistic loss

- An alternative is to define a probabilistic model of y parametrized by $f(\mathbf{x})$, e.g.:

$$\forall \mathbf{y} \in \{-1, 1\}\,, \quad p\left(y \mid f(\mathbf{x})\right) = \frac{1}{1 + e^{-yf(\mathbf{x})}} = \sigma\left(yf(\mathbf{x})\right)$$



- The logistic loss is the negative conditional likelihood:

$$\ell_{logistic}\left(f(\mathbf{x}), y\right) = -\ln p\left(y \mid f\left(\mathbf{x}\right)\right) = \ln\left(1 + e^{-yf(\mathbf{x})}\right)$$

Ridge logistic regression [Le Cessie and van Houwelingen, 1992]

$$\min_{\beta \in \mathbb{R}^p} J(\beta) = \frac{1}{n} \sum_{i=1}^{n} \ln \left(1 + e^{-y_i \beta^\top x_i} \right) + \lambda \|\beta\|_2^2$$

- Can be interpreted as a regularized conditional maximum likelihood estimator
- No explicit solution, but smooth convex optimization problem that can be solved numerically

Solving ridge logistic regression

$$\min_{\beta} J(\beta) = \frac{1}{n}\sum_{i=1}^{n} \ln\left(1 + e^{-y_i\beta^\top x_i}\right) + \lambda\|\beta\|_2^2$$

No explicit solution, but convex problem with:

$$\begin{aligned}
\nabla_\beta J(\beta) &= -\frac{1}{n}\sum_{i=1}^{n} \frac{y_i x_i}{1 + e^{y_i\beta^\top x_i}} + 2\lambda\beta \\
&= -\frac{1}{n}\sum_{i=1}^{n} y_i\left[1 - P_\beta(y_i \mid x_i)\right] x_i + 2\lambda\beta \\
\nabla_\beta^2 J(\beta) &= \frac{1}{n}\sum_{i=1}^{n} \frac{x_i x_i^\top e^{y_i\beta^\top x_i}}{\left(1 + e^{y_i\beta^\top x_i}\right)^2} + 2\lambda I \\
&= \frac{1}{n}\sum_{i=1}^{n} P_\beta(1 \mid x_i)\left(1 - P_\beta(1 \mid x_i)\right) x_i x_i^\top + 2\lambda I
\end{aligned}$$

Solving ridge logistic regression (cont.)

$$\min_{\beta} J(\beta) = \frac{1}{n}\sum_{i=1}^{n} \ln\left(1 + e^{-y_i \beta^\top x_i}\right) + \lambda \|\beta\|_2^2$$

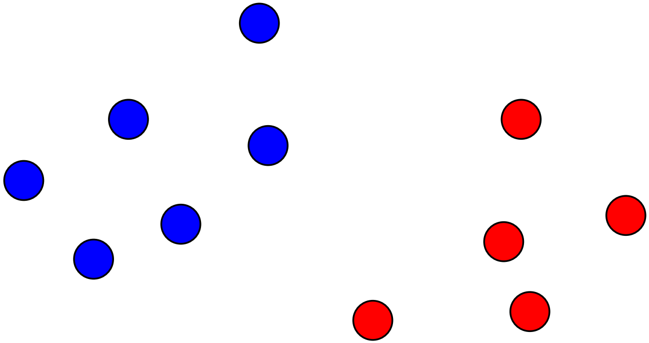
- The solution can then be found by Newton-Raphson iterations:

$$\beta^{new} \leftarrow \beta^{old} - \left[\nabla_\beta^2 J\left(\beta^{old}\right)\right]^{-1} \nabla_\beta J\left(\beta^{old}\right) .$$

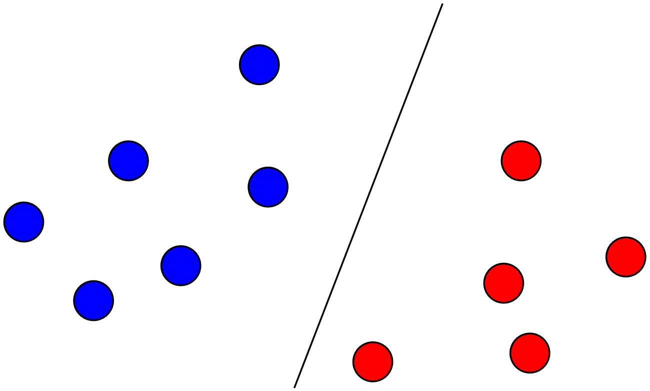
- Each step is equivalent to solving a weighted ridge regression problem (*left as exercise*)
- This method is therefore called iteratively reweighted least squares (IRLS).

Outline

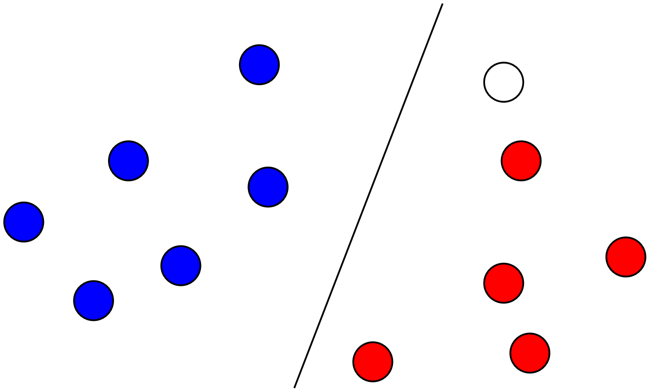
Linear classifier



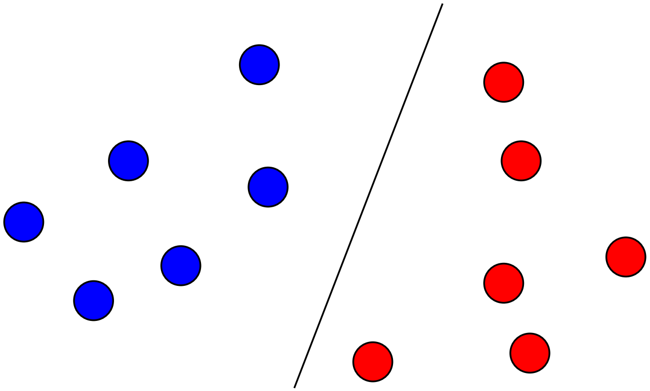
Linear classifier



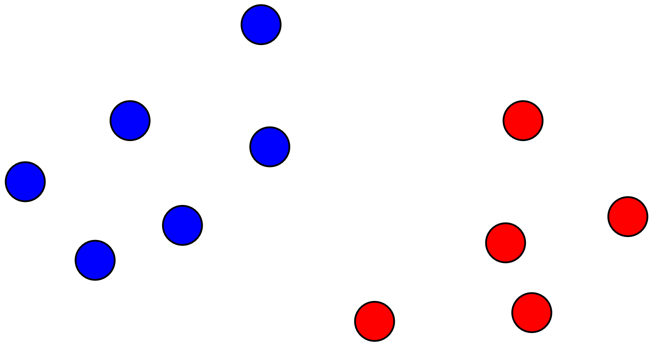
Linear classifier



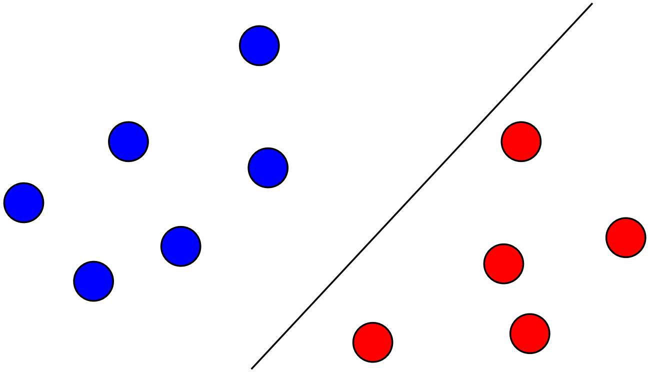
Linear classifier



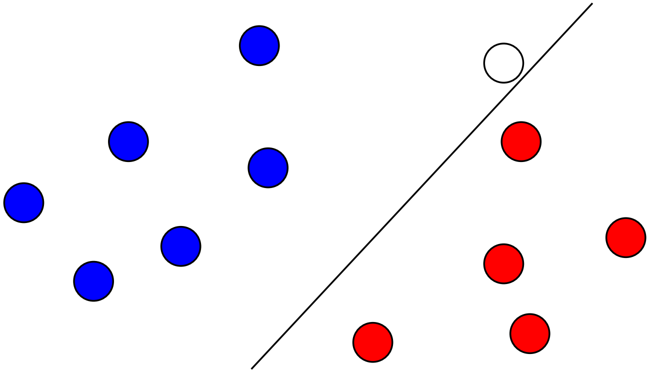
Linear classifier



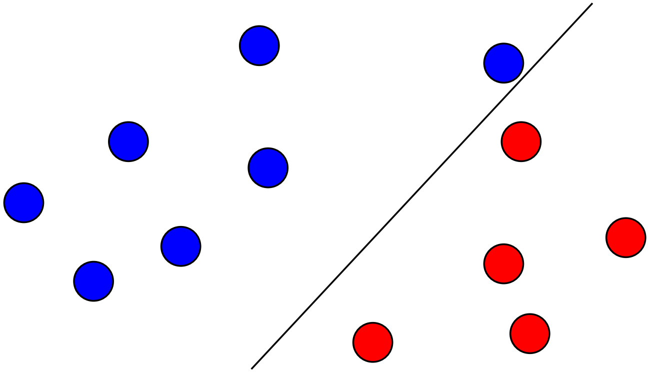
Linear classifier



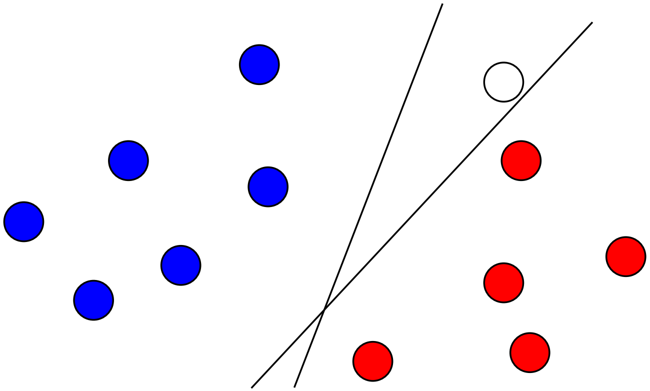
Linear classifier



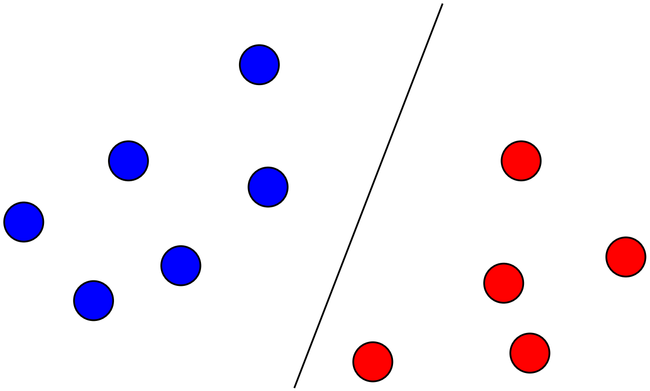
Linear classifier



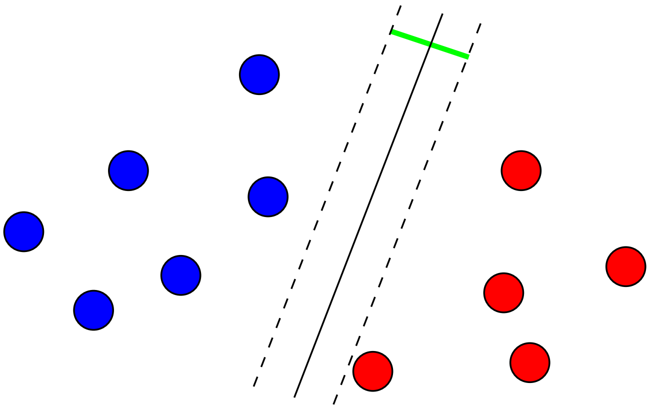
Which one is better?



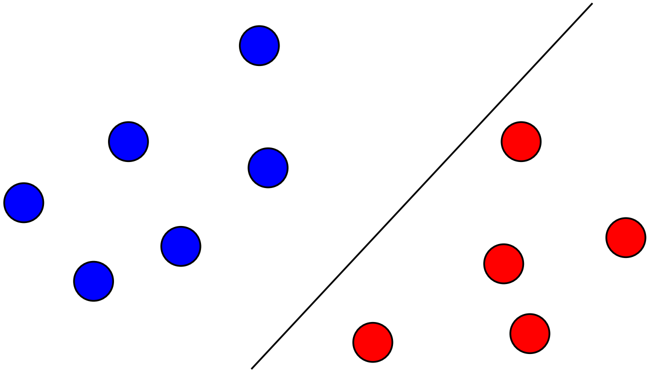
The margin of a linear classifier



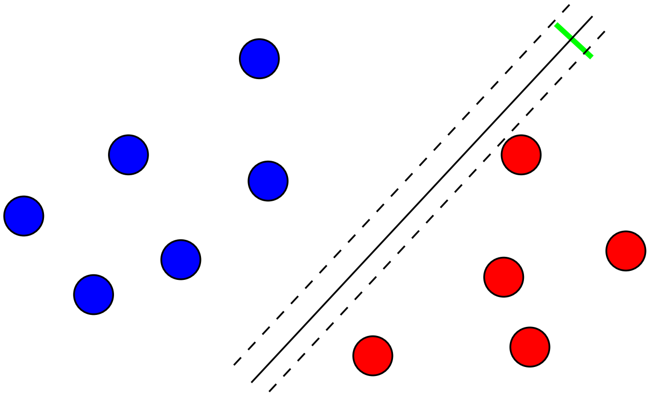
The margin of a linear classifier



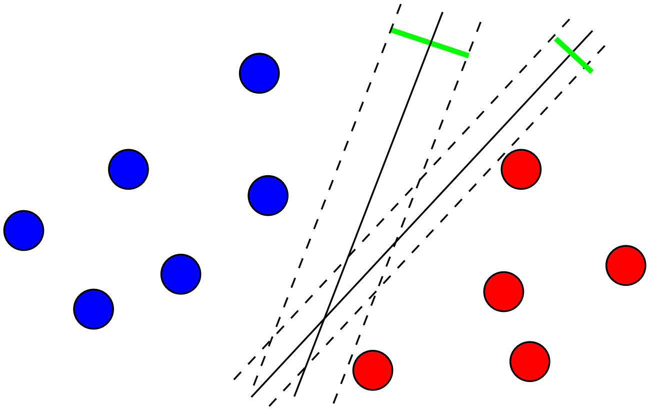
The margin of a linear classifier



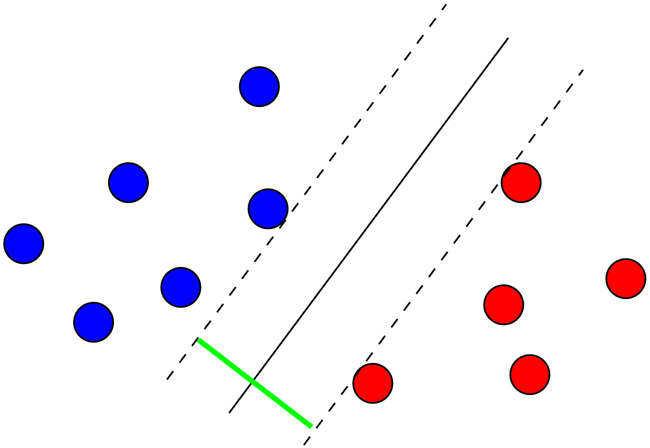
The margin of a linear classifier



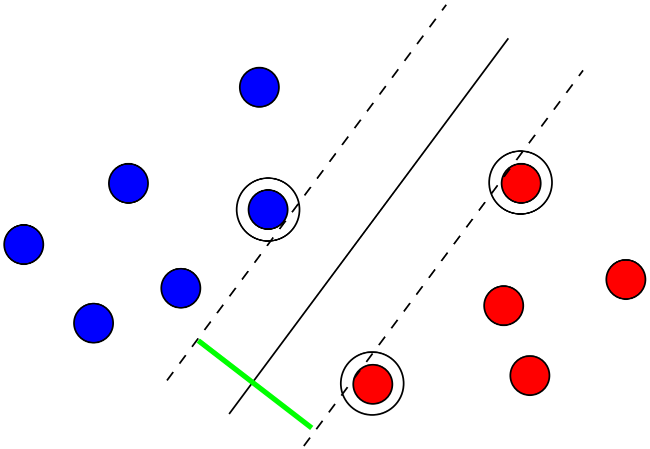
The margin of a linear classifier



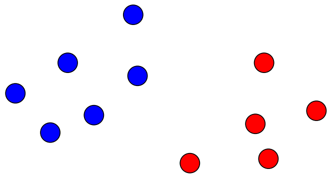
Largest margin classifier (*hard-margin SVM*)



Support vectors



More formally



- The training set is a finite set of n data/class pairs:

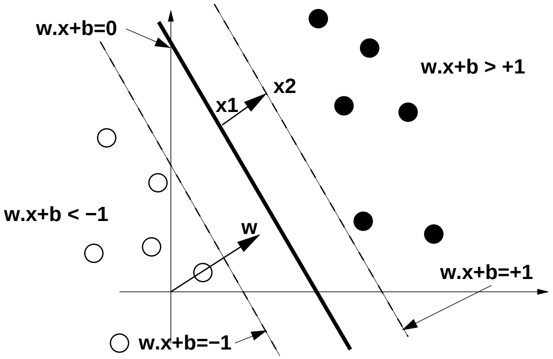
$$\mathcal{S} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\} \ ,$$

  where $\vec{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$.
- We assume (for the moment) that the data are linearly separable, i.e., that there exists $(\vec{w}, b) \in \mathbb{R}^p \times \mathbb{R}$ such that:

$$\begin{cases} \vec{w}.\vec{x}_i + b > 0 & \text{if } y_i = 1\,, \\ \vec{w}.\vec{x}_i + b < 0 & \text{if } y_i = -1\,. \end{cases}$$

How to find the largest separating hyperplane?

For a given linear classifier $f(x) = \vec{w}.\vec{x} + b$ consider the "tube" defined by the values -1 and $+1$ of the decision function:



The margin is $2/\| \vec{w} \|_2$

Indeed, the points $\vec{x}_1$ and $\vec{x}_2$ satisfy:

$$\begin{cases} \vec{w}.\vec{x}_1 + b = 0\,, \\ \vec{w}.\vec{x}_2 + b = 1\,. \end{cases}$$

By subtracting we get $\vec{w}.(\vec{x}_2 - \vec{x}_1) = 1$, and therefore:

$$\gamma = 2\| \vec{x}_2 - \vec{x}_1 \|_2 = \frac{2}{\| \vec{w} \|_2}\,.$$

All training points should be on the correct side of the dotted line

For positive examples ($y_i = 1$) this means:

$$\vec{w}.\vec{x}_i + b \geq 1\,.$$

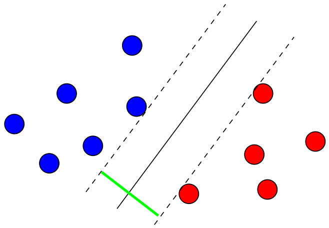
For negative examples ($y_i = -1$) this means:

$$\vec{w}.\vec{x}_i + b \leq -1\,.$$

Both cases are summarized by:

$$\forall i = 1, \dots, n\,, \qquad y_i\left(\vec{w}.\vec{x}_i + b\right) \geq 1\,.$$

Finding the optimal hyperplane



Find $(\vec{w}, b)$ which minimize:

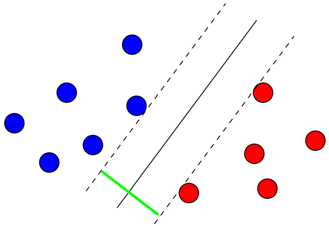
$$\| \vec{w} \|_2^2$$

under the constraints:

$$\forall i = 1, \ldots, n\,, \qquad y_i \left(\vec{w}.\vec{x}_i + b \right) - 1 \geq 0\,.$$

This is a classical quadratic program on $\mathbb{R}^{p+1}$.

Another view of hard-margin SVM



$$\min_{\vec{w},b} \left\{ \sum_{i=1}^{n} \ell_{hard-margin}\left(\vec{w}.x_i + b, y_i\right) + \lambda \| \vec{w} \|_2^2 \right\} ,$$

for the hard-margin loss function:

$$\ell_{hard-margin}(u,y) = \begin{cases} 0 & \text{if } yu \geq 1 \,, \\ +\infty & \text{otherwise.} \end{cases}$$

Outline

Optimization problems

Setting

- We consider an equality and inequality constrained optimization problem over a variable $x \in \mathcal{X}$:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & h_i(x) = 0\ , \quad i = 1, \ldots, m\ , \\ & g_j(x) \leq 0\ , \quad j = 1, \ldots, r\ , \end{aligned}$$

  making no assumption of f, g and h.
- Let us denote by f^* the optimal value of the decision function under the constraints, i.e., $f^* = f(x^*)$ if the minimum is reached at a global minimum x^*.

Lagrangian and dual function

Lagrangian

The Lagrangian of this problem is the function $L : \mathcal{X} \times \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}$ defined by:

$$L\left(x, \lambda, \mu\right) = f\left(x\right) + \sum_{i=1}^{m} \lambda_i h_i\left(x\right) + \sum_{j=1}^{r} \mu_j g_j(x) \,.$$

Lagrangian dual function

The Lagrange dual function $g : \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}$ is:

$$\begin{aligned} q(\lambda, \mu) &= \inf_{x \in \mathcal{X}} L\left(x, \lambda, \mu\right) \\ &= \inf_{x \in \mathcal{X}} \left(f\left(x\right) + \sum_{i=1}^{m} \lambda_i h_i\left(x\right) + \sum_{j=1}^{r} \mu_j g_j(x) \right) \,. \end{aligned}$$

Properties of the dual function

- q is concave in (λ, μ), even if the original problem is not convex.
- The dual function yields lower bounds on the optimal value f^* of the original problem when μ is nonnegative:

$$q(\lambda, \mu) \leq f^* \,, \quad \forall \lambda \in \mathbb{R}^m, \forall \mu \in \mathbb{R}^r, \mu \geq 0 \,.$$

Proofs

- For each x, the function $(\lambda, \mu) \mapsto L(x, \lambda, \mu)$ is linear, and therefore both convex and concave in (λ, μ). The pointwise minimum of concave functions is concave, therefore q is concave.
- Let $\bar{x}$ be any feasible point, i.e., $h(\bar{x}) = 0$ and $g(\bar{x}) \leq 0$. Then we have, for any λ and $\mu \geq 0$:

$$\sum_{i=1}^{m} \lambda_i h_i(\bar{x}) + \sum_{i=1}^{r} \mu_i g_i(\bar{x}) \leq 0 ,$$

$$\implies \quad L(\bar{x}, \lambda, \mu) = f(\bar{x}) + \sum_{i=1}^{m} \lambda_i h_i(\bar{x}) + \sum_{i=1}^{r} \mu_i g_i(\bar{x}) \leq f(\bar{x}) ,$$

$$\implies \quad q(\lambda, \mu) = \inf_x L(x, \lambda, \mu) \leq L(\bar{x}, \lambda, \mu) \leq f(\bar{x}) , \quad \forall \bar{x} . \quad \square$$

Dual problem

Definition

For the (primal) problem:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & h(x) = 0\ , \quad g(x) \leq 0\ , \end{aligned}$$

the Lagrange dual problem is:

$$\begin{aligned} \text{maximize} \quad & q(\lambda, \mu) \\ \text{subject to} \quad & \mu \geq 0\ , \end{aligned}$$

where q is the (concave) Lagrange dual function and λ and μ are the Lagrange multipliers associated to the constraints $h(x) = 0$ and $g(x) \leq 0$.

Weak duality

- Let d^* the optimal value of the Lagrange dual problem. Each $q(\lambda, \mu)$ is an lower bound for f^* and by definition d^* is the best lower bound that is obtained. The following weak duality inequality therefore always hold:

$$d^* \leq f^* \ .$$

- This inequality holds when d^* or f^* are infinite. The difference $d^* - f^*$ is called the optimal duality gap of the original problem.

Strong duality

- We say that strong duality holds if the optimal duality gap is zero, i.e.:

$$d^* = f^* \ .$$

- If strong duality holds, then the best lower bound that can be obtained from the Lagrange dual function is tight
- Strong duality does not hold for general nonlinear problems.
- It usually holds for convex problems.
- Conditions that ensure strong duality for convex problems are called constraint qualification.

Slater's constraint qualification

Strong duality holds for a convex problem:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_j(x) \leq 0 \,, \quad j = 1, \ldots, r \,, \\ & Ax = b \,, \end{aligned}$$

if it is strictly feasible, i.e., there exists at least one feasible point that satisfies:

$$g_j(x) < 0 \,, \quad j = 1, \ldots, r \,, \quad Ax = b \,.$$

Remarks

- Slater's conditions also ensure that the maximum d^* (if $> -\infty$) is attained, i.e., there exists a point (λ^*, μ^*) with

$$q(\lambda^*, \mu^*) = d^* = f^*$$

- They can be sharpened. For example, strict feasibility is not required for affine constraints.
- There exist many other types of constraint qualifications

Dual optimal pairs

Suppose that strong duality holds, x^* is primal optimal, (λ^*, μ^*) is dual optimal. Then we have:

$$\begin{aligned}
f(x^*) &= q\left(\lambda^*, \mu^*\right) \\
&= \inf_{x \in \mathbb{R}^n} \left\{ f(x) + \sum_{i=1}^{m} \lambda_i^* h_i(x) + \sum_{j=1}^{r} \mu_j^* g_j(x) \right\} \\
&\leq f(x^*) + \sum_{i=1}^{m} \lambda_i^* h_i(x^*) + \sum_{j=1}^{r} \mu_j^* g_j(x^*) \\
&\leq f(x^*)
\end{aligned}$$

Hence both inequalities are in fact equalities.

Complimentary slackness

The first equality shows that:

$$L(x^*, \lambda^*, \mu^*) = \inf_{x \in \mathbb{R}^n} L(x, \lambda^*, \mu^*) \ ,$$

showing that x^* minimizes the Lagrangian at (λ^*, μ^*). The second equality shows that:

$$\mu_j g_j(x^*) = 0 \ , \quad j = 1, \ldots, r \ .$$

This property is called complementary slackness:
the *i*th optimal Lagrange multiplier is zero unless the *i*th constraint is active at the optimum.

Outline

Lagrangian

In order to minimize:

$$\frac{1}{2}\parallel \vec{w} \parallel_2^2$$

under the constraints:

$$\forall i = 1, \ldots, n\,, \qquad y_i \left(\vec{w}.\vec{x_i} + b\right) - 1 \geq 0\,,$$

we introduce one dual variable α_i for each constraint, i.e., for each training point. The Lagrangian is:

$$L\left(\vec{w}, b, \vec{\alpha}\right) = \frac{1}{2}||\vec{w}||^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\vec{w}.\vec{x_i} + b\right) - 1\right)\,.$$

Lagrangian

- $L\left(\vec{w}, b, \vec{\alpha}\right)$ is convex quadratic in $\vec{w}$. It is minimize for:

$$\nabla_{\vec{w}} L = \vec{w} - \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i = 0 \quad \Longrightarrow \quad \vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i \,.$$

- $L\left(\vec{w}, b, \vec{\alpha}\right)$ is affine in b. Its minimum is $-\infty$ except if:

$$\nabla_b L = \sum_{i=1}^{n} \alpha_i y_i = 0 \,.$$

Dual function

- We therefore obtain the Lagrange dual function:

$$
\begin{aligned}
q\left(\vec{\alpha}\right) &= \inf_{\vec{w}\in\mathbb{R}^p, b\in\mathbb{R}} L\left(\vec{w}, b, \vec{\alpha}\right)\\
&= \begin{cases} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n y_i y_j \alpha_i \alpha_j \vec{x}_i.\vec{x}_j & \text{if } \sum_{i=1}^n \alpha_i y_i = 0\,,\\ -\infty & \text{otherwise.}\end{cases}
\end{aligned}
$$

- The dual problem is:

$$
\begin{aligned}
\text{maximize} \quad & q\left(\vec{\alpha}\right)\\
\text{subject to} \quad & \vec{\alpha} \geq 0\,.
\end{aligned}
$$

Dual problem

Find $\alpha^* \in \mathbb{R}^n$ which maximizes

$$L(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{x}_i . \vec{x}_j ,$$

under the (simple) constraints $\alpha_i \geq 0$ (for $i = 1, \ldots, n$), and

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

This is a quadratic program on $\mathbb{R}^N$, with "box constraints". $\vec{\alpha}^*$ can be found efficiently using dedicated optimization softwares.

Recovering the optimal hyperplane

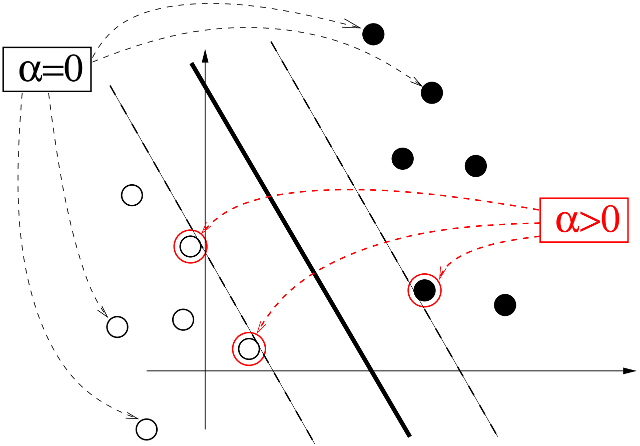
Once $\vec{\alpha}^*$ is found, we recover $(\vec{w}^*, b^*)$ corresponding to the optimal hyperplane. w^* is given by:

$$\vec{w}^* = \sum_{i=1}^{n} \alpha_i \vec{x}_i ,$$

and the decision function is therefore:

$$\begin{aligned} f^*(\vec{x}) &= \vec{w}^* . \vec{x} + b^* \\ &= \sum_{i=1}^{n} \alpha_i \vec{x}_i . \vec{x} + b^* . \end{aligned} \qquad (2)$$

Interpretation: support vectors



Primal (for large n) vs dual (for large p) optimization

1. Find $(\vec{w}, b) \in \mathbb{R}^{p+1}$ which minimize:

$$\| \vec{w} \|_2^2$$

under the constraints:

$$\forall i = 1, \ldots, n, \qquad y_i \left(\vec{w}.\vec{x}_i + b \right) - 1 \geq 0 \,.$$

2. Find $\alpha^* \in \mathbb{R}^n$ which maximizes

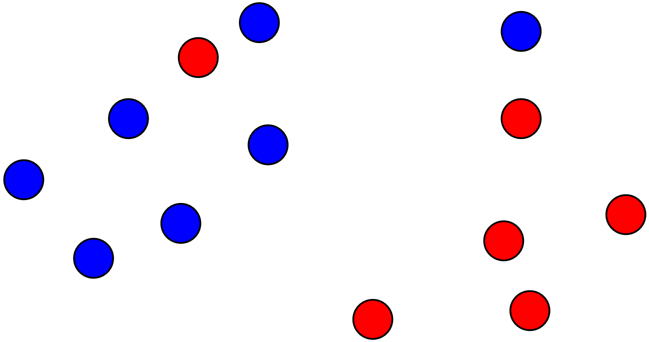
$$L(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{x}_i . \vec{x}_j ,$$

under the (simple) constraints $\alpha_i \geq 0$ (for $i = 1, \ldots, n$), and

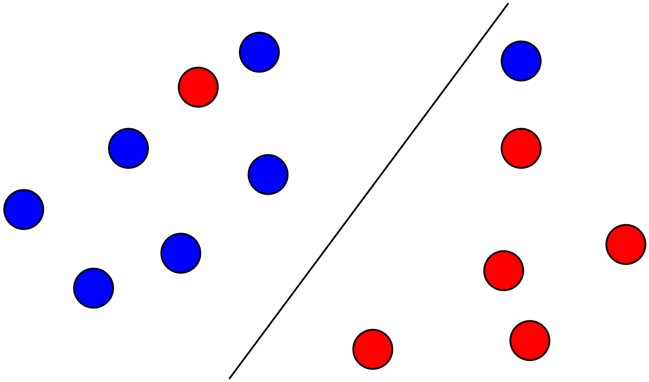
$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

Outline

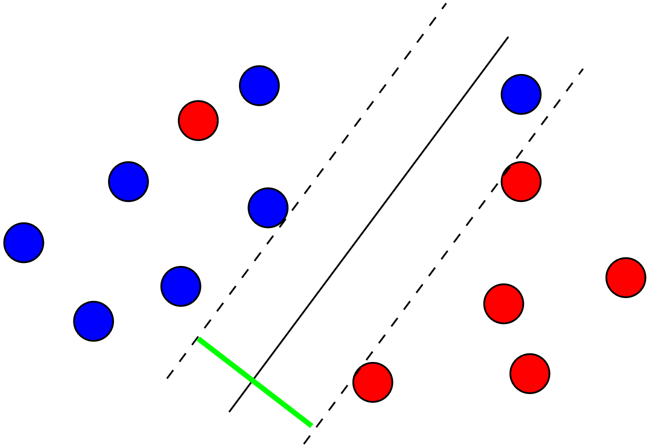
What if data are not linearly separable?



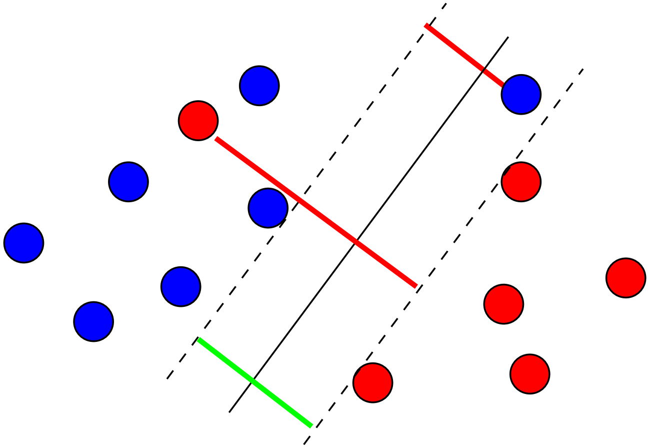
What if data are not linearly separable?



What if data are not linearly separable?



What if data are not linearly separable?

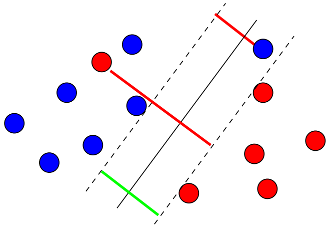


Soft-margin SVM

- Find a trade-off between large margin and few errors.
- Mathematically:

$$\min_f \left\{ \frac{1}{margin(f)} + C \times errors(f) \right\}$$

- C is a parameter



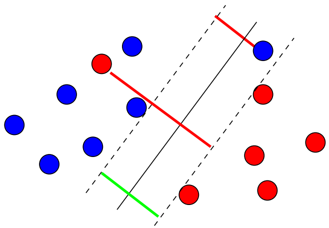
Soft-margin SVM formulation

- The margin of a labeled point $(\vec{x}, y)$ is

$$margin(\vec{x}, y) = y\left(\vec{w}.\vec{x} + b\right)$$

- The error is
  - 0 if $margin(\vec{x}, y) > 1$,
  - $1 - margin(\vec{x}, y)$ otherwise.
- The soft margin SVM solves:

$$\min_{\vec{w},b}\left\{||\vec{w}||^2 + C\sum_{i=1}^{n}\max\left(0, 1 - y_i\left(\vec{w}.\vec{x}_i + b\right)\right)\right\}$$

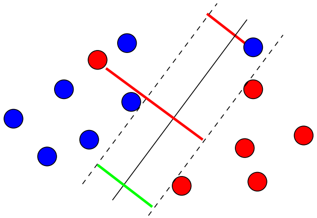


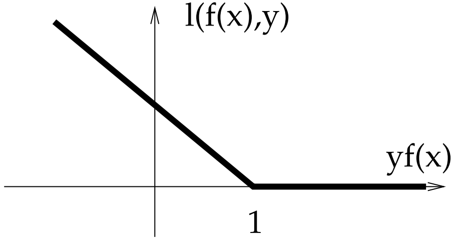
Soft-margin SVM and hinge loss

$$\min_{\vec{w},b} \left\{ \sum_{i=1}^{n} \ell_{\text{hinge}}\left(\vec{w}.x_i + b, y_i\right) + \lambda \| \vec{w} \|_2^2 \right\} ,$$

for $\lambda = 1/C$ and the hinge loss function:

$$\ell_{\text{hinge}}(u, y) = \max\left(1 - yu, 0\right) = \begin{cases} 0 & \text{if } yu \geq 1, \\ 1 - yu & \text{otherwise.} \end{cases}$$





Dual formulation of soft-margin SVM (*exercice*)

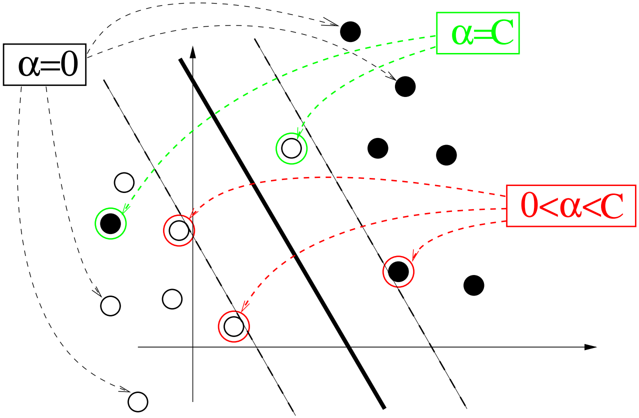
Maximize

$$L(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{x}_i . \vec{x}_j,$$

under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C, & \text{for } i = 1, \dots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases}$$

Interpretation: bounded and unbounded support vectors



Outline

Loss functions for classifications

We already saw 3 loss functions for binary classification problems

- The 0/1 loss $\ell_{0/1}(f(x), y) = \mathbf{1}(yf(x) < 0)$
- The logistic loss $\ell_{logistic}(f(x), y) = \ln\left(1 + e^{-yf(x)}\right)$
- The hinge loss $\ell_{hinge}(f(x), y) = \max(0, 1 - yf(x))$

Definition

In binary classification ($\mathcal{Y} = \{-1, 1\}$), the margin of the function f for a pair $(\mathbf{x}, y)$ is:

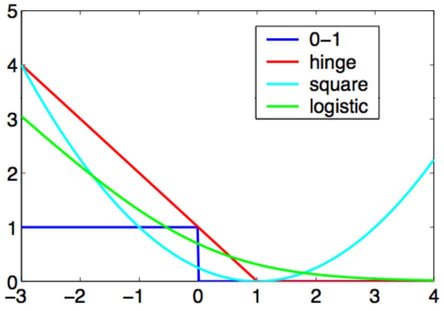
$$yf(\mathbf{x}) .$$

In all cases the loss is a decreasing function of the margin, i.e.,

$$\ell(f(\mathbf{x}), y) = \varphi(yf(\mathbf{x})) , \quad \text{with } \varphi \text{ non-increasing}$$

What about other similar loss functions?

Loss function examples



Method	$\varphi(u)$
Logistic regression	$\log(1+e^{-u})$
Support vector machine (1-SVM)	$\max(1-u,0)$
Support vector machine (2-SVM)	$\max(1-u,0)^2$
Boosting	e^{-u}

Large-margin classifiers

Definition

Given a non-increasing function $\varphi : \mathbb{R} \to \mathbb{R}_+$, a large-margin linear classifier is an algorithm that estimates a function $f_\beta(x) = \beta^\top x$ by solving

$$\min_\beta \frac{1}{n} \sum_{i=1}^{n} \varphi\left(y_i f_\beta(\mathbf{x}_i)\right) + \lambda \| \beta \|_2^2$$

Hence, ridge logistic regression and SVM are large-margin classifier, corresponding to $\varphi(u) = \ln\left(1 + e^{-u}\right)$ and $\varphi(u) = \max(0, 1 - u)$, respectively. Many more are possible.

Questions:

1. Can we solve the optimization problem for other φ's?
2. Is it a good idea to optimize this objective function, if at the end of the day we are interested in the $\ell_{0/1}$ loss, i.e., learning models that make few errors?

Solving large-margin classifiers

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \varphi\left(y_i \beta^\top \mathbf{x}_i\right) + \lambda \| \beta \|_2^2$$

- When φ is convex, this is a stricly convex function of β
- It can then be solved numerically by generic or specific algorithms for convex optimization, e.g., Newton's or gradient method
- When n is large, stochastic optimization is particularly useful (at each step, only approximate the gradient with one or a batch of examples)

A tiny bit of learning theory

Assumptions and notations

- Let $\mathbb{P}$ be an (unknown) distribution on $\mathcal{X} \times \mathcal{Y}$, and $\eta(\mathbf{x}) = \mathbb{P}(Y = 1 \,|\, X = \mathbf{x})$ a measurable version of the conditional distribution of Y given X
- Assume the training set $\mathcal{S}_n = (X_i, Y_i)_{i=1,\dots,n}$ are i.i.d. random variables according to $\mathbb{P}$.
- The risk of a classifier $f : \mathcal{X} \to \mathbb{R}$ is $R(f) = \mathbb{P}\left(sign(f(X)) \neq Y\right)$
- The Bayes risk is
$$R^* = \inf_{f \text{ measurable}} R(f)$$
which is attained for $f^*(\mathbf{x}) = \eta(\mathbf{x}) - 1/2$
- The empirical risk of a classifier $f : \mathcal{X} \to \mathbb{R}$ is
$$R^n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left(sign(f(X_i)) \neq Y_i\right)$$

φ-risk

- Let the empirical φ-risk be the empirical risk optimized by a large-margin classifier:

$$R_\varphi^n(f) = \frac{1}{n}\sum_{i=1}^{n} \varphi\left(Y_i f(X_i)\right)$$

- It is the empirical version of the φ-risk

$$R_\varphi(f) = \mathbb{E}[\varphi\left(Yf\left(X\right)\right)]$$

- Can we hope to have a small risk $R(f)$ if we focus instead on the φ-risk $R_\varphi(f)$?

A small φ-risk ensures a small $0/1$ risk

Theorem [Bartlett et al., 2003]

Let $\varphi : \mathbb{R} \to \mathbb{R}_+$ be convex, non-increasing, differentiable at 0 with $\varphi'(0) < 0$. Let $f : \mathcal{X} \to \mathbb{R}$ measurable such that

$$R_\varphi(f) = \min_{g \text{ measurable}} R_\varphi(g) = R_\varphi^* \,.$$

Then

$$R(f) = \min_{g \text{ measurable}} R(g) = R^* \,.$$

Remarks:

- This tells us that, if we know $\mathbb{P}$, then minimizing the φ-risk is a good idea even if our focus is on the classification error.
- The assumptions on φ can be relaxed; it works for the broader class of *classification-calibrated* loss functions [Bartlett et al., 2003].
- More generally, we can show that if $R_\varphi(f) - R_\varphi^*$ is small, then $R(f) - R^*$ is small too [Bartlett et al., 2003].

A small φ-risk ensures a small $0/1$ risk

Proof sketch:
Condition on $X = \mathbf{x}$:

$$R_\varphi(f \mid X = \mathbf{x}) = \mathbb{E}\left[\varphi\left(Yf\left(X\right)\right) \mid X = \mathbf{x}\right] = \eta(\mathbf{x})\varphi\left(f(\mathbf{x})\right) + \left(1 - \eta(\mathbf{x})\right)\varphi\left(-f(\mathbf{x})\right)$$
$$R_\varphi(-f \mid X = \mathbf{x}) = \mathbb{E}\left[\varphi\left(-Yf\left(X\right)\right) \mid X = \mathbf{x}\right] = \eta(\mathbf{x})\varphi\left(-f(\mathbf{x})\right) + \left(1 - \eta(\mathbf{x})\right)\varphi\left(f(\mathbf{x})\right)$$

Therefore:

$$R_\varphi(f \mid X = \mathbf{x}) - R_\varphi(-f \mid X = \mathbf{x}) = \left[2\eta(\mathbf{x}) - 1\right] \times \left[\varphi\left(f(\mathbf{x})\right) - \varphi\left(-f(\mathbf{x})\right)\right]$$

This must be a.s. ≤ 0 because $R_\varphi(f) \leq R_\varphi(-f)$, which implies:

- if $\eta(\mathbf{x}) > \frac{1}{2}$, $\varphi\left(f(\mathbf{x})\right) \leq \varphi\left(-f(\mathbf{x})\right) \implies f(x) \geq 0$
- if $\eta(\mathbf{x}) < \frac{1}{2}$, $\varphi\left(f(\mathbf{x})\right) \geq \varphi\left(-f(\mathbf{x})\right) \implies f(x) \leq 0$

These inequalities are in fact strict thanks to the assumptions we made on φ (*left as exercice*). $\square$

Empirical risk minimization (ERM)

To find a function with a small φ-risk, the following is a good candidate:

Definition

The ERM estimator on a functional class $\mathcal{F}$ is the solution (when it exists) of:

$$\hat{f}_n = \operatorname*{argmin}_{f \in \mathcal{F}} R_\varphi^n(f)\,.$$

Empirical risk minimization (ERM)

To find a function with a small φ-risk, the following is a good candidate:

Definition

The ERM estimator on a functional class $\mathcal{F}$ is the solution (when it exists) of:

$$\hat{f}_n = \operatorname*{argmin}_{f \in \mathcal{F}} R_\varphi^n(f) \, .$$

Questions:

1. Is $R_\varphi^n(f)$ a good estimate of the true risk $R_\varphi(f)$?
2. Is $R_\varphi(\hat{f}_n)$ small?

Class capacity

Motivations

- The ERM principle gives a good solution if $R_\varphi\left(\hat{f}_n\right)$ is similar to the minimum achievable risk $\inf_{f\in\mathcal{F}} R_\varphi(f)$.
- This can be ensured if $\mathcal{F}$ is not "too large".
- We need a measure of the "capacity" of $\mathcal{F}$.

Definition: Rademacher complexity

The Rademacher complexity of a class of functions $\mathcal{F}$ is:

$$\mathrm{Rad}_n\left(\mathcal{F}\right) = \mathbb{E}_{X,\sigma}\left[\sup_{f\in\mathcal{F}}\left|\frac{2}{n}\sum_{i=1}^{n}\sigma_i f\left(X_i\right)\right|\right],$$

where the expectation is over $(X_i)_{i=1,\dots,n}$ and the independent uniform $\{\pm 1\}$-valued (Rademacher) random variables $(\sigma_i)_{i=1,\dots,n}$.

Basic learning bounds

Theorem

Suppose φ is Lipschitz with constant L_φ:

$$\forall u, u' \in \mathbb{R}, \quad \left| \varphi(u) - \varphi(u') \right| \leq L_\varphi \left| u - u' \right| .$$

Then the φ-risk of the ERM estimator satisfies (on average over the sampling of training set)

$$\underbrace{\mathbb{E}_{\mathcal{S}_n} R_\varphi \left(\hat{f}_n \right) - R_\varphi^*}_{\text{Excess } \varphi\text{-risk}} \leq \underbrace{4 L_\varphi \mathrm{Rad}_n \left(\mathcal{F} \right)}_{\text{Estimation error}} + \underbrace{\inf_{f \in \mathcal{F}} R_\varphi(f) - R_\varphi^*}_{\text{Approximation error}}$$

This quantifies a trade-off between:

- $\mathcal{F}$ "large" $=$ overfitting (approximation error small, estimation error large)
- $\mathcal{F}$ "small" $=$ underfitting (estimation error small, approximation error large)

ERM for bounded linear classifiers

Consider the set of linear functions $f_\beta(x) = \beta^\top x$ where β is bounded:

$$\mathcal{F}_B = \{ f_\beta \,:\, \| \beta \|_2 \leq B \} \,.$$

Theorem

$$\mathrm{Rad}_n \left(\mathcal{F}_B \right) \leq \frac{2B\sqrt{\mathbb{E}\| X \|_2^2}}{\sqrt{n}} \,.$$

Proof (1/2)

$$\begin{aligned}
\mathrm{Rad}_n(\mathcal{F}_B) &= \mathbb{E}_{X,\sigma}\left[\sup_{f\in\mathcal{F}_B}\left|\frac{2}{n}\sum_{i=1}^n \sigma_i f(X_i)\right|\right] \\
&= \mathbb{E}_{X,\sigma}\left[\sup_{\|\beta\|\le B}\left|\left\langle \beta, \frac{2}{n}\sum_{i=1}^n \sigma_i X_i\right\rangle\right|\right] \quad \text{(linearity)} \\
&= \mathbb{E}_{X,\sigma}\left[B\left\|\frac{2}{n}\sum_{i=1}^n \sigma_i X_i\right\|_2\right] \quad \text{(Cauchy-Schwarz)} \\
&= \frac{2B}{n}\mathbb{E}_{X,\sigma}\left[\sqrt{\left\|\sum_{i=1}^n \sigma_i X_i\right\|_2^2}\right] \\
&\le \frac{2B}{n}\sqrt{\mathbb{E}_{X,\sigma}\left[\sum_{i,j=1}^n \sigma_i\sigma_j X_i^\top X_j\right]} \quad \text{(Jensen)}
\end{aligned}$$

Proof (2/2)

But $\mathbb{E}_\sigma\left[\sigma_i\sigma_j\right]$ is 1 if $i = j$, 0 otherwise. Therefore:

$$\begin{aligned}
\mathrm{Rad}_n\left(\mathcal{F}_B\right) &\leq \frac{2B}{n}\sqrt{\mathbb{E}_X\left[\sum_{i,j=1}^{n}\mathbb{E}_\sigma\left[\sigma_i\sigma_j\right]X_i^\top X_j\right]} \\
&\leq \frac{2B}{n}\sqrt{\mathbb{E}_X\sum_{i=1}^{n}\| X_i \|_2^2} \\
&= \frac{2B\sqrt{\mathbb{E}_X\| X \|_2^2}}{\sqrt{n}}. \quad \square
\end{aligned}$$

Basic learning bounds in RKHS balls

Corollary

Suppose $\| X \| \leq \kappa$ a.s. Then the ERM estimator in $\mathcal{F}_B$ satisfies

$$\mathbb{E}R_\varphi\left(\hat{f}_n\right) - R_\varphi^* \leq \frac{8L_\varphi \kappa B}{\sqrt{n}} + \left[\inf_{f \in \mathcal{F}_B} R_\varphi(f) - R_\varphi^*\right].$$

Remarks

- B controls the trade-off between approximation and estimation error
- The bound on expression error is independent of $\mathcal{P}$ and decreases with n
- The approximation error is harder to analyze in general
- In practice, B (or λ, next slide) is tuned by cross-validation

ERM as penalized risk minimization

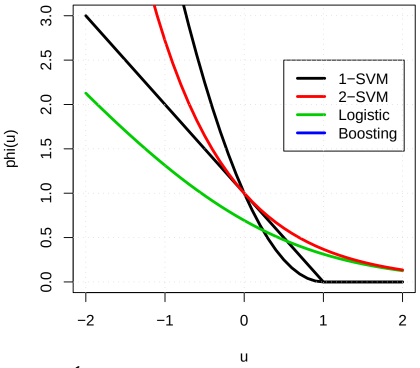
- ERM over $\mathcal{F}_B$ solves the constrained minimization problem:

$$\begin{cases} \min_\beta \frac{1}{n}\sum_{i=1}^n \varphi\left(y_i f_\beta\left(\mathbf{x}_i\right)\right) \\ \text{subject to } \| \beta \|_2 \leq B\,. \end{cases}$$

- To make this practical we assume that φ is convex.
- The problem is then a convex problem in β for which strong duality holds. In particular β solves the problem if and only if it solves for some dual parameter λ the unconstrained problem:

$$\min_{\beta\in\mathbb{R}^p} \left\{ \frac{1}{n}\sum_{i=1}^n \varphi\left(y_i f_\beta\left(\mathbf{x}_i\right)\right) + \lambda \| \beta \|_2^2 \right\}\,.$$

Summary: large margin classifiers

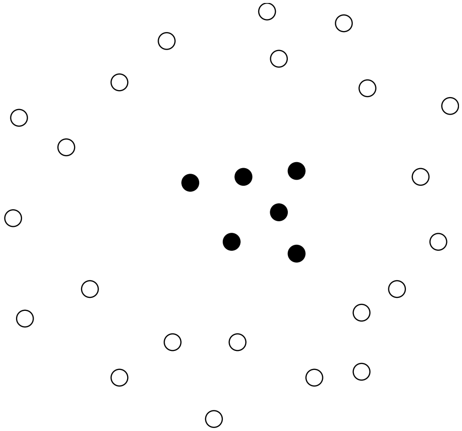


$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \varphi\left(y_i f\left(\mathbf{x}_i\right)\right) + \lambda \| f \|_{\mathcal{H}}^2 \right\}$$

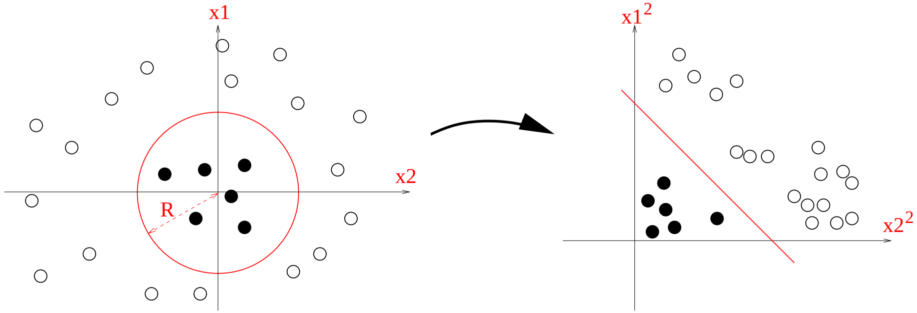
- φ calibrated (e.g., decreasing, $\varphi'(0) < 0$) $\implies$ good proxy for classification error
- φ convex + representer theorem $\implies$ efficient algorithms

Outline

Sometimes linear methods are not interesting



Solution: non-linear mapping to a feature space



Let $\vec{\Phi}(\vec{x}) = (x_1^2, x_2^2)'$, $\vec{w} = (1,1)'$ and $b = 1$. Then the decision function is:

$$f(\vec{x}) = x_1^2 + x_2^2 - R^2 = \vec{w}.\vec{\Phi}(\vec{x}) + b,$$

Kernels

Definition

For a given mapping Φ from the space of objects $\mathcal{X}$ to some feature space, the kernel between two objects x and x' is the inner product of their images in the features space:

$$\forall x, x' \in \mathcal{X}, \quad K(x,x') = \Phi(x)^\top \Phi(x') \,.$$

Example: if $\vec{\Phi}(\vec{x}) = (x_1^2, x_2^2)'$, *then*

$$K(\vec{x},\vec{x}') = \vec{\Phi}(\vec{x}).\vec{\Phi}(\vec{x}') = (x_1)^2(x_1')^2 + (x_2)^2(x_2')^2.$$

The kernel tricks

2 tricks

1. Many linear algorithms (in particular ℓ_2-regularized methods) can be performed in the feature space of $\Phi(x)$ without explicitly computing the images $\Phi(x)$, but instead by computing kernels $K(x, x')$.
2. It is sometimes possible to easily compute kernels which correspond to complex large-dimensional feature spaces: $K(x, x')$ is often much simpler to compute than $\Phi(x)$ and $\Phi(x')$

Trick 1 illustration: SVM in the original space

- Train the SVM by maximizing

$$L(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{x}_i^\top \vec{x}_j ,$$

under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C \,, & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \,. \end{cases}$$

- Predict with the decision function

$$f(\vec{x}) = \sum_{i=1}^{n} \alpha_i \vec{x}_i^\top \vec{x} + b^* \,.$$

Trick 1 illustration: SVM in the feature space

- Train the SVM by maximizing

$$L(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \Phi\left(\vec{x}_i\right)^\top \Phi\left(\vec{x}_j\right)\,,$$

  under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C\,, & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0\,. \end{cases}$$

- Predict with the decision function

$$f\left(\vec{x}\right) = \sum_{i=1}^{n} \alpha_i \Phi\left(\vec{x}_i\right)^\top \Phi\left(\vec{x}\right) + b^*\,.$$

Trick 1 illustration: SVM in the feature space with a kernel

- Train the SVM by maximizing

$$L(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K\left(\vec{x}_i, \vec{x}_j\right) ,$$

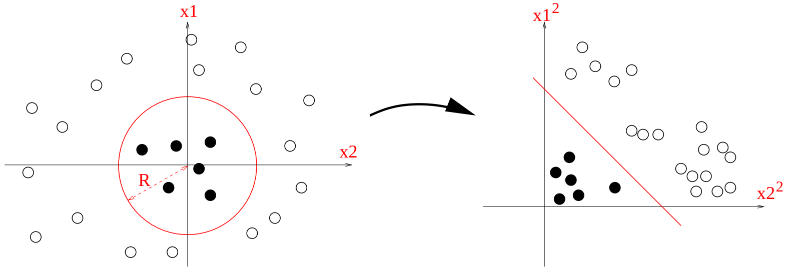
under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C , & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 . \end{cases}$$

- Predict with the decision function

$$f\left(\vec{x}\right) = \sum_{i=1}^{n} \alpha_i K\left(\vec{x}_i, \vec{x}\right) + b^* .$$

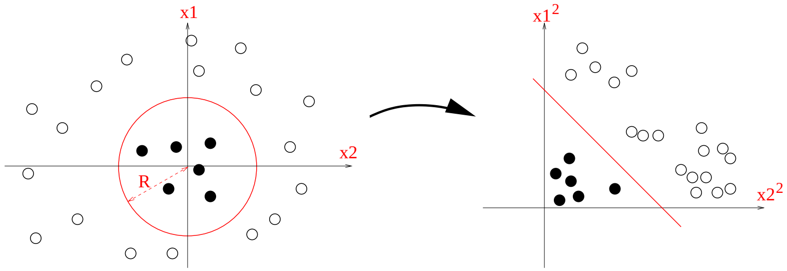
Trick 2 illustration: polynomial kernel



For $\vec{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\vec{\Phi}(\vec{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$:

$$\begin{aligned}
K(\vec{x}, \vec{x}') &= x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 \\
&= \left(x_1x_1' + x_2x_2'\right)^2 \\
&= \left(\vec{x}.\vec{x}'\right)^2 \ .
\end{aligned}$$

Trick 2 illustration: polynomial kernel



More generally,

$$K(\vec{x}, \vec{x}') = \left(\vec{x}.\vec{x}' + 1\right)^d$$

is an inner product in a feature space of all monomials of degree up to d *(left as exercice.)*

Combining tricks: learn a polynomial discrimination rule with SVM

- Train the SVM by maximizing

$$L(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\vec{x}_i^\top \vec{x}_j + 1 \right)^d ,$$

under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C \,, & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \,. \end{cases}$$

- Predict with the decision function

$$f\left(\vec{x}\right) = \sum_{i=1}^{n} \alpha_i \left(\vec{x}_i^\top \vec{x} + 1 \right)^d + b^* \,.$$

Illustration: toy nonlinear problem

```
> plot(x,col=ifelse(y>0,1,2),pch=ifelse(y>0,1,2))
```

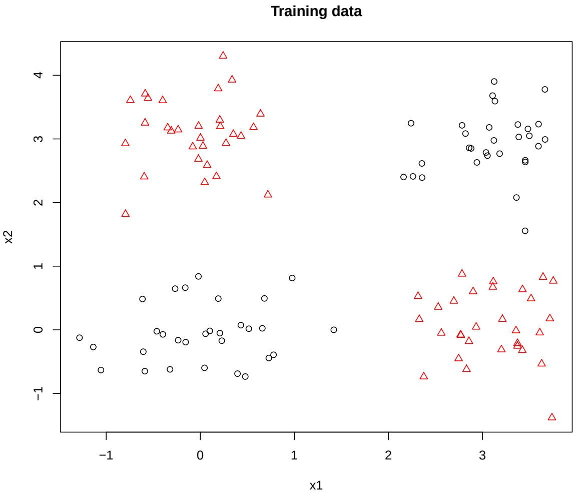


Illustration: toy nonlinear problem, linear SVM

```
> library(kernlab)
> svp <- ksvm(x,y,type="C-svc",kernel='vanilladot')
> plot(svp,data=x)
```

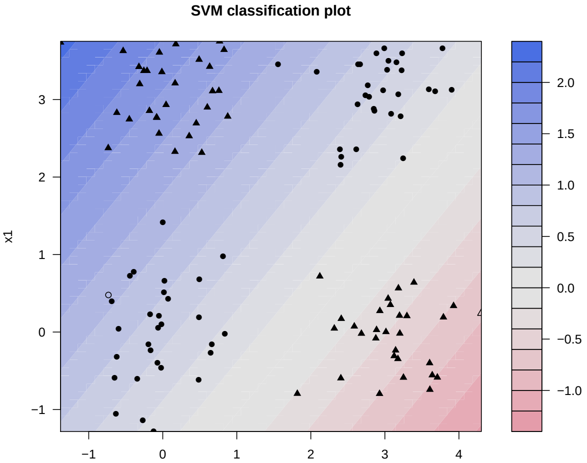
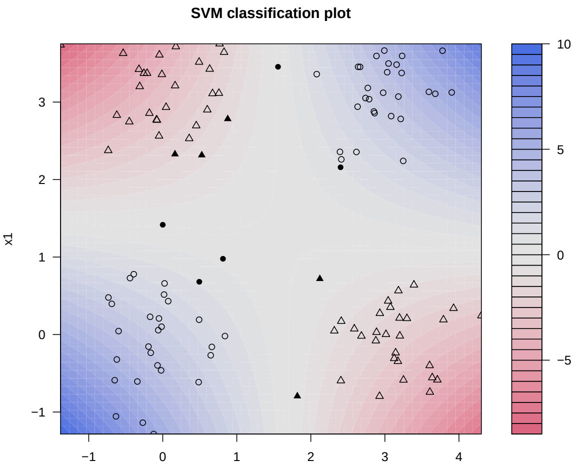


Illustration: toy nonlinear problem, polynomial SVM

```
> svp <- ksvm(x,y,type="C-svc", ...
                        kernel=polydot(degree=2))
> plot(svp,data=x)
```



More generally: trick 1 for ℓ_2-regularized estimators

Representer theorem

Let $f_\beta(x) = \beta^\top \Phi(x)$. Then any solution $\hat{f}_\beta$ of

$$\min_\beta \frac{1}{n} \sum_{i=1}^n \ell(f_\beta(x_i), y_i) + \lambda \|\beta\|_2^2$$

can be expanded as

$$\hat{f}_\beta(x) = \sum_{i=1}^n \alpha_i K(x_i, x)\,,$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is a solution of:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^n \ell\left(\sum_{j=1}^n \alpha_j K(x_i, x_j), y_i\right) + \lambda \sum_{i,j=1}^n \alpha_i \alpha_j K(x_i, x_j)\,.$$

Representer theorem: proof

- For any $\beta \in \mathbb{R}^p$, decompose $\beta = \beta_\mathcal{S} + \beta_\perp$ where $\beta_\mathcal{S} \in span(\Phi(x_1), \ldots, \Phi(x_n))$ and $\beta_\perp$ is orthogonal to it.
- On any point x_i of the training set, we have:

$$f_\beta(x_i) = \beta^\top \Phi(x_i) = \beta_\mathcal{S}^\top \Phi(x_i) + \beta_\perp^\top \Phi(x_i) = \beta_\mathcal{S}^\top \Phi(x_i) = f_{\beta_\mathcal{S}}(x_i)\,.$$

- On the other hand, we have $\| \beta \|_2^2 = \| \beta_\mathcal{S} \|_2^2 + \| \beta_\perp \|_2^2 \geq \| \beta_\mathcal{S} \|_2^2$, with strict inequality if $\beta_\perp \neq 0$.
- Consequently, $\beta_\mathcal{S}$ is always as good as β in terms of objective function, and strictly better if $\beta_\perp \neq 0$. This implies that at any minimum, $\beta_\perp = 0$ and therefore $\beta = \beta_\mathcal{S} = \sum_{i=1}^n \alpha_i \Phi(x_i)$ for some $\boldsymbol{\alpha} \in \mathbb{R}^N$.
- We then just replace β by this expression in the objective function, noting that

$$\|\beta\|_2^2 = \| \sum_{i=1}^n \alpha_i \Phi(x_i) \|_2^2 = \sum_{i,j=1}^n \alpha_i \alpha_j \Phi(x_i)^\top \Phi(x_j) = \sum_{i,j=1}^n \alpha_i \alpha_j K(x_i, x_j)\,.$$

Example: kernel ridge regression

- Let $\Phi : \mathcal{X} \to \mathbb{R}^p$ be a feature mapping from the space of data to a Euclidean or Hilbert space.
- Let $f_\beta(x) = \beta^\top \Phi(x)$ and K the corresponding kernel.
- By the representer theorem, any solution of:

$$\hat{f} = \arg\min_{f_\beta} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\beta(x_i))^2 + \lambda \| \beta \|_2^2$$

can be expanded as:

$$\hat{f} = \sum_{i=1}^{n} \alpha_i K(x_i, x).$$

Example: kernel ridge regression

- Let $Y = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$ the vector of response variables.
- Let $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^\top \in \mathbb{R}^n$ the unknown coefficients.
- Let K be the $n \times n$ Gram matrix: $K_{i,j} = K(x_i, x_j)$.
- We can then write in matrix form:

$$\left(\hat{f}(x_1), \dots, \hat{f}(x_n)\right)^\top = K\boldsymbol{\alpha},$$

- Moreover,

$$\| \beta \|_2^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) = \boldsymbol{\alpha}^\top K \boldsymbol{\alpha}.$$

Example: kernel ridge regression

- The problem is therefore equivalent to:

$$\arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{n}\left(K\boldsymbol{\alpha} - Y\right)^\top \left(K\boldsymbol{\alpha} - Y\right) + \lambda\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\,.$$

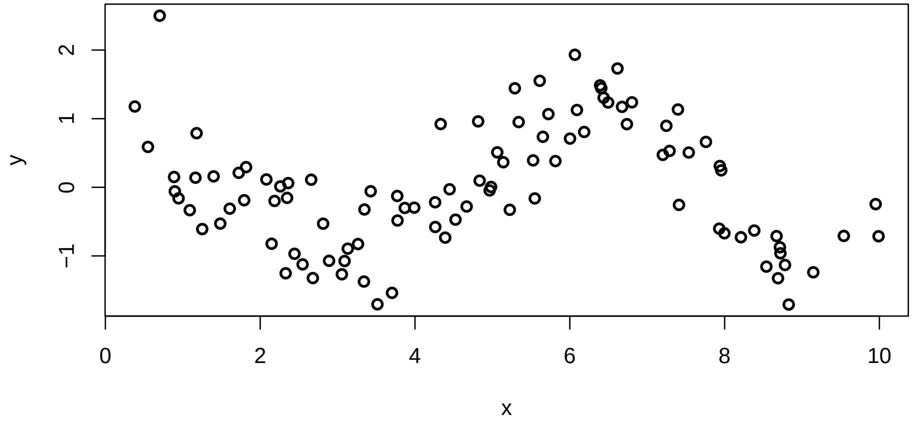
- This is a convex and differentiable function of $\boldsymbol{\alpha}$. Its minimum can therefore be found by setting the gradient in $\boldsymbol{\alpha}$ to zero:

$$\begin{aligned} 0 &= \frac{2}{n}K\left(K\boldsymbol{\alpha} - Y\right) + 2\lambda K\boldsymbol{\alpha} \\ &= K\left[\left(K + \lambda n I\right)\boldsymbol{\alpha} - Y\right] \end{aligned}$$

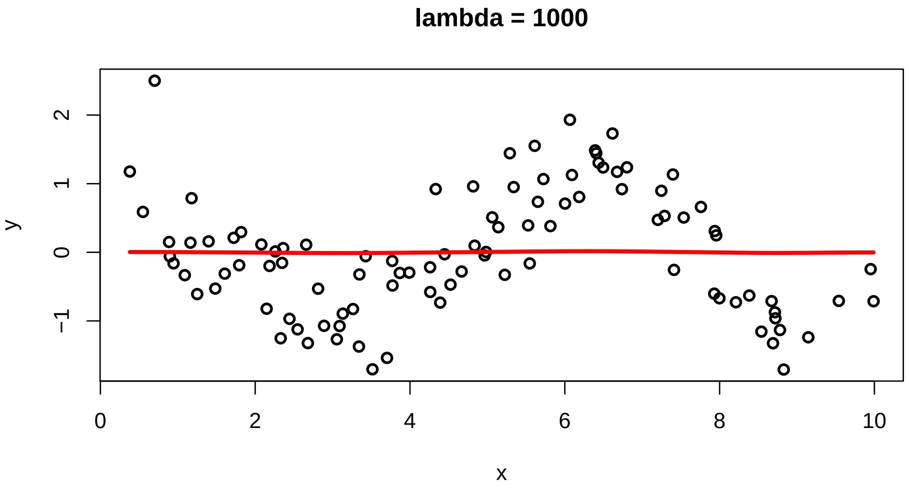
- For $\lambda > 0$, $\mathbf{K} + \lambda n\mathbf{I}$ is invertible (because $\mathbf{K}$ is positive semidefinite) so one solution is to take:

$$\boldsymbol{\alpha} = \left(K + \lambda n\mathbf{I}\right)^{-1} Y.$$

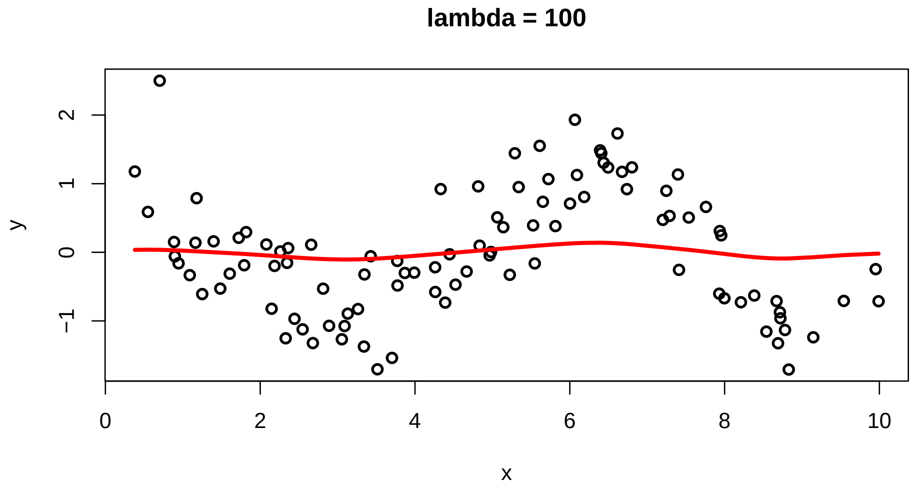
Example (KRR with Gaussian RBF kernel)



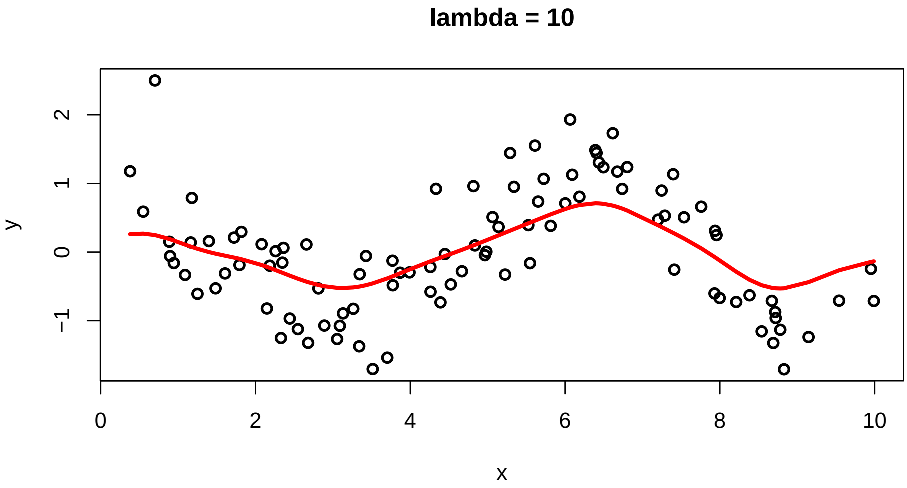
Example (KRR with Gaussian RBF kernel)



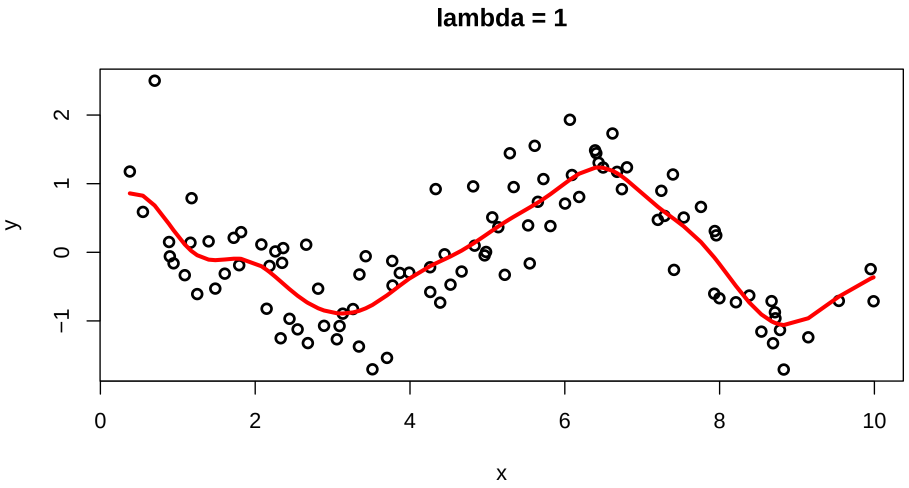
Example (KRR with Gaussian RBF kernel)



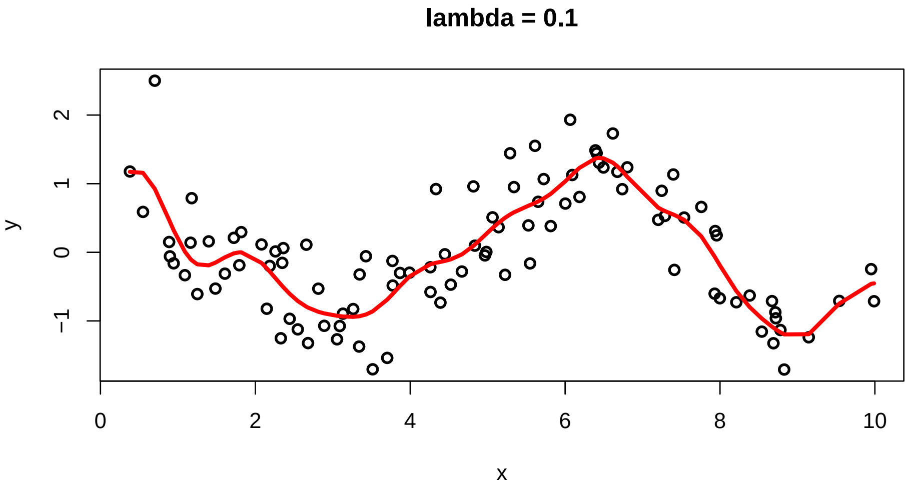
Example (KRR with Gaussian RBF kernel)



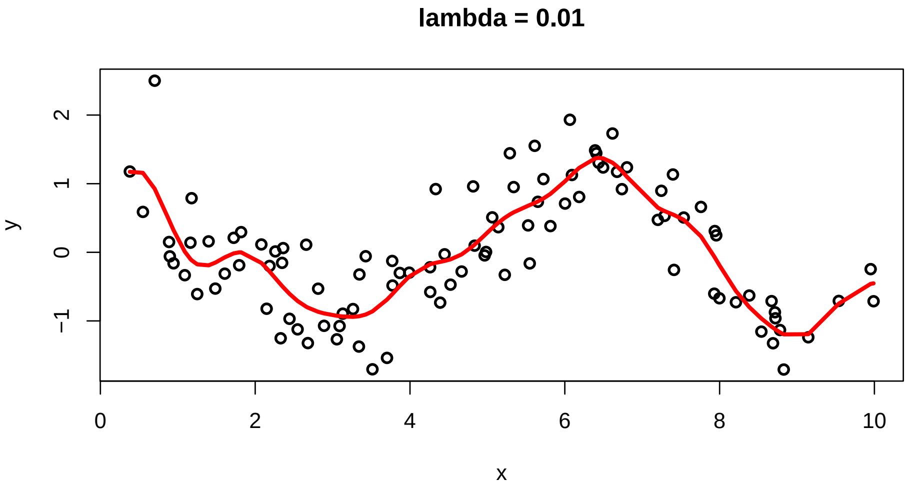
Example (KRR with Gaussian RBF kernel)



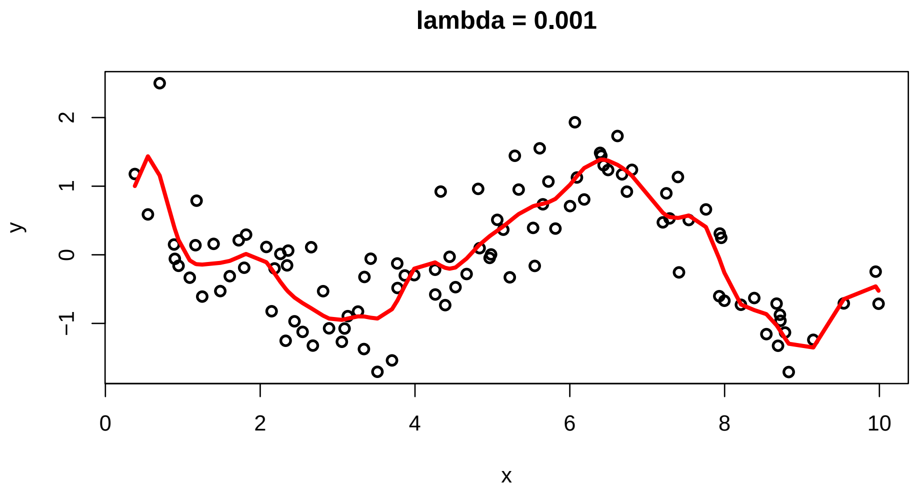
Example (KRR with Gaussian RBF kernel)



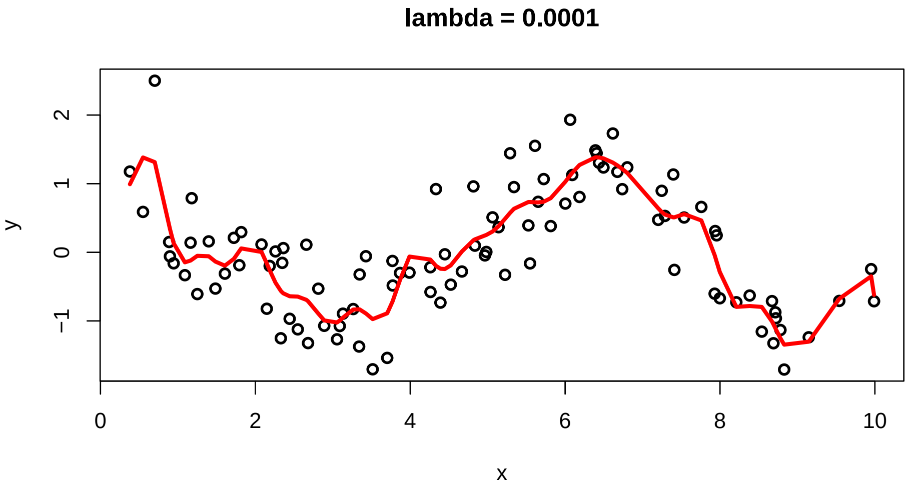
Example (KRR with Gaussian RBF kernel)



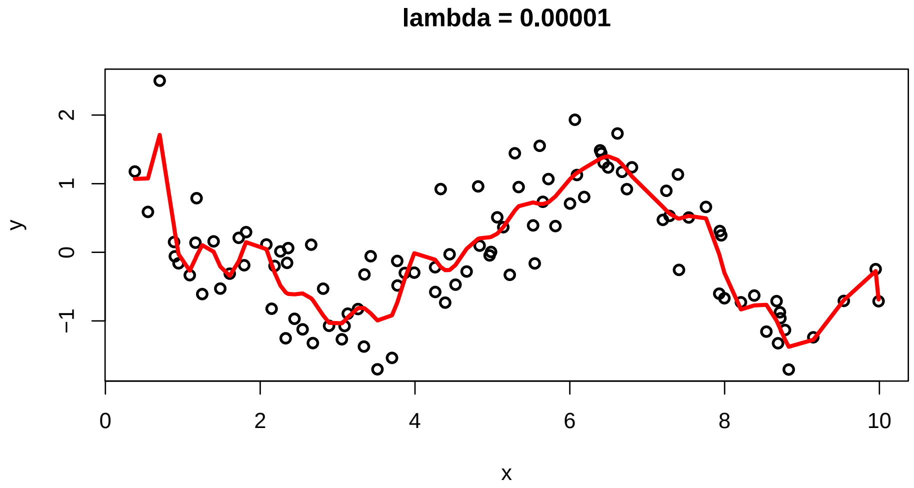
Example (KRR with Gaussian RBF kernel)



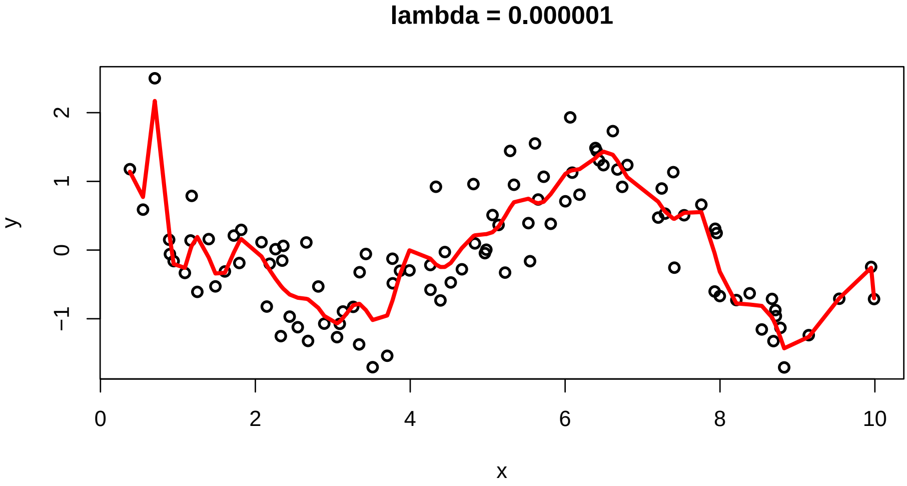
Example (KRR with Gaussian RBF kernel)



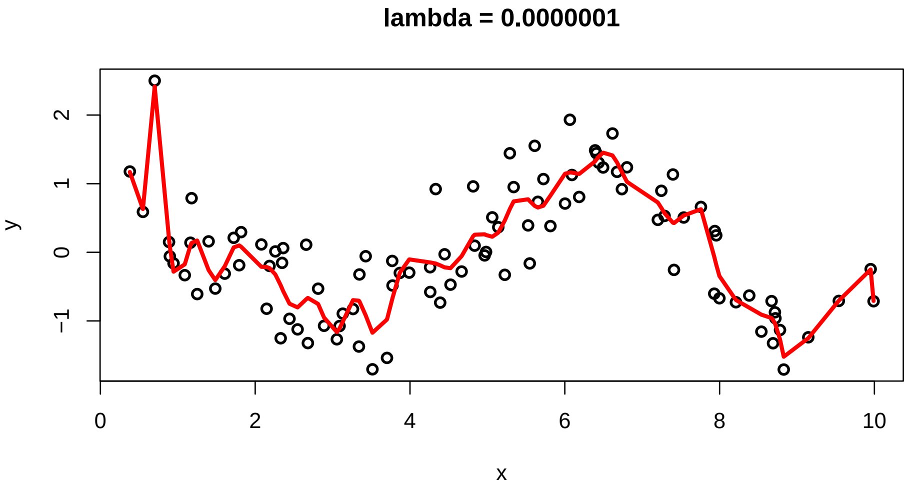
Example (KRR with Gaussian RBF kernel)



Example (KRR with Gaussian RBF kernel)



Example (KRR with Gaussian RBF kernel)



Remark: uniqueness of the solution

Let us find *all* $\boldsymbol{\alpha}$'s that solve

$$K\left[(K+\lambda n I)\boldsymbol{\alpha} - Y\right] = 0$$

- K being a symmetric matrix, it can be diagonalized in an orthonormal basis and $Ker(K) \perp Im(K)$.
- In this basis we see that $(K+\lambda n I)^{-1}$ leaves $Im(K)$ and $Ker(K)$ invariant.
- The problem is therefore equivalent to:

$$\begin{aligned} &(K+\lambda n I)\boldsymbol{\alpha} - Y \in Ker(K) \\ \Leftrightarrow &\boldsymbol{\alpha} - (K+\lambda n I)^{-1} Y \in Ker(K) \\ \Leftrightarrow &\boldsymbol{\alpha} = (K+\lambda n I)^{-1} Y + \epsilon, \text{ with } K\epsilon = 0. \end{aligned}$$

- However, if $\boldsymbol{\alpha}' = \boldsymbol{\alpha} + \epsilon$ with $K\epsilon = 0$, then:

$$\| \beta - \beta' \|_2^2 = \left(\boldsymbol{\alpha} - \boldsymbol{\alpha}'\right)^\top K \left(\boldsymbol{\alpha} - \boldsymbol{\alpha}'\right) = 0\,,$$

  therefore $\beta = \beta'$. KRR has a unique solution β, which can possibly be expressed by several $\boldsymbol{\alpha}$'s if K is singular.

Comparison with "standard" ridge regression

- Let X the $n \times p$ data matrix, $K = XX^\top$ the kernel Gram matrix.
- In "standard" ridge regression, we have $\hat{f}(x) = \hat{\beta}^\top x$ with

$$\hat{\beta} = \left(X^\top X + n\lambda I\right)^{-1} X^\top Y\,.$$

- In "kernel" ridge regression, we have $\tilde{f}(x) = \sum_{i=1}^{n} \alpha_i x_i^\top x = \tilde{\beta}^\top x$ with

$$\tilde{\beta} = \sum_{i=1}^{n} \alpha_i x_i = X^\top \boldsymbol{\alpha} = X^\top \left(XX^\top + \lambda n I\right)^{-1} Y\,.$$

- Oups... which one is correct?

Comparison with "standard" ridge regression

Matrix inversion lemma

For any matrices B and C, and $\gamma > 0$ the following holds (when it makes sense):

$$B\left(CB + \gamma \mathbf{I}\right)^{-1} = \left(BC + \gamma \mathbf{I}\right)^{-1} B$$

We deduce that (of course...):

$$\hat{\beta} = \underbrace{\left(X^\top X + n\lambda I\right)^{-1}}_{p \times p} X^\top Y = X^\top \underbrace{\left(XX^\top + \lambda n I\right)^{-1}}_{n \times n} Y = \tilde{\beta}$$

Computationally, inverting the matrix is the expensive part, which suggest to implement:

- KRR when $p > n$ (high dimension)
- RR when $p < n$ (many points)

Generalization

- We learn the function $f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)$ by solving in $\boldsymbol{\alpha}$ the following optimization problem, with adequate loss function ℓ:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \ell \left(\sum_{j=1}^{n} \alpha_j K(x_i, x_j), y_i \right) + \lambda \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \,.$$

- No explicit solution, but convex optimization problem
- Note that the dimension of the problem is now n instead of p (useful when $n < p$)

The case of SVM

- Soft-margin SVM with a kernel solves:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \left\{ \sum_{i=1}^n \ell_{\text{hinge}} \left(\sum_{j=1}^n \alpha_j K(x_i, x_j), y_i \right) + \lambda \sum_{i,j=1}^n \alpha_i \alpha_j K(x_i, x_j) \right\} .$$

- By Lagrange duality we saw that this is equivalent to

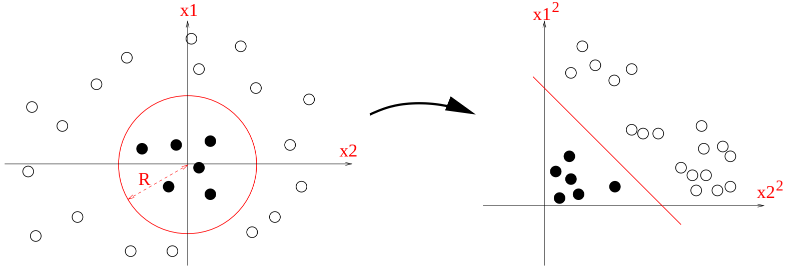
$$\max_{\alpha\in\mathbb{R}^n} L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j) ,$$

under the constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C, & \text{for } i = 1, \dots, n \\ \sum_{i=1}^n \alpha_i y_i = 0 . \end{cases}$$

- This is not a surprise, both problems are also dual to each other (*exercise*).

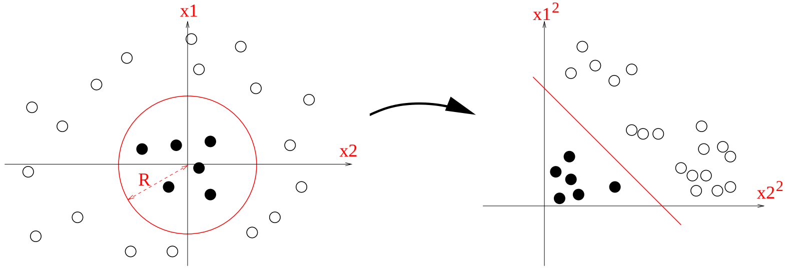
Kernel example: polynomial kernel



For $\vec{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\vec{\Phi}(\vec{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$:

$$\begin{aligned} K(\vec{x}, \vec{x}') &= x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 \\ &= \left(x_1x_1' + x_2x_2'\right)^2 \\ &= \left(\vec{x}.\vec{x}'\right)^2 \quad . \end{aligned}$$

Kernel example: polynomial kernel



More generally,

$$K(\vec{x}, \vec{x}') = \left(\vec{x}.\vec{x}' + 1\right)^d$$

is an inner product in a feature space of all monomials of degree up to d *(left as exercice.)*

Which functions $K(x, x')$ are kernels?

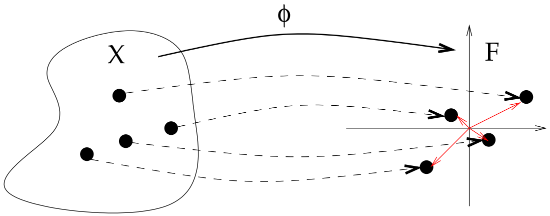
Definition

A function $K(x, x')$ defined on a set $\mathcal{X}$ is a kernel if and only if there exists a features space (Hilbert space) $\mathcal{H}$ and a mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H} \,,$$

such that, for any x, x' in $\mathcal{X}$:

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



Reminder ...

- An inner product on an $\mathbb{R}$-vector space $\mathcal{H}$ is a mapping $(f, g) \mapsto \langle f, g\rangle_{\mathcal{H}}$ from $\mathcal{H}^2$ to $\mathbb{R}$ that is bilinear, symmetric and such that $\langle f, f\rangle > 0$ for all $f \in \mathcal{H}\backslash\{0\}$.
- A vector space endowed with an inner product is called pre-Hilbert. It is endowed with a norm defined by the inner product as $\| f \|_{\mathcal{H}} = \langle f, f\rangle_{\mathcal{H}}^{\frac{1}{2}}$.
- A Hilbert space is a pre-Hilbert space complete for the norm defined by the inner product.

Positive Definite (p.d.) functions

Definition

A positive definite (p.d.) function on the set $\mathcal{X}$ is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ symmetric:

$$\forall \left(\mathbf{x}, \mathbf{x}'\right) \in \mathcal{X}^2, \quad K\left(\mathbf{x}, \mathbf{x}'\right) = K\left(\mathbf{x}', \mathbf{x}\right),$$

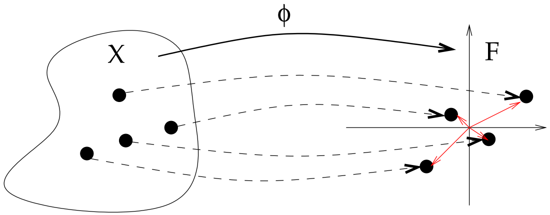
and which satisfies, for all $N \in \mathbb{N}$, $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \in \mathcal{X}^N$ et $(a_1, a_2, \ldots, a_N) \in \mathbb{R}^N$:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \geq 0.$$

Kernels are p.d. functions

Theorem (Aronszajn, 1950)

K is a kernel if and only if it is a positive definite function.



Proof: kernel $\implies$ p.d.

- $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathbb{R}^d} = \langle \Phi(\mathbf{x}'), \Phi(\mathbf{x})_{\mathbb{R}^d} \rangle$,
- $\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathbb{R}^d} = \| \sum_{i=1}^{N} a_i \Phi(\mathbf{x}_i) \|_{\mathbb{R}^d}^2 \geq 0$.

Proof: p.d. $\implies$ kernel (1/5)

- Assume $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is p.d.
- For any $\mathbf{x} \in \mathcal{X}$, let $K_{\mathbf{x}} : \mathcal{X} \mapsto \mathbb{R}$ defined by:

$$K_{\mathbf{x}} : \mathbf{t} \mapsto K(\mathbf{x}, \mathbf{t}) \ .$$

- Let $\mathcal{H}_0$ be the vector subspace of $\mathbb{R}^{\mathcal{X}}$ spanned by the functions $\{K_{\mathbf{x}}\}_{\mathbf{x} \in \mathcal{X}}$, i.e. the functions $f : \mathcal{X} \mapsto \mathbb{R}$ for the form:

$$f = \sum_{i=1}^{m} a_i K_{\mathbf{x}_i}$$

for some $m \in \mathbb{N}$ and $(a_1, \ldots, a_m) \in \mathbb{R}^m$.

Proof: p.d. $\Longrightarrow$ kernel (2/5)

- For any $f, g \in \mathcal{H}_0$, given by:

$$f = \sum_{i=1}^{m} a_i K_{\mathbf{x}_i}, \quad g = \sum_{j=1}^{n} b_j K_{\mathbf{y}_j},$$

  let:

$$\langle f, g \rangle_{\mathcal{H}_0} := \sum_{i,j} a_i b_j K(\mathbf{x}_i, \mathbf{y}_j).$$

- $\langle f, g \rangle_{\mathcal{H}_0}$ does not depend on the expansion of f and g because:

$$\langle f, g \rangle_{\mathcal{H}_0} = \sum_{i=1}^{m} a_i g(\mathbf{x}_i) = \sum_{j=1}^{n} b_j f(\mathbf{y}_j).$$

- This also shows that $\langle ., . \rangle_{\mathcal{H}_0}$ is a symmetric bilinear form.
- This also shows that for any $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}_0$:

$$\langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}_0} = f(\mathbf{x}).$$

Proof: p.d. $\implies$ kernel (3/5)

- K is assumed to be p.d., therefore:

$$\| f \|_{\mathcal{H}_0}^2 = \sum_{i,j=1}^{m} a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \,.$$

  In particular Cauchy-Schwarz is valid with $\langle .,. \rangle_{\mathcal{H}_0}$.

- By Cauchy-Schwarz we deduce that $\forall \mathbf{x} \in \mathcal{X}$:

$$| f(\mathbf{x}) | = \left| \langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}_0} \right| \leq \| f \|_{\mathcal{H}_0} . K(\mathbf{x}, \mathbf{x})^{\frac{1}{2}} \,,$$

  therefore $\| f \|_{\mathcal{H}_0} = 0 \implies f = 0$.

- $\mathcal{H}_0$ is therefore a pre-Hilbert space endowed with the inner product $\langle .,. \rangle_{\mathcal{H}_0}$.

Proof: p.d. $\implies$ kernel (4/5)

- For any Cauchy sequence $(f_n)_{n \geq 0}$ in $\left(\mathcal{H}_0, \langle ., . \rangle_{\mathcal{H}_0}\right)$, we note that:

$$\forall (\mathbf{x}, m, n) \in \mathcal{X} \times \mathbb{N}^2, \quad | f_m(\mathbf{x}) - f_n(\mathbf{x}) | \leq \| f_m - f_n \|_{\mathcal{H}_0} . K(\mathbf{x}, \mathbf{x})^{\frac{1}{2}} .$$

  Therefore for any $\mathbf{x}$ the sequence $(f_n(\mathbf{x}))_{n \geq 0}$ is Cauchy in $\mathbb{R}$ and has therefore a limit.
- If we add to $\mathcal{H}_0$ the functions defined as the pointwise limits of Cauchy sequences, then the space becomes complete and is therefore a Hilbert space (up to a few technicalities, left as exercice). $\square$

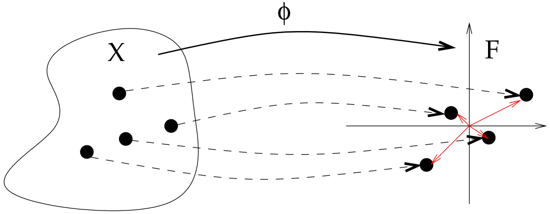
Proof: p.d. $\implies$ kernel (5/5)

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

$$\forall \mathbf{x} \in \mathcal{X}, \quad \Phi(\mathbf{x}) = K_{\mathbf{x}}.$$

- By the reproducing property we have:

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2, \quad \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}} = \langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle_{\mathcal{H}} = K(\mathbf{x}, \mathbf{y}). \qquad \square$$



Kernel examples

- Polynomial (on $\mathbb{R}^d$):

$$K(x,x') = (x.x' + 1)^d$$

- Gaussian radial basis function (RBF) (on $\mathbb{R}^d$)

$$K(x,x') = \exp\left(-\frac{||x-x'||^2}{2\sigma^2}\right)$$

- Laplace kernel (on $\mathbb{R}$)

$$K(x,x') = \exp\left(-\gamma|x-x'|\right)$$

- Min kernel (on $\mathbb{R}_+$)

$$K(x,x') = \min(x,x')$$

Exercice

Exercice: for each kernel, find a Hilbert space $\mathcal{H}$ and a mapping $\Phi : \mathcal{X} \to \mathcal{H}$ such that $K(x,x') = \langle \Phi(x), \Phi(x') \rangle$

Example: SVM with a Gaussian kernel

- Training:

$$\min_{\alpha\in\mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \exp\left(-\frac{||\vec{x}_i - \vec{x}_j||^2}{2\sigma^2}\right)$$

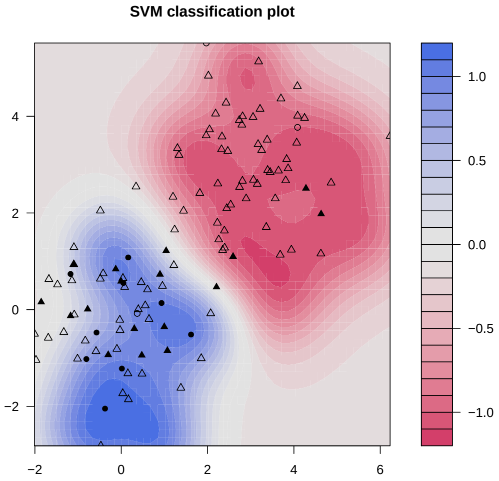
$$\text{s.t. } 0 \leq \alpha_i \leq C, \quad \text{and } \sum_{i=1}^{n} \alpha_i y_i = 0.$$

- Prediction

$$f(\vec{x}) = \sum_{i=1}^{n} \alpha_i \exp\left(-\frac{||\vec{x} - \vec{x}_i||^2}{2\sigma^2}\right)$$

Example: SVM with a Gaussian kernel

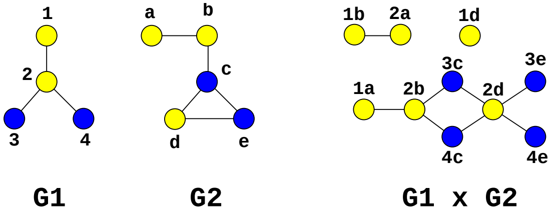
$$f(\vec{x}) = \sum_{i=1}^{n} \alpha_i \exp\left(-\frac{||\vec{x} - \vec{x}_i||^2}{2\sigma^2}\right)$$



How to choose or make a kernel?

- I don't really know...
- Design features?
- Adapt a distance or similarity measure?
- Design a regularizer on f?

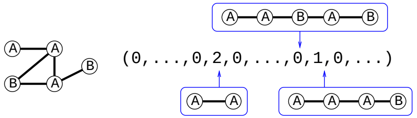
Example: design features (Gärtner et al., 2003)



$$K(G_1, G_2) = \mathbf{1}^\top A^n_{G_1 \times G_2} \mathbf{1}$$

Exercice

Show that the features are the counts of labeled walks of length n in the graph.



Example: adapt a similarity measure (Saigo et al., 2004)

```
CGGSLIAMM------WFGV
|...|||||....||||
C----LIVMMNRLMWFGV
```

$$
\begin{aligned}
s_{S,g}(\pi) = & \; S(C,C) + S(L,L) + S(I,I) + S(A,V) + 2S(M,M) \\
& + S(W,W) + S(F,F) + S(G,G) + S(V,V) - g(3) - g(4)
\end{aligned}
$$

$$SW_{S,g}(\mathbf{x},\mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x},\mathbf{y})} s_{S,g}(\pi) \quad \text{is not a kernel}$$

$$K_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x},\mathbf{y})} \exp\left(\beta s_{S,g}(\mathbf{x},\mathbf{y},\pi)\right) \quad \text{is a kernel}$$

Example: design a regularizer

- Remember $f_\beta(x) = x^\top \Phi(x)$, the regularizer is $\Omega(f_\beta) = \|\beta\|^2$
- Regularize in the Fourier domain:

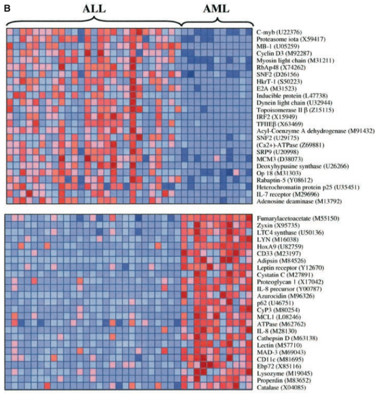
$$\Omega(f) = \int \|\hat{f}(\omega)\|^2 \exp \frac{\sigma^2 \omega^2}{2} d\omega \qquad K(x, y) = \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$$

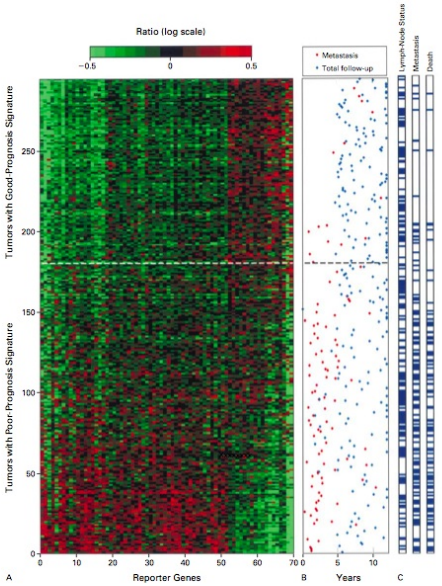
- Sobolev norms

$$\Omega(f) = \int_0^1 f'(u)^2 du \qquad K(x, y) = \min(x, y)$$

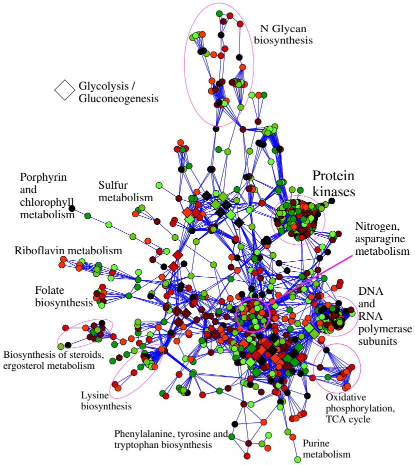
Outline

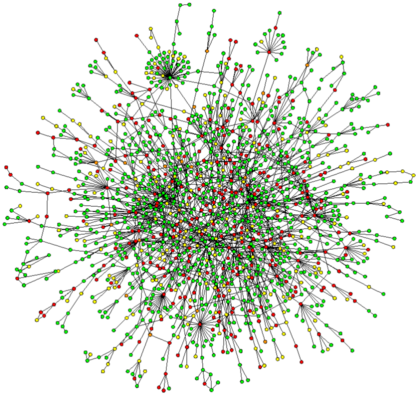
Molecular diagnosis / prognosis / theragnosis





Gene networks

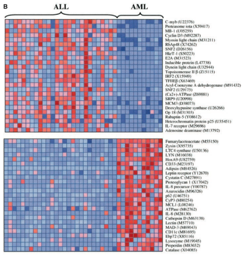

N Glycan biosynthesis
Glycolysis / Gluconeogenesis
Porphyrin and chlorophyll metabolism
Sulfur metabolism
Protein kinases
Nitrogen, asparagine metabolism
Riboflavin metabolism
Folate biosynthesis
DNA and RNA polymerase subunits
Biosynthesis of steroids, ergosterol metabolism
Lysine biosynthesis
Oxidative phosphorylation, TCA cycle
Phenylalanine, tyrosine and tryptophan biosynthesis
Purine metabolism

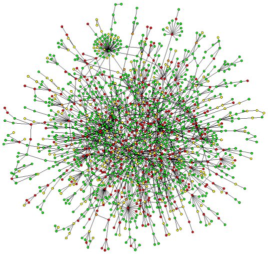


Gene networks and expression data

Motivation

- Basic biological functions usually involve the coordinated action of several proteins:
  - Formation of protein complexes
  - Activation of metabolic, signalling or regulatory pathways
- Many pathways and protein-protein interactions are already known
- Hypothesis: the weights of the classifier should be "coherent" with respect to this prior knowledge





Graph based penalty

$$f_\beta(x) = \beta^\top x \qquad \min_\beta R(f_\beta) + \lambda\Omega(\beta)$$

Prior hypothesis

Genes near each other on the graph should have similar weights.

Graph based penalty

$$f_\beta(x) = \beta^\top x \qquad\qquad \min_\beta R(f_\beta) + \lambda\Omega(\beta)$$

Prior hypothesis

Genes near each other on the graph should have similar weights.

An idea Rapaport et al. [2007]

$$\Omega(\beta) = \sum_{i\sim j} (\beta_i - \beta_j)^2 \,,$$

$$\min_{\beta\in\mathbb{R}^p} R(f_\beta) + \lambda \sum_{i\sim j} (\beta_i - \beta_j)^2 \,.$$

Graph-based penalty as a kernel

Theorem

The function $f(x) = \beta^\top x$ where β is solution of

$$\min_{\beta \in \mathbb{R}^p, \sum_{i=1}^p \beta_i = 0} \frac{1}{n} \sum_{i=1}^n \ell\left(\beta^\top x_i, y_i\right) + \lambda \sum_{i \sim j} (\beta_i - \beta_j)^2$$

is equal to $g(x) = \gamma^\top \Phi(x)$ where γ is solution of

$$\min_{\gamma \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \ell\left(\gamma^\top \Phi(x_i), y_i\right) + \lambda \gamma^\top \gamma\,,$$

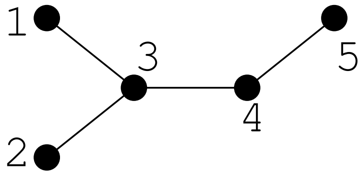
and where

$$\Phi(x)^\top \Phi(x') = x^\top L^* x'$$

for L^* the pseudo-inverse of the graph Laplacian.

So we can just train a kernel method with $K(x, x') = x^\top L^* x'$

Graph Laplacian

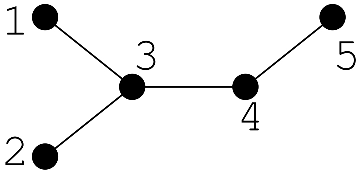


$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Graph Laplacian

Definition

The Laplacian of the graph is the matrix $L = D - A$.



$$L = D - A = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Properties of the Laplacian

Lemma

Let $L = D - A$ be the Laplacian of a connected graph with p vertices:

- *For any $f \in \mathbb{R}^p$,*
$$\sum_{i \sim j} (f_i - f_j)^2 = f^\top L f$$
- *L is a symmetric positive semi-definite matrix*
- *0 is an eigenvalue with multiplicity 1 associated to the constant eigenvector $\mathbf{1} = (1, \ldots, 1)$*
- *The image of L is*
$$Im(L) = \left\{ f \in \mathbb{R}^p : \sum_{i=1}^{p} f_i = 0 \right\}$$

Proof: link between $\Omega(f)$ and L

$$\begin{aligned}
\sum_{i\sim j}(f_i - f_j)^2 &= \sum_{i\sim j}\left(f_i^2 + f_j^2 - 2f_i f_j\right) \\
&= \sum_{i=1}^{m} D_{i,i} f_i^2 - 2\sum_{i\sim j} f_i f_j \\
&= f^\top D f - f^\top A f \\
&= f^\top L f
\end{aligned}$$

Proof: eigenstructure of L

- L is symmetric because A and D are symmetric.
- For any $f \in \mathbb{R}^p$, $f^\top L f = \Omega(f) \geq 0$, therefore the (real-valued) eigenvalues of L are ≥ 0 : L is therefore positive semi-definite.
- f is an eigenvector associated to eigenvalue 0
  iff $f^\top L f = 0$
  iff $\sum_{i \sim j} (f_i - f_j)^2 = 0$,
  iff $f_i = f_j$ when $i \sim j$,
  iff f is constant (because the graph is connected).
- L being symmetric, $Im(L)$ is the orthogonal supplement of $Ker(L)$, that is, the set of functions orthogonal to $\mathbf{1}$. $\square$

Pseudo-inverse of L

Remember the pseudo-inverse L^* of L is the linear application that is equal to:

- 0 on $Ker(L)$
- L^{-1} on $Im(L)$, that is, if we write:

$$L = \sum_{i=1}^{p} \lambda_i u_i u_i^\top$$

  the eigendecomposition of L:

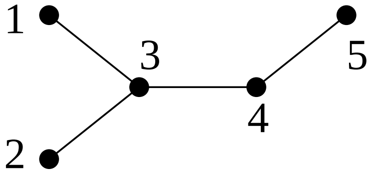
$$L^* = \sum_{\lambda_i \neq 0} (\lambda_i)^{-1} u_i u_i^\top .$$

- In particular it holds that $L^* L = L L^* = \Pi_{\mathcal{H}}$, the projection onto $Im(L) = \mathcal{H}$.
- Similarly, $L^{-\frac{1}{2}} L^{\frac{1}{2}} = L^{\frac{1}{2}} L^{-\frac{1}{2}} = \Pi_{\mathcal{H}}$, where $L^{-\frac{1}{2}} = (L^*)^{\frac{1}{2}}$.

Proof of the theorem

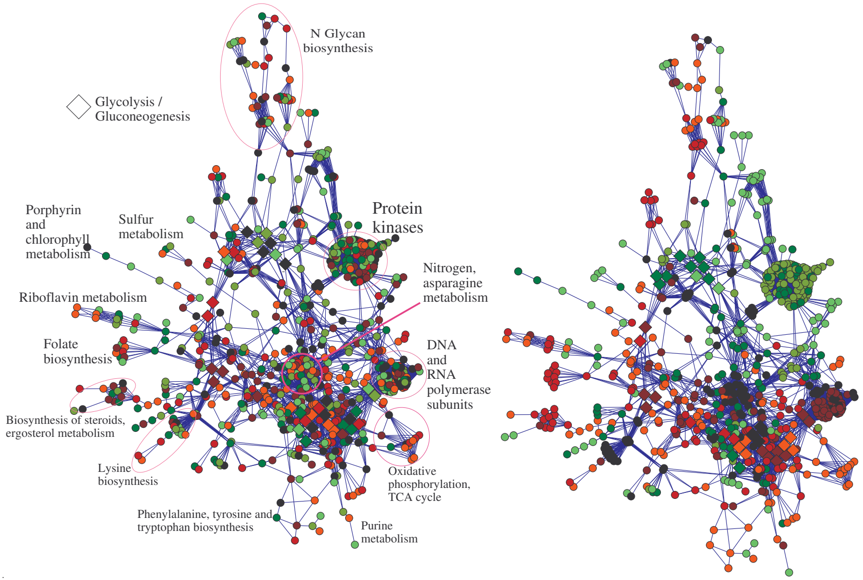
$$
\begin{aligned}
\min_{\beta\in\mathbb{R}^p, \sum_{i=1}^p \beta_i = 0} & \frac{1}{n}\sum_{i=1}^n \ell\left(\beta^\top x_i, y_i\right) + \lambda \sum_{i\sim j}(\beta_i - \beta_j)^2 \\
&= \min_{\beta\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \ell\left(\beta^\top x_i, y_i\right) + \lambda\beta^\top L\beta \\
&= \min_{\beta\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \ell\left((\Pi_{\mathcal{H}}\beta)^\top x_i, y_i\right) + \lambda\beta^\top L\beta \\
&= \min_{\beta\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \ell\left((L^{\frac{1}{2}}\beta)^\top L^{-\frac{1}{2}} x_i, y_i\right) + \lambda(L^{\frac{1}{2}}\beta)^\top(L^{\frac{1}{2}}\beta) \\
&= \min_{\gamma\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \ell\left(\gamma^\top L^{-\frac{1}{2}} x_i, y_i\right) + \lambda\gamma^\top\gamma \\
&= \min_{\gamma\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^n \ell\left(\gamma^\top L^{-\frac{1}{2}} x_i, y_i\right) + \lambda\gamma^\top\gamma
\end{aligned}
$$

Example

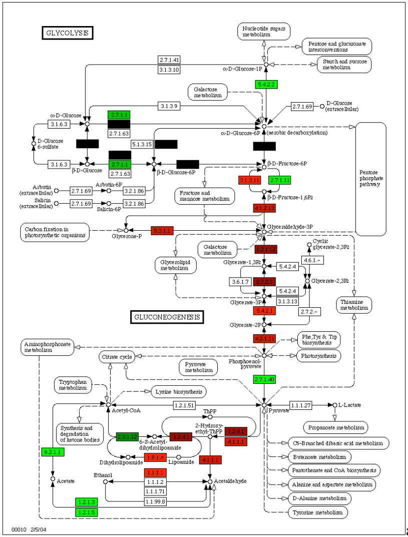


$$L^* = \begin{pmatrix} 0.88 & -0.12 & 0.08 & -0.32 & -0.52 \\ -0.12 & 0.88 & 0.08 & -0.32 & -0.52 \\ 0.08 & 0.08 & 0.28 & -0.12 & -0.32 \\ -0.32 & -0.32 & -0.12 & 0.48 & 0.28 \\ -0.52 & -0.52 & -0.32 & 0.28 & 1.08 \end{pmatrix}$$

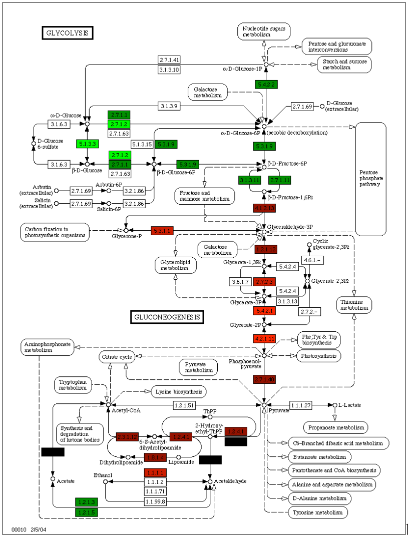
Classifiers


N Glycan biosynthesis
Glycolysis / Gluconeogenesis
Porphyrin and chlorophyll metabolism
Sulfur metabolism
Protein kinases
Nitrogen, asparagine metabolism
Riboflavin metabolism
Folate biosynthesis
DNA and RNA polymerase subunits
Biosynthesis of steroids, ergosterol metabolism
Lysine biosynthesis
Oxidative phosphorylation, TCA cycle
Phenylalanine, tyrosine and tryptophan biosynthesis
Purine metabolism

Classifier



a)



b)

Other penalties with kernels

$$\Phi(x)^\top \Phi(x') = x^\top K_G x'$$

with:

- $K_G = (c + L)^-1$ leads to

$$\Omega(\beta) = c\sum_{i=1}^{p} \beta_i^2 + \sum_{i\sim j} (\beta_i - \beta_j)^2 \ .$$

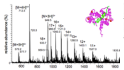
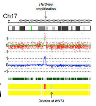
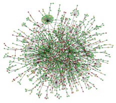
- The diffusion kernel:

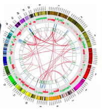
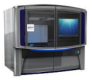
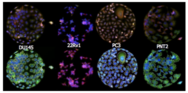
$$K_G = \exp_M(-2tL)\,.$$

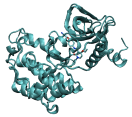
penalizes high frequencies of β in the Fourier domain.

Outline

Motivation



- Assume we observe K types of data and would like to learn a joint model (e.g., predict susceptibility from SNP and expression data).
- We saw in the previous part how to make kernels for each type of data, and learn with kernels
- Kernels are also well suited for data integration!

Setting

- For a kernel $K(x, x') = \Phi(x)^\top \Phi(x')$, we know how to learn a function $f_\beta(x) = \beta^\top \Phi(x)$ by solving:

$$\min_\beta R(f_\beta) + \lambda \|\beta\|^2 \,.$$

- By the representer theorem, we know that the solution is

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) \,,$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the solution of another optimization problem:

$$\min_{\boldsymbol{\alpha}} R(K\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^\top K \boldsymbol{\alpha} = \min_{\boldsymbol{\alpha}} J_K(\boldsymbol{\alpha}) \,.$$

The sum kernel

- Let $K_1, \dots, K_M$ be M kernels corresponding to M sources of data
- Summing the kernel together defines a new "integrated" kernel

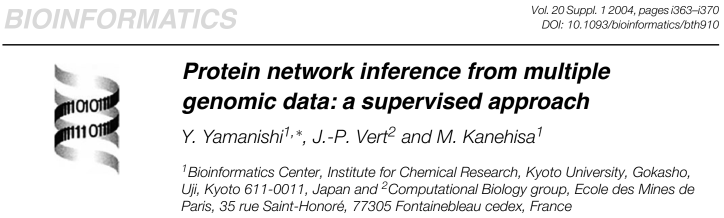
Theorem

Learning with $K = \sum_{i=1}^{M} K_i$ is equivalent to work with a feature vector $\Phi(x)$ obtained by concatenation of $\Phi_1(x), \dots, \Phi_M(x)$. It solves the following problem:

$$\min_{f_{\beta_1}, \dots, f_{\beta_M}} R\left(\sum_{i=1}^{M} f_{\beta_i}\right) + \lambda \sum_{i=1}^{M} \|\beta_i\|^2$$

Proof left as exercise.

Example: protein network inference



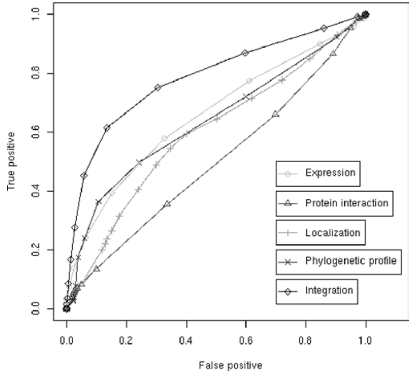
BIOINFORMATICS

Vol. 20 Suppl. 1 2004, pages i363–i370
DOI: 10.1093/bioinformatics/bth910

Protein network inference from multiple genomic data: a supervised approach

Y. Yamanishi[1,*], J.-P. Vert[2] and M. Kanehisa[1]

[1]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611-0011, Japan and [2]Computational Biology group, Ecole des Mines de Paris, 35 rue Saint-Honoré, 77305 Fontainebleau cedex, France



K_{exp} (Expression)
K_{ppi} (Protein interaction)
K_{loc} (Localization)
K_{phy} (Phylogenetic profile)
$K_{\text{exp}} + K_{\text{ppi}} + K_{\text{loc}} + K_{\text{phy}}$ (Integration)

Multiple kernel learning (Lanckriet et al., 2004)

- Perhaps a more clever approach is to learn a weighted linear combination of kernels:

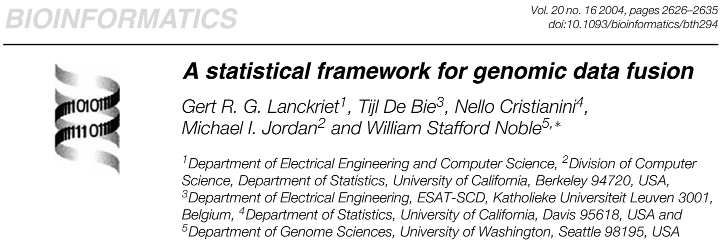
$$K_\eta = \sum_{i=1}^{M} \eta_i K_i \quad \text{with} \quad \eta_i \geq 0\,.$$

- MKL learns the weights with the predictor by solving:

$$\min_{\eta, \boldsymbol{\alpha}} J_{K_\eta}(\boldsymbol{\alpha}) \quad \text{such that} \quad Trace(K_\eta) = 1\,.$$

- The problem is jointly convex in $(\eta, \boldsymbol{\alpha})$ and can be solved efficiently
- The output is both a set of weights η, and a predictor corresponding to the kernel method trained with kernel K_η.

Example: protein annotation



BIOINFORMATICS

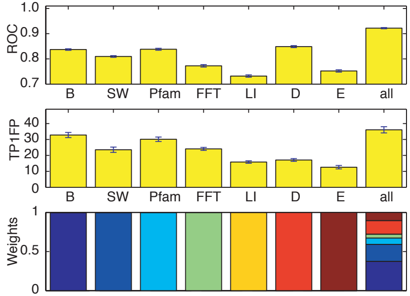
Vol. 20 no. 16 2004, pages 2626–2635
doi:10.1093/bioinformatics/bth294

A statistical framework for genomic data fusion

Gert R. G. Lanckriet[1], Tijl De Bie[3], Nello Cristianini[4], Michael I. Jordan[2] and William Stafford Noble[5,*]

[1]Department of Electrical Engineering and Computer Science, [2]Division of Computer Science, Department of Statistics, University of California, Berkeley 94720, USA, [3]Department of Electrical Engineering, ESAT-SCD, Katholieke Universiteit Leuven 3001, Belgium, [4]Department of Statistics, University of California, Davis 95618, USA and [5]Department of Genome Sciences, University of Washington, Seattle 98195, USA

Kernel	Data	Similarity measure
K_{SW}	protein sequences	Smith-Waterman
K_{B}	protein sequences	BLAST
K_{Pfam}	protein sequences	Pfam HMM
K_{FFT}	hydropathy profile	FFT
K_{LI}	protein interactions	linear kernel
K_{D}	protein interactions	diffusion kernel
K_{E}	gene expression	radial basis kernel
K_{RND}	random numbers	linear kernel



(B) Membrane proteins

MKL revisited

Theorem (Bach et al., 2004)

MKL solves the following problem:

$$\min_{f_{\beta_1},\dots,f_{\beta_M}} R\left(\sum_{i=1}^{M} f_{\beta_i}\right) + \lambda \sum_{i=1}^{M} \|\beta_i\|$$

- This is an instance of (kernelized) group lasso (more later...)
- This promotes sparsity at the kernel level
- MKL is mostly useful if only a few kernels are relevant; otherwise the sum kernel may be a better option.

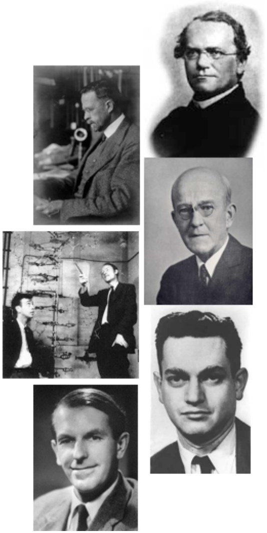
Outline

Part 5

Kernels for Biological Sequences

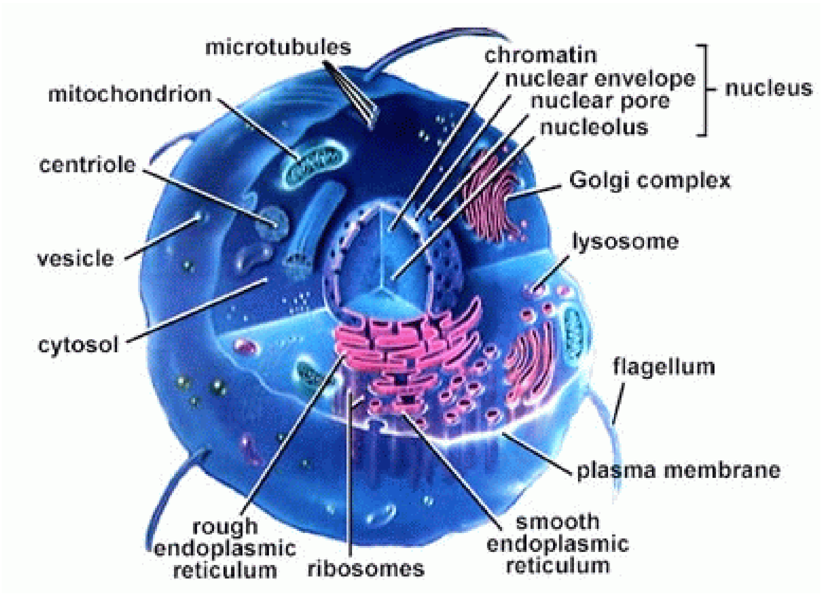
Outline

Short history of genomics

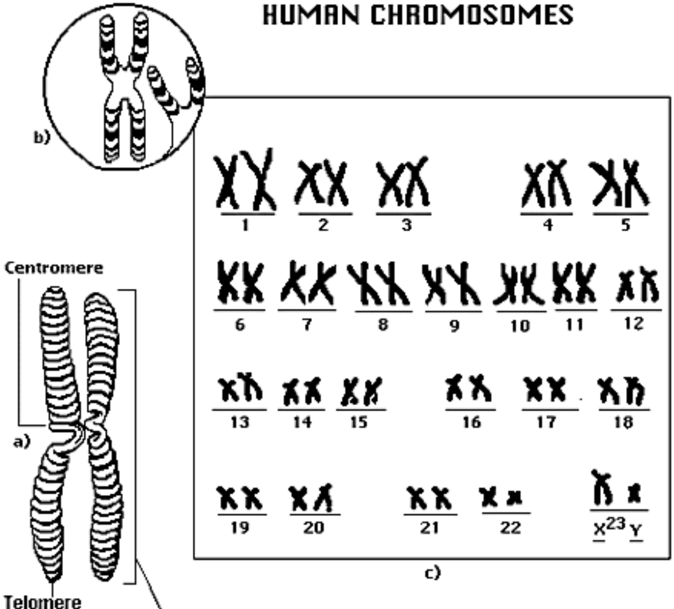


1866 : Laws of heredity (Mendel)
1909 : Morgan and the drosophilists
1944 : DNA supports heredity (Avery)
1953 : Structure of DNA (Crick and Watson)
1966 : Genetic code (Nirenberg)
1960-70 : Genetic engineering
1977 : Method for sequencing (Sanger)
1982 : Creation of Genbank
1990 : Human genome project launched
2003 : Human genome project completed

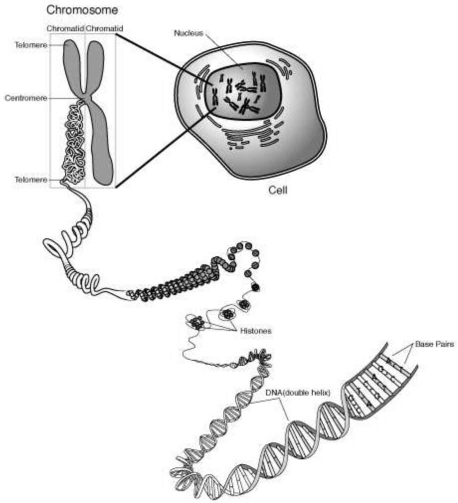
A cell


microtubules
mitochondrion
centriole
vesicle
cytosol
chromatin
nuclear envelope
nuclear pore
nucleolus
nucleus
Golgi complex
lysosome
flagellum
plasma membrane
rough endoplasmic reticulum
ribosomes
smooth endoplasmic reticulum

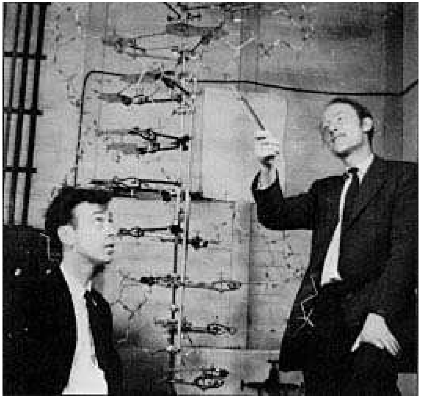
Chromosomes



Chromosomes and DNA

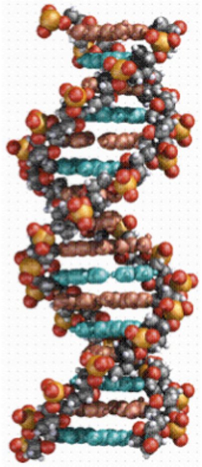


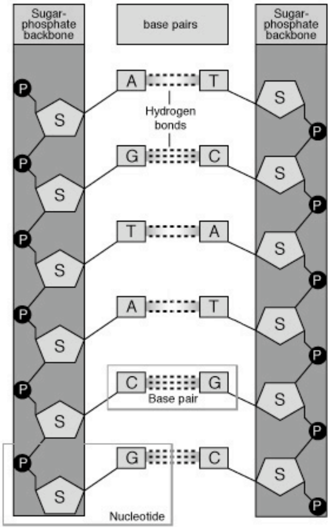
Structure of DNA



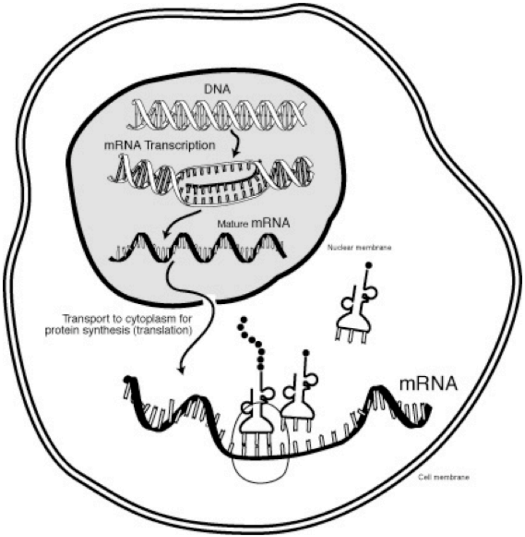
"We wish to suggest a structure for the salt of desoxyribose nucleic acid (D.N.A.). This structure have novel features which are of considerable biological interest" (Watson and Crick, 1953)

The double helix



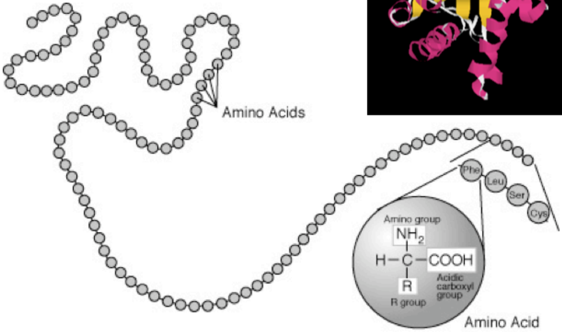


Central dogma

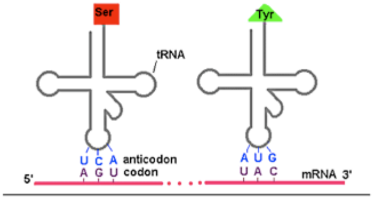


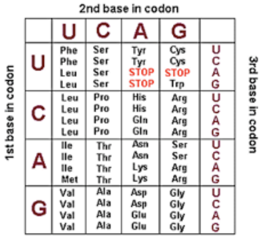
$$\begin{tikzcd}
\text{DNA} \arrow[loop right] \arrow[d] \\
\text{RNA} \arrow[d] \\
\text{Proteins}
\end{tikzcd}$$

Proteins


Amino Acids
Phe
Leu
Ser
Cys
Amino group
NH_2
$H-C-COOH$
R
R group
Acidic carboxyl group
Amino Acid

Genetic code





2nd base in codon

1st base in codon	U	C	A	G	3rd base in codon
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

The Genetic Code

DNA = 4 letters (ATCG)

↓

RNA = 4 letters (AUCG)

↓

Protein = 20 letters (amino acids)

1 amino acid
=
3 nucleotides

Human genome project

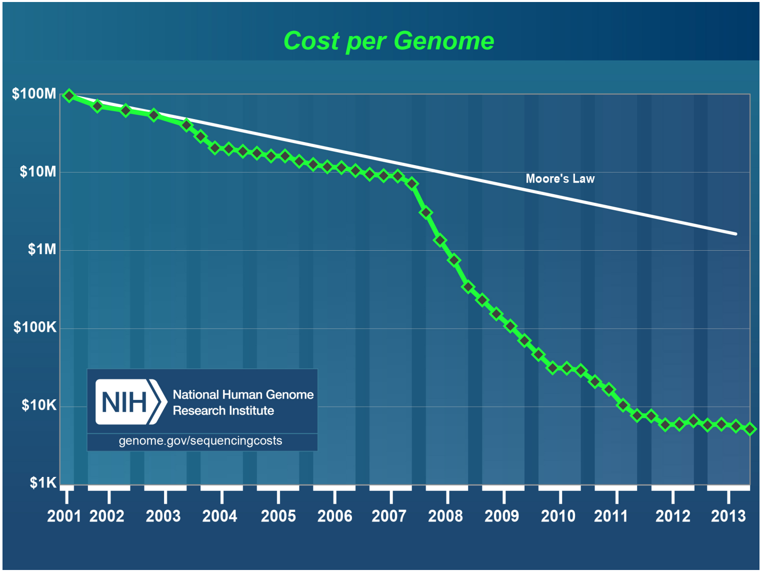
- Goal : sequence the 3,000,000,000 bases of the human genome
- Consortium with 20 labs, 6 countries
- Cost : about 3,000,000,000 USD

2003: we study "the" human genome

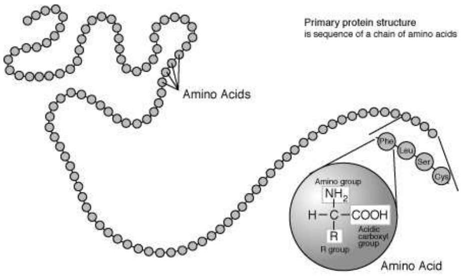
Findings

- About 25,000 genes only (representing 1.2% of the genome)
- Automatic gene finding with graphical models
- 97% of the genome is considered "junk DNA"
- Superposition of a variety of signals (many to be discovered)

2003-2014: towards personalized genomics



Protein sequence



A : Alanine	V : Valine	L : Leucine
F : Phenylalanine	P : Proline	M : Methionine
E : Acide glutamique	K : Lysine	R : Arginine
T : Threonine	C : Cysteine	N : Asparagine
H : Histidine	V : Thyrosine	W : Tryptophane
I : Isoleucine	S : Serine	Q : Glutamine
D : Acide aspartique	G : Glycine	

Challenges with protein sequences

- A protein sequences can be seen as a variable-length sequence over the 20-letter alphabet of amino-acids, e.g., insuline: `FVNQHLCGSHLVEALYLVCGERGFFYTPKA`
- These sequences are produced at a fast rate (result of the sequencing programs)
- Need for algorithms to compare, classify, analyze these sequences
- Applications: classification into functional or structural classes, prediction of cellular localization and interactions, ...

Example: supervised sequence classification

Data (training)

- Secreted proteins:

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

- Non-secreted proteins:

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

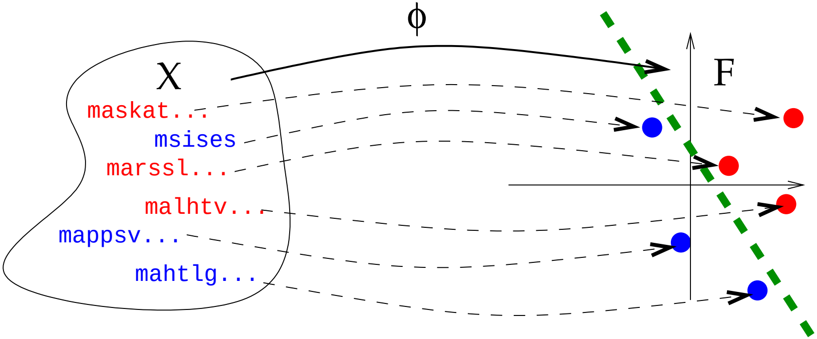
Goal

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

Supervised classification with vector embedding

The idea

- Map each string $x \in \mathcal{X}$ to a vector $\Phi(x) \in \mathcal{F}$.
- Train a classifier for vectors on the images $\Phi(x_1), \dots, \Phi(x_n)$ of the training set (nearest neighbor, linear perceptron, logistic regression, support vector machine...)



Kernels for protein sequences

- Kernel methods have been widely investigated since Jaakkola et al.'s seminal paper (1998).
- What is a good kernel?
    - it should be mathematically valid (symmetric, p.d. or c.p.d.)
    - fast to compute
    - adapted to the problem (give good performances)

Kernel engineering for protein sequences

- Define a (possibly high-dimensional) feature space of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels

Kernel engineering for protein sequences

- Define a (possibly high-dimensional) feature space of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels
- Derive a kernel from a generative model
  - Fisher kernel
  - Mutual information kernel
  - Marginalized kernel

Kernel engineering for protein sequences

- Define a (possibly high-dimensional) feature space of interest
  - Physico-chemical kernels
  - Spectrum, mismatch, substring kernels
  - Pairwise, motif kernels
- Derive a kernel from a generative model
  - Fisher kernel
  - Mutual information kernel
  - Marginalized kernel
- Derive a kernel from a similarity measure
  - Local alignment kernel

Outline

Vector embedding for strings

The idea

Represent each sequence $\mathbf{x}$ by a fixed-length numerical vector $\Phi(\mathbf{x}) \in \mathbb{R}^n$.
How to perform this embedding?

Vector embedding for strings

The idea

Represent each sequence $\mathbf{x}$ by a fixed-length numerical vector $\Phi(\mathbf{x}) \in \mathbb{R}^n$. How to perform this embedding?

Physico-chemical kernel

Extract relevant features, such as:

- length of the sequence
- time series analysis of numerical physico-chemical properties of amino-acids along the sequence (e.g., polarity, hydrophobicity), using for example:
    - Fourier transforms (Wang et al., 2004)
    - Autocorrelation functions (Zhang et al., 2003)

$$r_j = \frac{1}{n-j} \sum_{i=1}^{n-j} h_i h_{i+j}$$

Substring indexation

The approach

Alternatively, index the feature space by fixed-length strings, i.e.,

$$\Phi(\mathbf{x}) = (\Phi_u(\mathbf{x}))_{u \in \mathcal{A}^k}$$

where $\Phi_u(\mathbf{x})$ can be:

- the number of occurrences of u in $\mathbf{x}$ (without gaps) : spectrum kernel (Leslie et al., 2002)
- the number of occurrences of u in $\mathbf{x}$ up to m mismatches (without gaps) : mismatch kernel (Leslie et al., 2004)
- the number of occurrences of u in $\mathbf{x}$ allowing gaps, with a weight decaying exponentially with the number of gaps : substring kernel (Lohdi et al., 2002)

Example: spectrum kernel (1/2)

Kernel definition

- The 3-spectrum of

$$\mathbf{x} = \mathtt{CGGSLIAMMWFGV}$$

is:

$$(\mathtt{CGG},\mathtt{GGS},\mathtt{GSL},\mathtt{SLI},\mathtt{LIA},\mathtt{IAM},\mathtt{AMM},\mathtt{MMW},\mathtt{MWF},\mathtt{WFG},\mathtt{FGV}) \ .$$

- Let $\Phi_u(\mathbf{x})$ denote the number of occurrences of u in $\mathbf{x}$. The k-spectrum kernel is:

$$K\left(\mathbf{x},\mathbf{x}'\right) := \sum_{u\in\mathcal{A}^k} \Phi_u\left(\mathbf{x}\right)\Phi_u\left(\mathbf{x}'\right) \ .$$

Example: spectrum kernel (2/2)

Implementation

- The computation of the kernel is formally a sum over $|\mathcal{A}|^k$ terms, but at most $|\mathbf{x}| - k + 1$ terms are non-zero in $\Phi(\mathbf{x}) \implies$ Computation in $O(|\mathbf{x}| + |\mathbf{x}'|)$ with pre-indexation of the strings.
- Fast classification of a sequence $\mathbf{x}$ in $O(|\mathbf{x}|)$:

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_u w_u \Phi_u(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|-k+1} w_{x_i \ldots x_{i+k-1}}.$$

Remarks

- Work with any string (natural language, time series...)
- Fast and scalable, a good default method for string classification.
- Variants allow matching of k-mers up to m mismatches.

Example 2: Substring kernel (1/11)

Definition

- For $1 \leq k \leq n \in \mathbb{N}$, we denote by $\mathcal{I}(k,n)$ the set of sequences of indices $\mathbf{i} = (i_1, \ldots, i_k)$, with $1 \leq i_1 < i_2 < \ldots < i_k \leq n$.
- For a string $\mathbf{x} = x_1 \ldots x_n \in \mathcal{X}$ of length n, for a sequence of indices $\mathbf{i} \in \mathcal{I}(k,n)$, we define a substring as:

$$\mathbf{x}(\mathbf{i}) := x_{i_1} x_{i_2} \ldots x_{i_k} .$$

- The length of the substring is:

$$l(\mathbf{i}) = i_k - i_1 + 1.$$

Example 2: Substring kernel (2/11)

Example

ABRACADABRA

- $\mathbf{i} = (3, 4, 7, 8, 10)$
- $\mathbf{x}(\mathbf{i}) =$RADAR
- $l(\mathbf{i}) = 10 - 3 + 1 = 8$

Example 2: Substring kernel (3/11)

The kernel

- Let $k \in \mathbb{N}$ and $\lambda \in \mathbb{R}^+$ fixed. For all $\mathbf{u} \in \mathcal{A}^k$, let $\Phi_{\mathbf{u}} : \mathcal{X} \to \mathbb{R}$ be defined by:

$$\forall \mathbf{x} \in \mathcal{X}, \quad \Phi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i} \in \mathcal{I}(k, |\mathbf{x}|) : \quad \mathbf{x}(\mathbf{i}) = \mathbf{u}} \lambda^{l(\mathbf{i})} .$$

- The substring kernel is the p.d. kernel defined by:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad K_{k,\lambda}(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{u} \in \mathcal{A}^k} \Phi_{\mathbf{u}}(\mathbf{x}) \Phi_{\mathbf{u}}(\mathbf{x}') .$$

Example 2: Substring kernel (4/11)

Example

u	ca	ct	at	ba	bt	cr	ar	br
$\Phi_u(\text{cat})$	λ^2	λ^3	λ^2	0	0	0	0	0
$\Phi_u(\text{car})$	λ^2	0	0	0	0	λ^3	λ^2	0
$\Phi_u(\text{bat})$	0	0	λ^2	λ^2	λ^3	0	0	0
$\Phi_u(\text{bar})$	0	0	0	λ^2	0	0	λ^2	λ^3

$$\begin{cases} K(\text{cat,cat}) = K(\text{car,car}) = 2\lambda^4 + \lambda^6 \\ K(\text{cat,car}) = \lambda^4 \\ K(\text{cat,bar}) = 0 \end{cases}$$

Example 2: Substring kernel (5/11)

Kernel computation

- We need to compute, for any pair $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, the kernel:

$$
\begin{aligned}
K_{n,\lambda}\left(\mathbf{x}, \mathbf{x}'\right) &= \sum_{\mathbf{u} \in \mathcal{A}^k} \Phi_{\mathbf{u}}(\mathbf{x})\, \Phi_{\mathbf{u}}\left(\mathbf{x}'\right) \\
&= \sum_{\mathbf{u} \in \mathcal{A}^k} \sum_{\mathbf{i}:\mathbf{x}(\mathbf{i})=\mathbf{u}} \sum_{\mathbf{i}':\mathbf{x}'(\mathbf{i}')=\mathbf{u}} \lambda^{l(\mathbf{i})+l(\mathbf{i}')} .
\end{aligned}
$$

- Enumerating the substrings is too slow (of order $|\mathbf{x}|^k$).

Example 2: Substring kernel (6/11)

Kernel computation (cont.)

- For $\mathbf{u} \in \mathcal{A}^k$ remember that:

$$\Phi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i}:\mathbf{x}(\mathbf{i})=\mathbf{u}} \lambda^{i_n - i_1 + 1}.$$

- Let now:

$$\Psi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i}:\mathbf{x}(\mathbf{i})=\mathbf{u}} \lambda^{|\mathbf{x}| - i_1 + 1}.$$

Example 2: Substring kernel (7/11)

Kernel computation (cont.)

Let us note $\mathbf{x}(1,j) = x_1 \ldots x_j$. A simple rewriting shows that, if we note $a \in \mathcal{A}$ the last letter of $\mathbf{u}$ ($\mathbf{u} = \mathbf{v}a$):

$$\Phi_{\mathbf{v}a}(\mathbf{x}) = \sum_{j \in [1, |\mathbf{x}|] : x_j = a} \Psi_{\mathbf{v}}(\mathbf{x}(1, j-1)) \lambda ,$$

and

$$\Psi_{\mathbf{v}a}(\mathbf{x}) = \sum_{j \in [1, |\mathbf{x}|] : x_j = a} \Psi_{\mathbf{v}}(\mathbf{x}(1, j-1)) \lambda^{|\mathbf{x}| - j + 1} .$$

Example 2: Substring kernel (8/11)

Kernel computation (cont.)

Moreover we observe that if the string is of the form $\mathbf{x}a$ (i.e., the last letter is $a \in \mathcal{A}$), then:

- If the last letter of **u** is not a:

$$\begin{cases} \Phi_{\mathbf{u}}(\mathbf{x}a) & = \Phi_{\mathbf{u}}(\mathbf{x}) \,, \\ \Psi_{\mathbf{u}}(\mathbf{x}a) & = \lambda \Psi_{\mathbf{u}}(\mathbf{x}) \,. \end{cases}$$

- If the last letter of **u** is a (i.e., $\mathbf{u} = \mathbf{v}a$ with $\mathbf{v} \in \mathcal{A}^{n-1}$):

$$\begin{cases} \Phi_{\mathbf{v}a}(\mathbf{x}a) & = \Phi_{\mathbf{v}a}(\mathbf{x}) + \lambda \Psi_{\mathbf{v}}(\mathbf{x}) \,, \\ \Psi_{\mathbf{v}a}(\mathbf{x}a) & = \lambda \Psi_{\mathbf{v}a}(\mathbf{x}) + \lambda \Psi_{\mathbf{v}}(\mathbf{x}) \,. \end{cases}$$

Example 2: Substring kernel (9/11)

Kernel computation (cont.)

Let us now show how the function:

$$B_n\left(\mathbf{x},\mathbf{x}'\right) := \sum_{\mathbf{u}\in\mathcal{A}^n} \Psi_{\mathbf{u}}\left(\mathbf{x}\right)\Psi_{\mathbf{u}}\left(\mathbf{x}'\right)$$

and the kernel:

$$K_n\left(\mathbf{x},\mathbf{x}'\right) := \sum_{\mathbf{u}\in\mathcal{A}^n} \Phi_{\mathbf{u}}\left(\mathbf{x}\right)\Phi_{\mathbf{u}}\left(\mathbf{x}'\right)$$

can be computed recursively. We note that:

$$\begin{cases} B_0\left(\mathbf{x},\mathbf{x}'\right) = K_0\left(\mathbf{x},\mathbf{x}'\right) = 0 & \text{for all } \mathbf{x},\mathbf{x}' \\ B_k\left(\mathbf{x},\mathbf{x}'\right) = K_k\left(\mathbf{x},\mathbf{x}'\right) = 0 & \text{if } \min\left(|\,\mathbf{x}\,|,|\,\mathbf{x}'\,|\right) < k \end{cases}$$

Example 2: Substring kernel (10/11)

Recursive computation of B_n

$$
\begin{aligned}
& B_n\left(\mathbf{x}a, \mathbf{x}'\right) \\
& = \sum_{\mathbf{u}\in\mathcal{A}^n} \Psi_{\mathbf{u}}\left(\mathbf{x}a\right) \Psi_{\mathbf{u}}\left(\mathbf{x}'\right) \\
& = \lambda \sum_{\mathbf{u}\in\mathcal{A}^n} \Psi_{\mathbf{u}}\left(\mathbf{x}\right) \Psi_{\mathbf{u}}\left(\mathbf{x}'\right) + \lambda \sum_{\mathbf{v}\in\mathcal{A}^{n-1}} \Psi_{\mathbf{v}}\left(\mathbf{x}\right) \Psi_{\mathbf{v}a}\left(\mathbf{x}'\right) \\
& = \lambda B_n\left(\mathbf{x}, \mathbf{x}'\right) + \\
& \quad \lambda \sum_{\mathbf{v}\in\mathcal{A}^{n-1}} \Psi_{\mathbf{v}}\left(\mathbf{x}\right) \left(\sum_{j\in[1,|\mathbf{x}'|]:x'_j=a} \Psi_{\mathbf{v}}\left(\mathbf{x}'\left(1, j-1\right)\right) \lambda^{|\mathbf{x}'|-j+1} \right) \\
& = \lambda B_n\left(\mathbf{x}, \mathbf{x}'\right) + \sum_{j\in[1,|\mathbf{x}'|]:x'_j=a} B_{n-1}\left(\mathbf{x}, \mathbf{x}'\left(1, j-1\right)\right) \lambda^{|\mathbf{x}'|-j+2}
\end{aligned}
$$

Example 2: Substring kernel (10/11)

Recursive computation of K_n

$$
\begin{aligned}
& K_n\left(\mathbf{x}a, \mathbf{x}'\right) \\
& = \sum_{\mathbf{u}\in\mathcal{A}^n} \Phi_{\mathbf{u}}\left(\mathbf{x}a\right) \Phi_{\mathbf{u}}\left(\mathbf{x}'\right) \\
& = \sum_{\mathbf{u}\in\mathcal{A}^n} \Phi_{\mathbf{u}}\left(\mathbf{x}\right) \Phi_{\mathbf{u}}\left(\mathbf{x}'\right) + \lambda \sum_{\mathbf{v}\in\mathcal{A}^{n-1}} \Psi_{\mathbf{v}}\left(\mathbf{x}\right) \Phi_{\mathbf{v}a}\left(\mathbf{x}'\right) \\
& = K_n\left(\mathbf{x}, \mathbf{x}'\right) + \\
& \quad \lambda \sum_{\mathbf{v}\in\mathcal{A}^{n-1}} \Psi_{\mathbf{v}}\left(\mathbf{x}\right) \left(\sum_{j\in[1,\mid \mathbf{x}' \mid]:x'_j=a} \Psi_{\mathbf{v}}\left(\mathbf{x}'\left(1, j-1\right)\right) \lambda \right) \\
& = \lambda K_n\left(\mathbf{x}, \mathbf{x}'\right) + \lambda^2 \sum_{j\in[1,\mid \mathbf{x}' \mid]:x'_j=a} B_{n-1}\left(\mathbf{x}, \mathbf{x}'\left(1, j-1\right)\right)
\end{aligned}
$$

Summary: Substring indexation

- Implementation in $O(|\mathbf{x}| + |\mathbf{x}'|)$ in memory and time for the spectrum and mismatch kernels (with suffix trees)
- Implementation in $O(|\mathbf{x}| \times |\mathbf{x}'|)$ in memory and time for the substring kernels
- The feature space has high dimension ($|\mathcal{A}|^k$), so learning requires regularized methods (such as SVM)

Dictionary-based indexation

The approach

- Chose a dictionary of sequences $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$
- Chose a measure of similarity $s(\mathbf{x}, \mathbf{x}')$
- Define the mapping $\Phi_{\mathcal{D}}(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_i))_{\mathbf{x}_i \in \mathcal{D}}$

Dictionary-based indexation

The approach

- Chose a dictionary of sequences $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$
- Chose a measure of similarity $s(\mathbf{x}, \mathbf{x}')$
- Define the mapping $\Phi_{\mathcal{D}}(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_i))_{\mathbf{x}_i \in \mathcal{D}}$

Examples

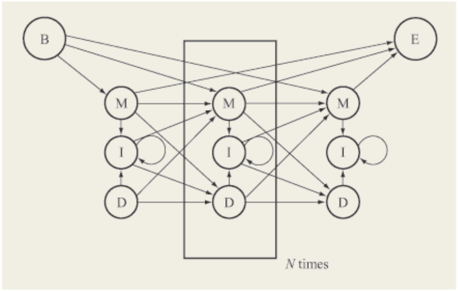
This includes:

- Motif kernels (Logan et al., 2001): the dictionary is a library of motifs, the similarity function is a matching function
- Pairwise kernel (Liao & Noble, 2003): the dictionary is the training set, the similarity is a classical measure of similarity between sequences.

Outline

Probabilistic models for sequences

Probabilistic modeling of biological sequences is older than kernel designs. Important models include HMM for protein sequences, SCFG for RNA sequences.



Parametric model

A model is a family of distribution

$$\{P_\theta, \theta \in \Theta \subset \mathbb{R}^m\} \subset \mathcal{M}_1^+(\mathcal{X})$$

Fisher kernel

Definition

- Fix a parameter $\theta_0 \in \Theta$ (e.g., by maximum likelihood over a training set of sequences)
- For each sequence $\mathbf{x}$, compute the Fisher score vector:

$$\Phi_{\theta_0}(\mathbf{x}) = \nabla_\theta \log P_\theta(\mathbf{x})|_{\theta=\theta_0} \ .$$

- Form the kernel (Jaakkola et al., 1998):

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \Phi_{\theta_0}(\mathbf{x})^\top I(\theta_0)^{-1} \Phi_{\theta_0}(\mathbf{x}') \ ,$$

where $I(\theta_0) = E_{\theta_0}\left[\Phi_{\theta_0}(\mathbf{x})\Phi_{\theta_0}(\mathbf{x})^\top\right]$ is the Fisher information matrix.

Fisher kernel properties

- The Fisher score describes how each parameter contributes to the process of generating a particular example
- The Fisher kernel is invariant under change of parametrization of the model
- A kernel classifier employing the Fisher kernel derived from a model that contains the label as a latent variable is, asymptotically, at least as good a classifier as the MAP labelling based on the model (Jaakkola and Haussler, 1998).
- A variant of the Fisher kernel (called the Tangent of Posterior kernel) can also improve over the direct posterior classification by helping to correct the effect of estimation errors in the parameter (Tsuda et al., 2002).

Fisher kernel in practice

- $\Phi_{\theta_0}(\mathbf{x})$ can be computed explicitly for many models (e.g., HMMs)
- $I(\theta_0)$ is often replaced by the identity matrix
- Several different models (i.e., different θ_0) can be trained and combined
- Feature vectors are explicitly computed

Mutual information kernels

Definition

- Chose a prior $w(d\theta)$ on the measurable set Θ
- Form the kernel (Seeger, 2002):

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \int_{\theta \in \Theta} P_\theta(\mathbf{x}) P_\theta(\mathbf{x}') w(d\theta) \ .$$

- No explicit computation of a finite-dimensional feature vector
- $K\left(\mathbf{x}, \mathbf{x}'\right) = < \varphi\left(\mathbf{x}\right), \varphi\left(\mathbf{x}'\right) >_{L_2(w)}$ with

$$\varphi\left(\mathbf{x}\right) = \left(P_\theta\left(\mathbf{x}\right)\right)_{\theta \in \Theta} \ .$$

Example: coin toss

- Let $P_\theta(X=1)=\theta$ and $P_\theta(X=0)=1-\theta$ a model for random coin toss, with $\theta \in [0,1]$.
- Let $d\theta$ be the Lebesgue measure on $[0,1]$
- The mutual information kernel between $\mathbf{x}=001$ and $\mathbf{x}'=1010$ is:

$$\begin{cases} P_\theta(\mathbf{x}) & = \theta(1-\theta)^2 \ , \\ P_\theta(\mathbf{x}') & = \theta^2(1-\theta)^2 \ , \end{cases}$$

$$K(\mathbf{x},\mathbf{x}') = \int_0^1 \theta^3(1-\theta)^4 \, d\theta = \frac{3!4!}{8!} = \frac{1}{280} \ .$$

Context-tree model

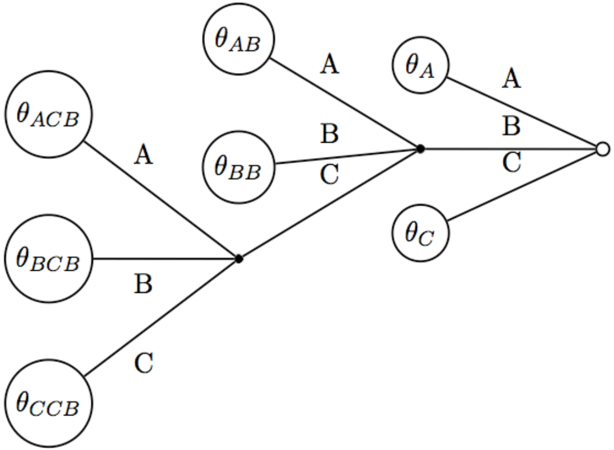
Definition

A context-tree model is a variable-memory Markov chain:

$$P_{\mathcal{D},\theta}(\mathbf{x}) = P_{\mathcal{D},\theta}\left(x_1 \ldots x_D\right) \prod_{i=D+1}^{n} P_{\mathcal{D},\theta}\left(x_i \,|\, x_{i-D} \ldots x_{i-1}\right)$$

- $\mathcal{D}$ is a suffix tree
- $\theta \in \Sigma^{\mathcal{D}}$ is a set of conditional probabilities (multinomials)

Context-tree model: example



$$P(AABACBACC) = P(AAB)\theta_{AB}(A)\theta_A(C)\theta_C(B)\theta_{ACB}(A)\theta_A(C)\theta_C(A) .$$

The context-tree kernel

Theorem (Cuturi et al., 2004)

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

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

  can be computed in $O(|\mathbf{x}|+|\mathbf{x}'|)$ with a variant of the Context-Tree Weighting algorithm.
- *This is a valid mutual information kernel.*
- *The similarity is related to information-theoretical measure of mutual information between strings.*

Marginalized kernels

Definition

- For any observed data $\mathbf{x} \in \mathcal{X}$, let a latent variable $\mathbf{y} \in \mathcal{Y}$ be associated probabilistically through a conditional probability $P_{\mathbf{x}}(d\mathbf{y})$.
- Let $K_{\mathcal{Z}}$ be a kernel for the complete data $\mathbf{z} = (\mathbf{x}, \mathbf{y})$
- Then the following kernel is a valid kernel on $\mathcal{X}$, called a marginalized kernel (Kin et al., 2002):

$$\begin{aligned} K_{\mathcal{X}}\left(\mathbf{x}, \mathbf{x}'\right) &:= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) \\ &= \int \int K_{\mathcal{Z}}\left((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')\right) P_{\mathbf{x}}(d\mathbf{y}) P_{\mathbf{x}'}\left(d\mathbf{y}'\right) . \end{aligned}$$

Marginalized kernels: proof of positive definiteness

- $K_{\mathcal{Z}}$ is p.d. on $\mathcal{Z}$. Therefore there exists a Hilbert space $\mathcal{H}$ and $\Phi_{\mathcal{Z}} : \mathcal{Z} \to \mathcal{H}$ such that:

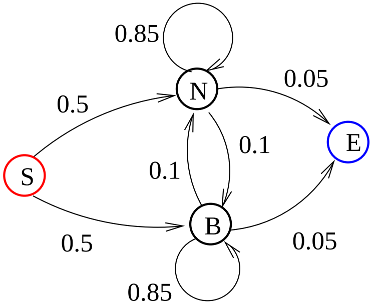
$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = \left\langle \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right)\right\rangle_{\mathcal{H}} \; .$$

- Marginalizing therefore gives:

$$\begin{aligned} K_{\mathcal{X}}\left(\mathbf{x}, \mathbf{x}'\right) &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) \\ &= E_{P_{\mathbf{x}}(d\mathbf{y}) \times P_{\mathbf{x}'}(d\mathbf{y}')} \left\langle \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right)\right\rangle_{\mathcal{H}} \\ &= \left\langle E_{P_{\mathbf{x}}(d\mathbf{y})} \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), E_{P_{\mathbf{x}}(d\mathbf{y}')} \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right)\right\rangle_{\mathcal{H}} \; , \end{aligned}$$

therefore $K_{\mathcal{X}}$ is p.d. on $\mathcal{X}$. $\square$

Example: HMM for normal/biased coin toss



- Normal (N) and biased (B) coins (not observed)
- Observed output are 0/1 with probabilities:

$$\begin{cases} \pi(0|N) = 1 - \pi(1|N) = 0.5, \\ \pi(0|B) = 1 - \pi(1|B) = 0.8. \end{cases}$$

- Example of realization (complete data):

```
NNNNNBBBBBBBBBNNNNNNNNNNNNBBBBBB
10010111011110100101110011111011
```

1-spectrum kernel on complete data

- If both $\mathbf{x} \in \mathcal{A}^*$ and $\mathbf{y} \in \mathcal{S}^*$ were observed, we might rather use the 1-spectrum kernel on the complete data $\mathbf{z} = (\mathbf{x}, \mathbf{y})$:

$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = \sum_{(a,s) \in \mathcal{A} \times \mathcal{S}} n_{a,s}\left(\mathbf{z}\right) n_{a,s}\left(\mathbf{z}\right),$$

  where $n_{a,s}\left(\mathbf{x}, \mathbf{y}\right)$ for $a = 0, 1$ and $s = N, B$ is the number of occurrences of s in $\mathbf{y}$ which emit a in $\mathbf{x}$.

- Example:

$$\mathbf{z} = 1001011101111010010111100 1111011,$$
$$\mathbf{z}' = 00110101100111110110101111101100101,$$

$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = n_0\left(\mathbf{z}\right) n_0\left(\mathbf{z}'\right) + n_0\left(\mathbf{z}\right) n_0\left(\mathbf{z}'\right) + n_1\left(\mathbf{z}\right) n_1\left(\mathbf{z}'\right) + n_1\left(\mathbf{z}\right) n_1\left(\mathbf{z}'\right)$$
$$= 7 \times 15 + 9 \times 12 + 13 \times 6 + 2 \times 1 = 293.$$

1-spectrum marginalized kernel on observed data

- The marginalized kernel for observed data is:

$$K_{\mathcal{X}}\left(\mathbf{x},\mathbf{x}'\right) = \sum_{\mathbf{y},\mathbf{y}'\in\mathcal{S}^*} K_{\mathcal{Z}}\left(\left(\mathbf{x},\mathbf{y}\right),\left(\mathbf{x},\mathbf{y}\right)\right) P\left(\mathbf{y}|\mathbf{x}\right) P\left(\mathbf{y}'|\mathbf{x}'\right)$$
$$= \sum_{(a,s)\in\mathcal{A}\times\mathcal{S}} \Phi_{a,s}\left(\mathbf{x}\right)\Phi_{a,s}\left(\mathbf{x}'\right),$$

with

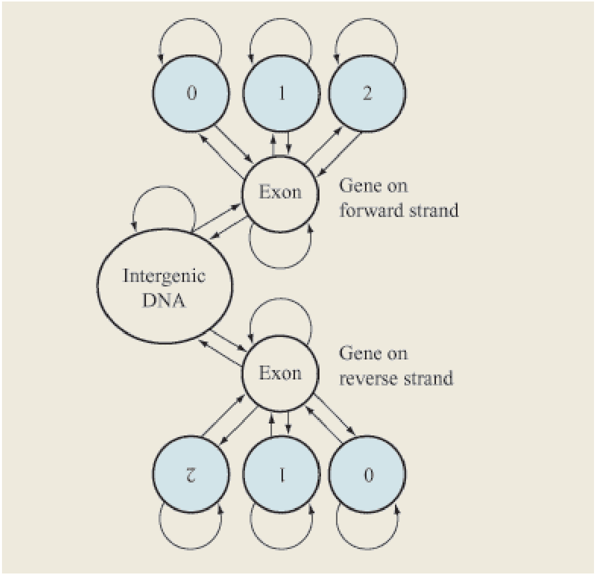
$$\Phi_{a,s}\left(\mathbf{x}\right) = \sum_{\mathbf{y}\in\mathcal{S}^*} P\left(\mathbf{y}|\mathbf{x}\right) n_{a,s}\left(\mathbf{x},\mathbf{y}\right)$$

Computation of the 1-spectrum marginalized kernel

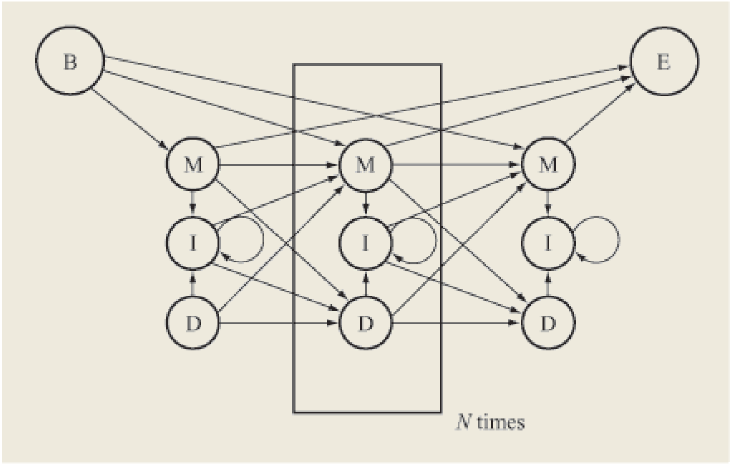
$$
\begin{aligned}
\Phi_{a,s}(\mathbf{x}) &= \sum_{\mathbf{y}\in\mathcal{S}^*} P(\mathbf{y}|\mathbf{x})\, n_{a,s}(\mathbf{x},\mathbf{y}) \\
&= \sum_{\mathbf{y}\in\mathcal{S}^*} P(\mathbf{y}|\mathbf{x}) \left\{ \sum_{i=1}^{n} \delta(x_i, a)\, \delta(y_i, s) \right\} \\
&= \sum_{i=1}^{n} \delta(x_i, a) \left\{ \sum_{\mathbf{y}\in\mathcal{S}^*} P(\mathbf{y}|\mathbf{x})\, \delta(y_i, s) \right\} \\
&= \sum_{i=1}^{n} \delta(x_i, a)\, P(y_i = s|\mathbf{x})\,.
\end{aligned}
$$

and $P(y_i = s|\mathbf{x})$ can be computed efficiently by forward-backward algorithm!

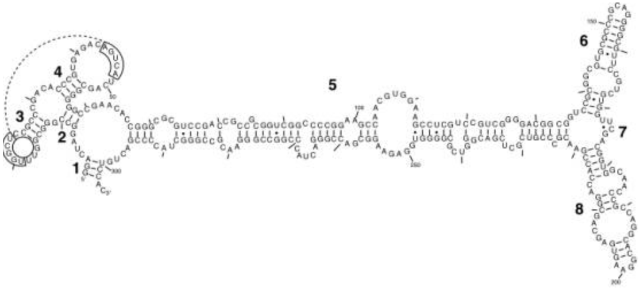
HMM example (DNA)



HMM example (protein)



SCFG for RNA sequences



SFCG rules

- $S \rightarrow SS$
- $S \rightarrow aSa$
- $S \rightarrow aS$
- $S \rightarrow a$

Marginalized kernel (Kin et al., 2002)

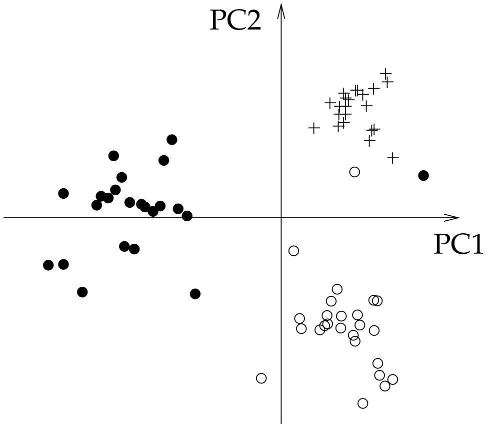
- Feature: number of occurrences of each (base,state) combination
- Marginalization using classical inside/outside algorithm

Marginalized kernels in practice

Examples

- Spectrum kernel on the hidden states of a HMM for protein sequences (Tsuda et al., 2002)
- Kernels for RNA sequences based on SCFG (Kin et al., 2002)
- Kernels for graphs based on random walks on graphs (Kashima et al., 2004)
- Kernels for multiple alignments based on phylogenetic models (Vert et al., 2005)

Marginalized kernels: example



A set of 74 human tRNA sequences is analyzed using a kernel for sequences (the second-order marginalized kernel based on SCFG). This set of tRNAs contains three classes, called Ala-AGC (*white circles*), Asn-GTT (*black circles*) and Cys-GCA (*plus symbols*) (from Tsuda et al., 2003).

Outline

Sequence alignment

Motivation

How to compare 2 sequences?

$$\mathbf{x}_1 = \texttt{CGGSLIAMMWFGV}$$
$$\mathbf{x}_2 = \texttt{CLIVMMNRLMWFGV}$$

Find a good alignment:

```
CGGSLIAMM------WFGV
|...|||||....||||
C-----LIVMMNRLMWFGV
```

Alignment score

In order to quantify the relevance of an alignment π, define:

- a substitution matrix $S \in \mathbb{R}^{\mathcal{A}\times\mathcal{A}}$
- a gap penalty function $g : \mathbb{N} \to \mathbb{R}$

Any alignment is then scored as follows

```
CGGSLIAMM------WFGV
 |...|||||....||||
C----LIVMMNRLMWFGV
```

$$
\begin{aligned}
s_{S,g}(\pi) = {} & S(C,C) + S(L,L) + S(I,I) + S(A,V) + 2S(M,M) \\
& + S(W,W) + S(F,F) + S(G,G) + S(V,V) - g(3) - g(4)
\end{aligned}
$$

Local alignment kernel

Smith-Waterman score

- The widely-used Smith-Waterman local alignment score is defined by:

$$SW_{S,g}(\mathbf{x},\mathbf{y}) := \max_{\pi\in\Pi(\mathbf{x},\mathbf{y})} s_{S,g}(\pi).$$

- It is symmetric, but not positive definite...

Local alignment kernel

Smith-Waterman score

- The widely-used Smith-Waterman local alignment score is defined by:

$$SW_{S,g}(\mathbf{x},\mathbf{y}) := \max_{\pi\in\Pi(\mathbf{x},\mathbf{y})} s_{S,g}(\pi).$$

- It is symmetric, but not positive definite...

LA kernel

The local alignment kernel:

$$K_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = \sum_{\pi\in\Pi(\mathbf{x},\mathbf{y})} \exp\left(\beta s_{S,g}(\mathbf{x},\mathbf{y},\pi)\right),$$

is symmetric positive definite.

LA kernel is p.d.: proof (1/11)

Lemma

- If K_1 and K_2 are p.d. kernels, then:

$$K_1 + K_2,$$
$$K_1 K_2, \text{ and}$$
$$cK_1, \text{ for } c \geq 0,$$

  are also p.d. kernels

- If $(K_i)_{i \geq 1}$ is a sequence of p.d. kernels that converges pointwisely to a function K:

$$\forall \left(\mathbf{x}, \mathbf{x}'\right) \in \mathcal{X}^2, \quad K\left(\mathbf{x}, \mathbf{x}'\right) = \lim_{n \to \infty} K_i\left(\mathbf{x}, \mathbf{x}'\right),$$

  then K is also a p.d. kernel.

LA kernel is p.d.: proof (2/11)

Proof of lemma

Let A and B be $n \times n$ positive semidefinite matrices. By diagonalization of A:

$$A_{i,j} = \sum_{p=1}^{n} f_p(i) f_p(j)$$

for some vectors $f_1, \ldots, f_n$. Then, for any $\boldsymbol{\alpha} \in \mathbb{R}^n$:

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j A_{i,j} B_{i,j} = \sum_{p=1}^{n} \sum_{i,j=1}^{n} \alpha_i f_p(i) \alpha_j f_p(j) B_{i,j} \geq 0.$$

The matrix $C_{i,j} = A_{i,j} B_{i,j}$ is therefore p.d. Other properties are obvious from definition. $\square$

LA kernel is p.d.: proof (3/11)

Lemma (direct sum and product of kernels)

Let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$. Let K_1 be a p.d. kernel on $\mathcal{X}_1$, and K_2 be a p.d. kernel on $\mathcal{X}_2$. Then the following functions are p.d. kernels on $\mathcal{X}$:

- the direct sum,

$$K\left(\left(\mathbf{x}_1, \mathbf{x}_2\right), \left(\mathbf{y}_1, \mathbf{y}_2\right)\right) = K_1\left(\mathbf{x}_1, \mathbf{y}_1\right) + K_2\left(\mathbf{x}_2, \mathbf{y}_2\right),$$

- The direct product:

$$K\left(\left(\mathbf{x}_1, \mathbf{x}_2\right), \left(\mathbf{y}_1, \mathbf{y}_2\right)\right) = K_1\left(\mathbf{x}_1, \mathbf{y}_1\right) K_2\left(\mathbf{x}_2, \mathbf{y}_2\right).$$

LA kernel is p.d.: proof (4/11)

Proof of lemma

If K_1 is a p.d. kernel, let $\Phi_1 : \mathcal{X}_1 \mapsto \mathcal{H}$ be such that:

$$K_1\left(\mathbf{x}_1, \mathbf{y}_1\right) = \left\langle \Phi_1\left(\mathbf{x}_1\right), \Phi_1\left(\mathbf{y}_1\right)\right\rangle_{\mathcal{H}}.$$

Let $\Phi : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{H}$ be defined by:

$$\Phi\left(\left(\mathbf{x}_1, \mathbf{x}_2\right)\right) = \Phi_1\left(\mathbf{x}_1\right).$$

Then for $\mathbf{x} = \left(\mathbf{x}_1, \mathbf{x}_2\right)$ and $\mathbf{y} = \left(\mathbf{y}_1, \mathbf{y}_2\right) \in \mathcal{X}$, we get

$$\left\langle \Phi\left(\left(\mathbf{x}_1, \mathbf{x}_2\right)\right), \Phi\left(\left(\mathbf{y}_1, \mathbf{y}_2\right)\right)\right\rangle_{\mathcal{H}} = K_1\left(\mathbf{x}_1, \mathbf{x}_2\right),$$

which shows that $K\left(\mathbf{x}, \mathbf{y}\right) := K_1\left(\mathbf{x}_1, \mathbf{y}_1\right)$ is p.d. on $\mathcal{X}_1 \times \mathcal{X}_2$. The lemma follows from the properties of sums and products of p.d. kernels. $\square$

LA kernel is p.d.: proof (5/11)

Lemma: kernel for sets

Let K be a p.d. kernel on $\mathcal{X}$, and let $\mathcal{P}(\mathcal{X})$ be the set of finite subsets of $\mathcal{X}$. Then the function K_P on $\mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X})$ defined by:

$$\forall A, B \in \mathcal{P}(\mathcal{X}), \quad K_P(A, B) := \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in B} K(\mathbf{x}, \mathbf{y})$$

is a p.d. kernel on $\mathcal{P}(\mathcal{X})$.

LA kernel is p.d.: proof (6/11)

Proof of lemma

Let $\Phi : \mathcal{X} \mapsto \mathcal{H}$ be such that

$$K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}} .$$

Then, for $A, B \in \mathcal{P}(\mathcal{X})$, we get:

$$\begin{aligned} K_P(A, B) &= \sum_{\mathbf{x} \in A} \sum_{\mathbf{y} \in B} \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}} \\ &= \left\langle \sum_{\mathbf{x} \in A} \Phi(\mathbf{x}), \sum_{\mathbf{y} \in B} \Phi(\mathbf{y}) \right\rangle_{\mathcal{H}} \\ &= \langle \Phi_P(A), \Phi_P(B) \rangle_{\mathcal{H}} , \end{aligned}$$

with $\Phi_P(A) := \sum_{\mathbf{x} \in A} \Phi(\mathbf{x})$. $\square$

LA kernel is p.d.: proof (7/11)

Definition: Convolution kernel (Haussler, 1999)

Let K_1 and K_2 be two p.d. kernels for strings. The convolution of K_1 and K_2, denoted $K_1 \star K_2$, is defined for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ by:

$$K_1 \star K_2(\mathbf{x}, \mathbf{y}) := \sum_{\mathbf{x}_1\mathbf{x}_2=\mathbf{x}, \mathbf{y}_1\mathbf{y}_2=\mathbf{y}} K_1(\mathbf{x}_1, \mathbf{y}_1) K_2(\mathbf{x}_2, \mathbf{y}_2).$$

Lemma

If K_1 and K_2 are p.d. then $K_1 \star K_2$ is p.d..

LA kernel is p.d.: proof (8/11)

Proof of lemma

Let $\mathcal{X}$ be the set of finite-length strings. For $\mathbf{x} \in \mathcal{X}$, let

$$R(\mathbf{x}) = \{(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{X} \times \mathcal{X} : \mathbf{x} = \mathbf{x}_1\mathbf{x}_2\} \subset \mathcal{X} \times \mathcal{X} \,.$$

We can then write

$$K_1 \star K_2(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x}_1,\mathbf{x}_2)\in R(\mathbf{x})} \sum_{(\mathbf{y}_1,\mathbf{y}_2)\in R(\mathbf{y})} K_1(\mathbf{x}_1, \mathbf{y}_1) K_2(\mathbf{x}_2, \mathbf{y}_2)$$

which is a p.d. kernel by the previous lemmas. $\square$

LA kernel is p.d.: proof (9/11)

3 basic string kernels

- The constant kernel:

$$K_0(\mathbf{x},\mathbf{y}) := 1 .$$

- A kernel for letters:

$$K_a^{(\beta)}(\mathbf{x},\mathbf{y}) := \begin{cases} 0 & \text{if } |\,\mathbf{x}\,| \neq 1 \text{ where } |\,\mathbf{y}\,| \neq 1 , \\ \exp\left(\beta S(\mathbf{x},\mathbf{y})\right) & \text{otherwise} . \end{cases}$$

- A kernel for gaps:

$$K_g^{(\beta)}(\mathbf{x},\mathbf{y}) = \exp\left[\beta\left(g\left(|\,\mathbf{x}\,|\right) + g\left(|\,\mathbf{x}\,|\right)\right)\right] .$$

LA kernel is p.d.: proof (10/11)

Remark

- $S : \mathcal{A}^2 \to \mathbb{R}$ is the similarity function between letters used in the alignment score. $K_a^{(\beta)}$ is only p.d. when the matrix:

$$\left(\exp\left(\beta s(a,b)\right)\right)_{(a,b)\in\mathcal{A}^2}$$

  is positive semidefinite (this is true for all β when s is conditionally p.d..
- g is the gap penalty function used in alignment score. The gap kernel is always p.d. (with no restriction on g) because it can be written as:

$$K_g^{(\beta)}(\mathbf{x},\mathbf{y}) = \exp\left(\beta g\left(\mid\mathbf{x}\mid\right)\right) \times \exp\left(\beta g\left(\mid\mathbf{y}\mid\right)\right) \,.$$

LA kernel is p.d.: proof (11/11)

Lemma

The local alignment kernel is a (limit) of convolution kernel:

$$K_{LA}^{(\beta)} = \sum_{n=0}^{\infty} K_0 \star \left(K_a^{(\beta)} \star K_g^{(\beta)} \right)^{(n-1)} \star K_a^{(\beta)} \star K_0 .$$

As such it is p.d..

Proof (sketch)

- By induction on n (simple but long to write).
- See details in Vert et al. (2004).

LA kernel computation

- We assume an affine gap penalty:

$$\begin{cases} g(0) & = 0, \\ g(n) & = d + e(n-1) \text{ si } n \geq 1, \end{cases}$$

- The LA kernel can then be computed by dynamic programming by:

$$K_{LA}^{(\beta)}(\mathbf{x},\mathbf{y}) = 1 + X_2(|\mathbf{x}|,|\mathbf{y}|) + Y_2(|\mathbf{x}|,|\mathbf{y}|) + M(|\mathbf{x}|,|\mathbf{y}|),$$

where $M(i,j), X(i,j), Y(i,j), X_2(i,j)$, and $Y_2(i,j)$ for $0 \leq i \leq |\mathbf{x}|$, and $0 \leq j \leq |\mathbf{y}|$ are defined recursively.

LA kernel is p.d.: proof (/)

Initialization

$$\begin{cases} M(i,0) = M(0,j) = 0, \\ X(i,0) = X(0,j) = 0, \\ Y(i,0) = Y(0,j) = 0, \\ X_2(i,0) = X_2(0,j) = 0, \\ Y_2(i,0) = Y_2(0,j) = 0, \end{cases}$$

LA kernel is p.d.: proof (/)

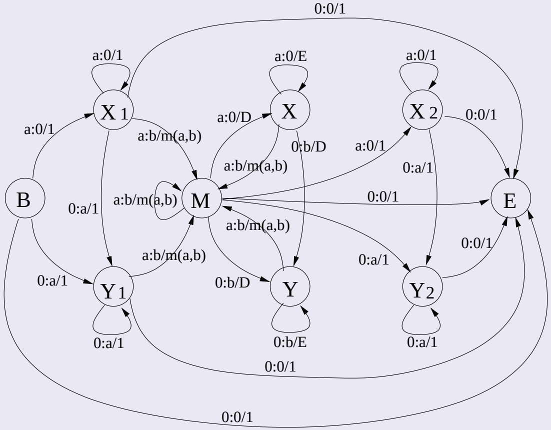
Recursion

For $i = 1, \ldots, |\mathbf{x}|$ and $j = 1, \ldots, |\mathbf{y}|$:

$$\begin{cases} M(i,j) &= \exp(\beta S(x_i, y_j)) \Big[1 + X(i-1, j-1) \\ & \qquad\qquad + Y(i-1, j-1) + M(i-1, j-1) \Big], \\ X(i,j) &= \exp(\beta d) M(i-1, j) + \exp(\beta e) X(i-1, j), \\ Y(i,j) &= \exp(\beta d) \left[M(i, j-1) + X(i, j-1) \right] \\ & \qquad\qquad + \exp(\beta e) Y(i, j-1), \\ X_2(i,j) &= M(i-1, j) + X_2(i-1, j), \\ Y_2(i,j) &= M(i, j-1) + X_2(i, j-1) + Y_2(i, j-1). \end{cases}$$

LA kernel in practice

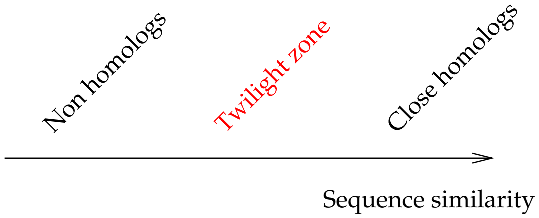
- Implementation by a finite-state transducer in $O(|\mathbf{x}| \times |\mathbf{x}'|)$



- In practice, values are too large (exponential scale) so taking its logarithm is a safer choice (but not p.d. anymore!)

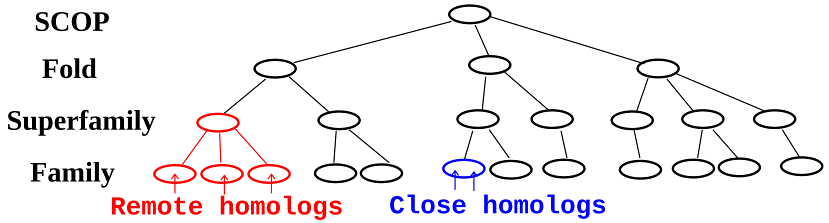
Outline

Remote homology



- Homologs have common ancestors
- Structures and functions are more conserved than sequences
- Remote homologs can not be detected by direct sequence comparison

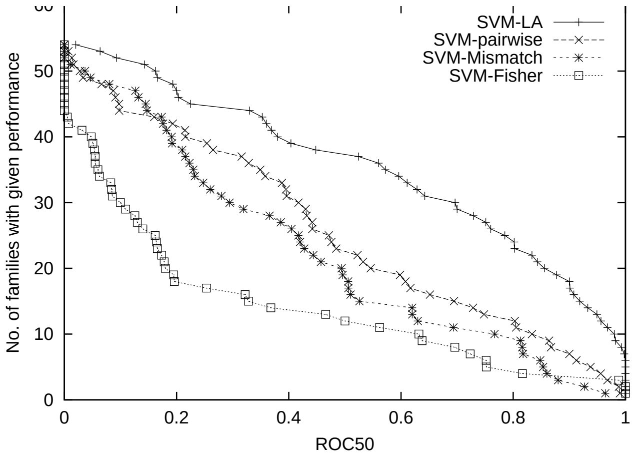
SCOP database



A benchmark experiment

- Goal: recognize directly the superfamily
- Training: for a sequence of interest, positive examples come from the same superfamily, but different families. Negative from other superfamilies.
- Test: predict the superfamily.

Difference in performance



Performance on the SCOP superfamily recognition benchmark (from Vert et al., 2004).

String kernels: Summary

- A variety of principles for string kernel design have been proposed.
- Good kernel design is important for each data and each task. Performance is not the only criterion.
- Still an art, although principled ways have started to emerge.
- Fast implementation with string algorithms is often possible.
- Their application goes well beyond computational biology.

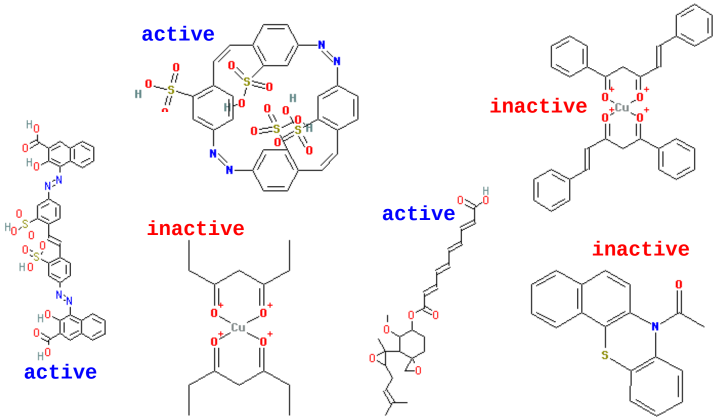
Outline

Part 3

Kernels for graphs

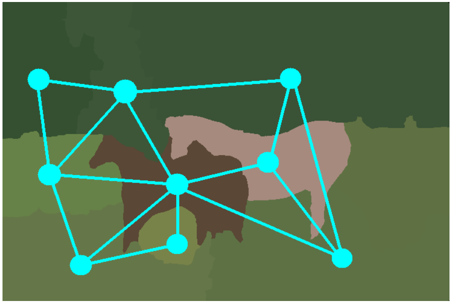
Outline

Virtual screening for drug discovery



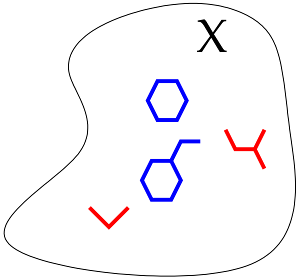
NCI AIDS screen results (from http://cactus.nci.nih.gov).

Image retrieval and classification



From Harchaoui and Bach (2007).

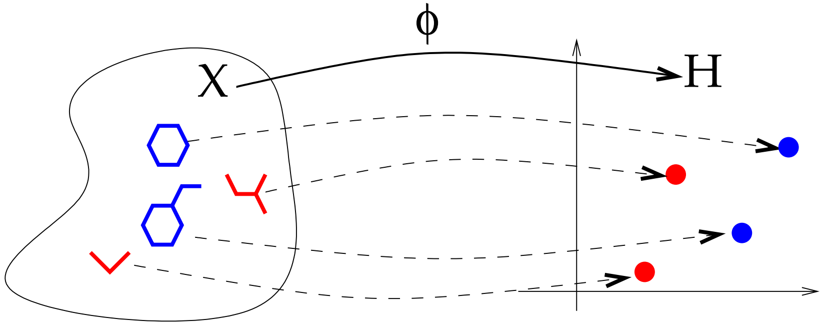
Our approach



Our approach

1. Represent each graph x by a vector $\Phi(x) \in \mathcal{H}$, either explicitly or implicitly through the kernel

$$K(x, x') = \Phi(x)^\top \Phi(x') .$$

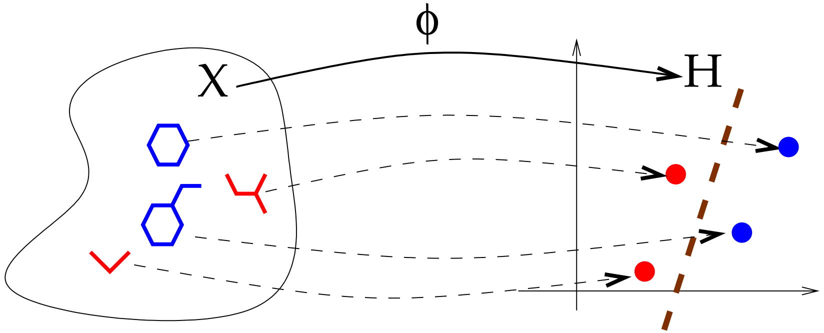


Our approach

1. Represent each graph x by a vector $\Phi(x) \in \mathcal{H}$, either explicitly or implicitly through the kernel

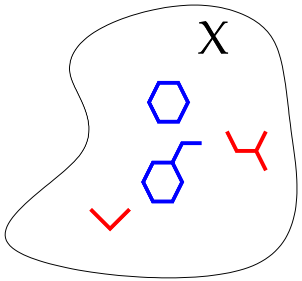
$$K(x, x') = \Phi(x)^\top \Phi(x') \,.$$

2. Use a linear method for classification in $\mathcal{H}$.



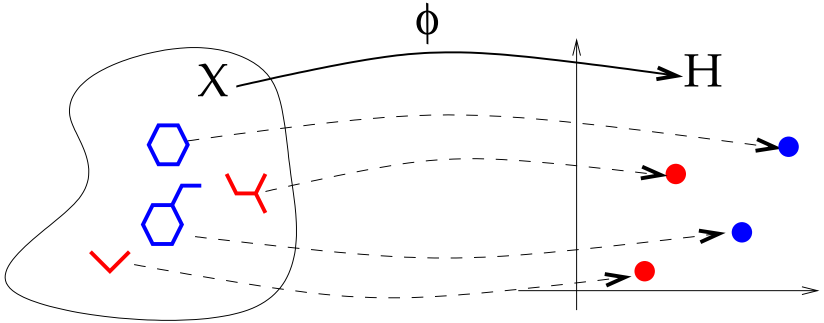
Outline

The approach



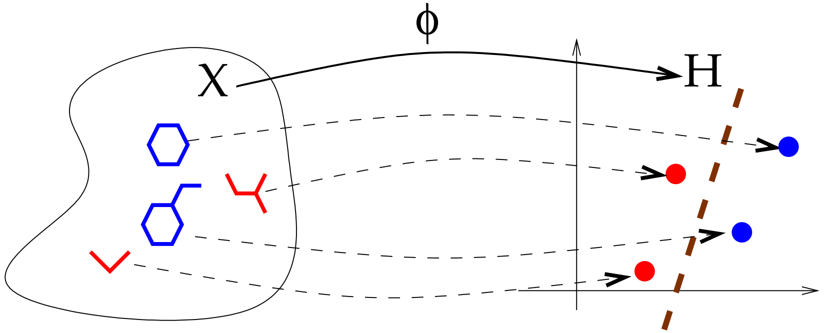
The approach

1. Represent explicitly each graph x by a vector of fixed dimension $\Phi(x) \in \mathbb{R}^p$.



The approach

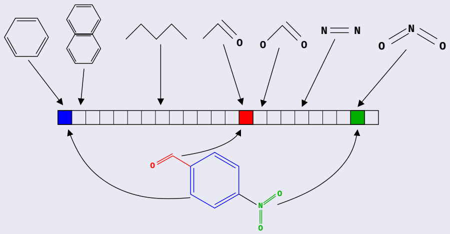
1. Represent explicitly each graph x by a vector of fixed dimension $\Phi(x) \in \mathbb{R}^p$.
2. Use an algorithm for regression or pattern recognition in $\mathbb{R}^p$.



Example

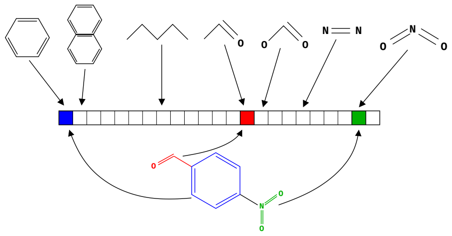
2D structural keys in chemoinformatics

- Index a molecule by a binary fingerprint defined by a limited set of pre-defined stuctures



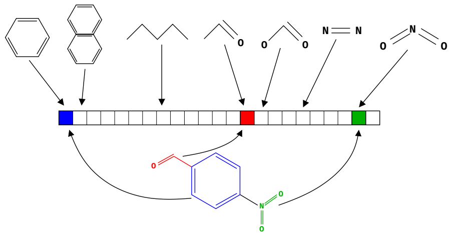
- Use a machine learning algorithms such as SVM, NN, PLS, decision tree, ...

Challenge: which descriptors (patterns)?



- Expressiveness: they should retain as much information as possible from the graph
- Computation : they should be fast to compute
- Large dimension of the vector representation: memory storage, speed, statistical issues

Indexing by substructures

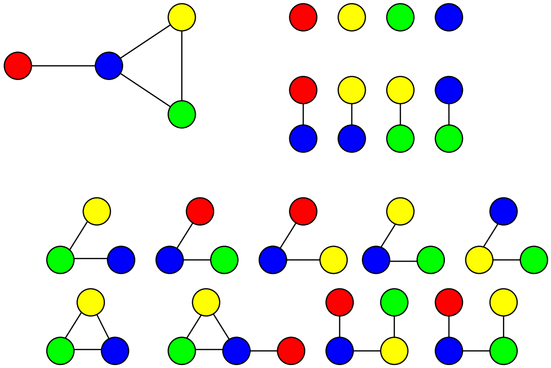


- Often we believe that the presence substructures are important predictive patterns
- Hence it makes sense to represent a graph by features that indicate the presence (or the number of occurrences) of particular substructures
- However, detecting the presence of particular substructures may be computationally challenging...

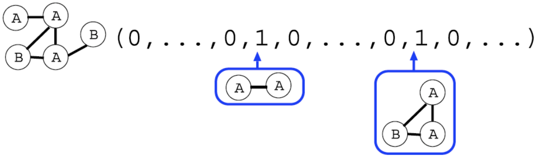
Subgraphs

Definition

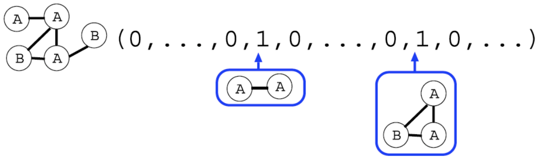
A subgraph of a graph (V, E) is a connected graph (V', E') with $V' \subset V$ and $E' \subset E$.



Indexing by all subgraphs?



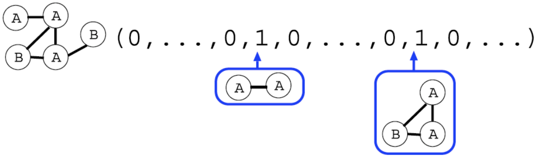
Indexing by all subgraphs?



Theorem

Computing all subgraph occurrences is NP-hard.

Indexing by all subgraphs?



Theorem

Computing all subgraph occurrences is NP-hard.

Proof.

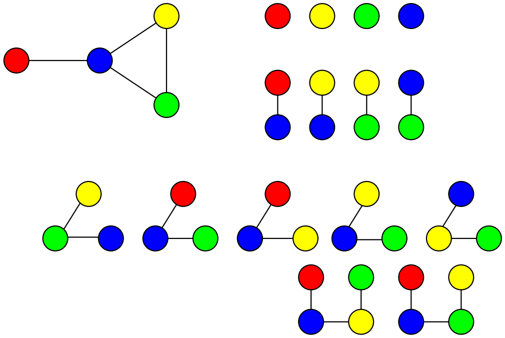
- The linear graph of size n is a subgraph of a graph X with n vertices iff X has an Hamiltonian path
- The decision problem whether a graph has a Hamiltonian path is NP-complete.

□

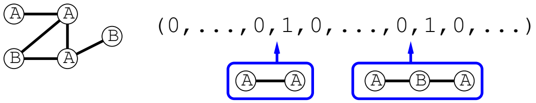
Paths

Definition

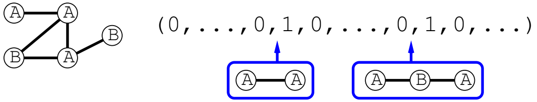
- A path of a graph (V, E) is sequence of distinct vertices $v_1, \ldots, v_n \in V$ $(i \neq j \implies v_i \neq v_j)$ such that $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, n-1$.
- Equivalently the paths are the linear subgraphs.



Indexing by all paths?



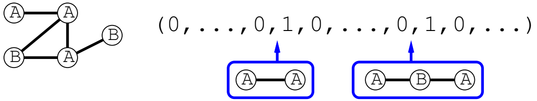
Indexing by all paths?



Theorem

Computing all path occurrences is NP-hard.

Indexing by all paths?



Theorem

Computing all path occurrences is NP-hard.

Proof.

Same as for subgraphs. □

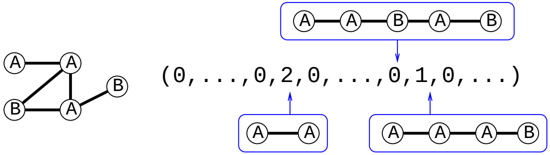
Indexing by what?

Substructure selection

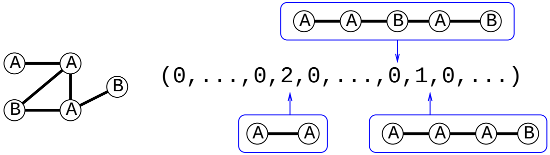
We can imagine more limited sets of substuctures that lead to more computationnally efficient indexing (non-exhaustive list)

- substructures selected by domain knowledge (MDL fingerprint)
- all path up to length k (Openeye fingerprint, Nicholls 2005)
- all shortest paths (Borgwardt and Kriegel, 2005)
- all subgraphs up to k vertices (graphlet kernel, Sherashidze et al., 2009)
- all frequent subgraphs in the database (Helma et al., 2004)

Example : Indexing by all shortest paths



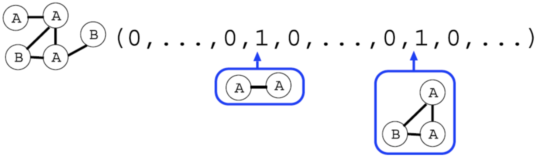
Example : Indexing by all shortest paths



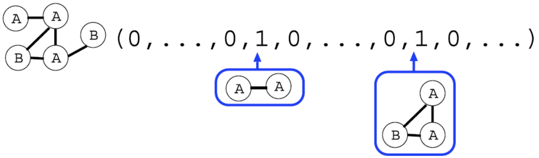
Properties (Borgwardt and Kriegel, 2005)

- There are $O(n^2)$ shortest paths.
- The vector of counts can be computed in $O(n^4)$ with the Floyd-Warshall algorithm.

Example : Indexing by all subgraphs up to k vertices



Example : Indexing by all subgraphs up to k vertices



Properties (Shervashidze et al., 2009)

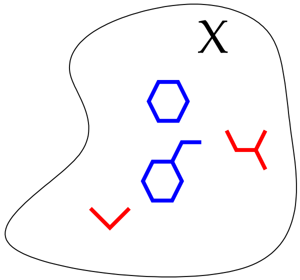
- Naive enumeration scales as $O(n^k)$.
- Enumeration of connected graphlets in $O(nd^{k-1})$ for graphs with degree $\leq d$ and $k \leq 5$.
- Randomly sample subgraphs if enumeration is infeasible.

Summary

- Explicit computation of substructure occurrences can be computationnally prohibitive (subgraph, paths)
- Several ideas to reduce the set of substructures considered
- In practice, NP-hardness may not be so prohibitive (e.g., graphs with small degrees), the strategy followed should depend on the data considered.

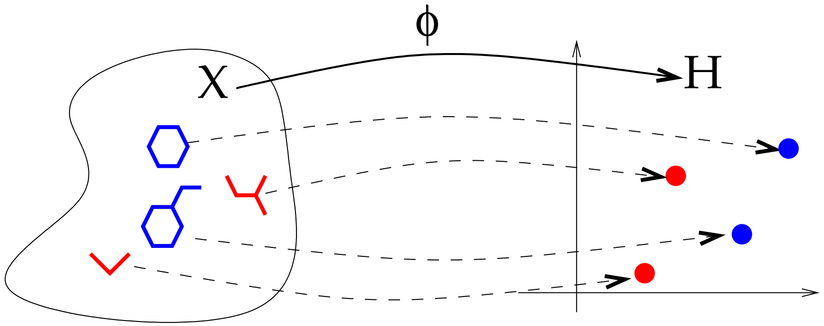
Outline

The idea



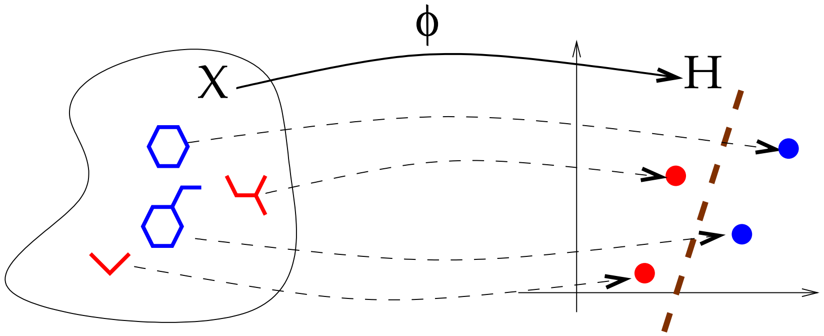
The idea

1. Represent implicitly each graph x by a vector $\Phi(x) \in \mathcal{H}$ through the kernel
$$K(x, x') = \Phi(x)^\top \Phi(x')\,.$$



The idea

1. Represent implicitly each graph x by a vector $\Phi(x) \in \mathcal{H}$ through the kernel
$$K(x, x') = \Phi(x)^\top \Phi(x')\,.$$
2. Use a kernel method for classification in $\mathcal{H}$.



Expressiveness vs Complexity

Definition: Complete graph kernels

A graph kernel is complete if it separates non-isomorphic graphs, i.e.:

$$\forall G_1, G_2 \in \mathcal{X}, \quad d_K(G_1, G_2) = 0 \implies G_1 \simeq G_2 \, .$$

Equivalently, $\Phi(G_1) \neq \Phi(G_1)$ if G_1 and G_2 are not isomorphic.

Expressiveness vs Complexity

Definition: Complete graph kernels

A graph kernel is complete if it separates non-isomorphic graphs, i.e.:

$$\forall G_1, G_2 \in \mathcal{X}, \quad d_K(G_1, G_2) = 0 \implies G_1 \simeq G_2 \, .$$

Equivalently, $\Phi(G_1) \neq \Phi(G_1)$ if G_1 and G_2 are not isomorphic.

Expressiveness vs Complexity trade-off

- If a graph kernel is not complete, then there is no hope to learn all possible functions over $\mathcal{X}$: the kernel is not expressive enough.
- On the other hand, kernel computation must be tractable, i.e., no more than polynomial (with small degree) for practical applications.
- Can we define tractable and expressive graph kernels?

Complexity of complete kernels

Proposition (Gärtner et al., 2003)

Computing any complete graph kernel is at least as hard as the graph isomorphism problem.

Complexity of complete kernels

Proposition (Gärtner et al., 2003)

Computing any complete graph kernel is at least as hard as the graph isomorphism problem.

Proof

- For any kernel K the complexity of computing d_K is the same as the complexity of computing K, because:

$$d_K(G_1, G_2)^2 = K(G_1, G_1) + K(G_2, G_2) - 2K(G_1, G_2) .$$

- If K is a complete graph kernel, then computing d_K solves the graph isomorphism problem ($d_K(G_1, G_2) = 0$ iff $G_1 \simeq G_2$). $\square$

Subgraph kernel

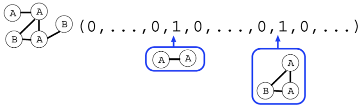
Definition

- Let $(\lambda_G)_{G \in \mathcal{X}}$ a set or nonnegative real-valued weights
- For any graph $G \in \mathcal{X}$, let

$$\forall H \in \mathcal{X}, \quad \Phi_H(G) = \left| \left\{ G' \text{ is a subgraph of } G \; : \; G' \simeq H \right\} \right| .$$

- The subgraph kernel between any two graphs G_1 and $G_2 \in \mathcal{X}$ is defined by:

$$K_{subgraph}(G_1, G_2) = \sum_{H \in \mathcal{X}} \lambda_H \Phi_H(G_1) \Phi_H(G_2) .$$



Subgraph kernel complexity

Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is NP-hard.

Subgraph kernel complexity

Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is NP-hard.

Proof (1/2)

- Let P_n be the path graph with n vertices.
- Subgraphs of P_n are path graphs:

$$\Phi(P_n) = ne_{P_1} + (n-1)e_{P_2} + \ldots + e_{P_n} .$$

- The vectors $\Phi(P_1), \ldots, \Phi(P_n)$ are linearly independent, therefore:

$$e_{P_n} = \sum_{i=1}^{n} \alpha_i \Phi(P_i) ,$$

where the coefficients α_i can be found in polynomial time (solving a $n \times n$ triangular system).

Subgraph kernel complexity

Proposition (Gärtner et al., 2003)

Computing the subgraph kernel is NP-hard.

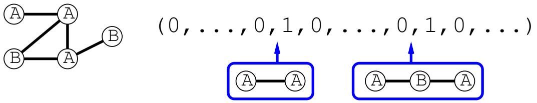
Proof (2/2)

- If G is a graph with n vertices, then it has a path that visits each node exactly once (Hamiltonian path) if and only if $\Phi(G)^\top e_n > 0$, i.e.,

$$\Phi(G)^\top \left(\sum_{i=1}^{n} \alpha_i \Phi(P_i) \right) = \sum_{i=1}^{n} \alpha_i K_{subgraph}(G, P_i) > 0 \,.$$

- The decision problem whether a graph has a Hamiltonian path is NP-complete. $\square$

Path kernel



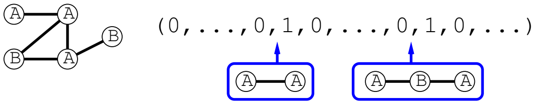
Definition

The path kernel is the subgraph kernel restricted to paths, i.e.,

$$K_{path}(G_1, G_2) = \sum_{H \in \mathcal{P}} \lambda_H \Phi_H(G_1) \Phi_H(G_2) \,,$$

where $\mathcal{P} \subset \mathcal{X}$ is the set of path graphs.

Path kernel



Definition

The path kernel is the subgraph kernel restricted to paths, i.e.,

$$K_{path}(G_1, G_2) = \sum_{H \in \mathcal{P}} \lambda_H \Phi_H(G_1) \Phi_H(G_2) \,,$$

where $\mathcal{P} \subset \mathcal{X}$ is the set of path graphs.

Proposition (Gärtner et al., 2003)

Computing the path kernel is NP-hard.

Summary

Expressiveness vs Complexity trade-off

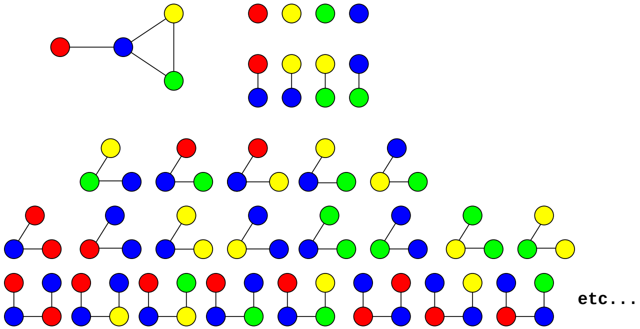
- It is intractable to compute complete graph kernels.
- It is intractable to compute the subgraph kernels.
- Restricting subgraphs to be linear does not help: it is also intractable to compute the path kernel.
- One approach to define polynomial time computable graph kernels is to have the feature space be made up of graphs homomorphic to subgraphs, e.g., to consider walks instead of paths.

Outline

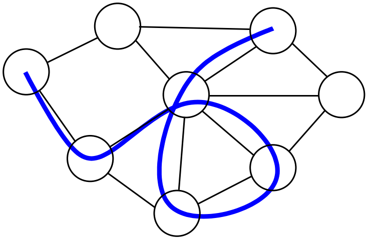
Walks

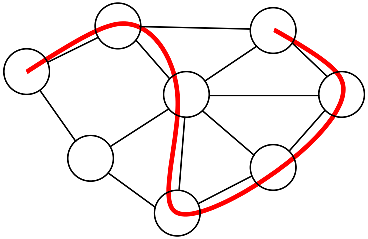
Definition

- A walk of a graph (V, E) is sequence of $v_1, \dots, v_n \in V$ such that $(v_i, v_{i+1}) \in E$ for $i = 1, \dots, n-1$.
- We note $\mathcal{W}_n(G)$ the set of walks with n vertices of the graph G, and $\mathcal{W}(G)$ the set of all walks.



Walks $\neq$ paths





Walk kernel

Definition

- Let $\mathcal{S}_n$ denote the set of all possible label sequences of walks of length n (including vertices and edges labels), and $\mathcal{S} = \cup_{n\geq 1}\mathcal{S}_n$.
- For any graph $\mathcal{X}$ let a weight $\lambda_G(w)$ be associated to each walk $w \in \mathcal{W}(G)$.
- Let the feature vector $\Phi(G) = (\Phi_s(G))_{s\in\mathcal{S}}$ be defined by:

$$\Phi_s(G) = \sum_{w\in\mathcal{W}(G)} \lambda_G(w)\mathbf{1}\left(s \text{ is the label sequence of } w\right) .$$

Walk kernel

Definition

- Let $\mathcal{S}_n$ denote the set of all possible label sequences of walks of length n (including vertices and edges labels), and $\mathcal{S} = \cup_{n \geq 1} \mathcal{S}_n$.
- For any graph $\mathcal{X}$ let a weight $\lambda_G(w)$ be associated to each walk $w \in \mathcal{W}(G)$.
- Let the feature vector $\Phi(G) = (\Phi_s(G))_{s \in \mathcal{S}}$ be defined by:

$$\Phi_s(G) = \sum_{w \in \mathcal{W}(G)} \lambda_G(w) \mathbf{1}\,(s \text{ is the label sequence of } w)\ .$$

- A walk kernel is a graph kernel defined by:

$$K_{walk}(G_1, G_2) = \sum_{s \in \mathcal{S}} \Phi_s(G_1) \Phi_s(G_2)\,.$$

Walk kernel examples

Examples

- The nth-order walk kernel is the walk kernel with $\lambda_G(w) = 1$ if the length of w is n, 0 otherwise. It compares two graphs through their common walks of length n.

Walk kernel examples

Examples

- The *n*th-order walk kernel is the walk kernel with $\lambda_G(w) = 1$ if the length of w is n, 0 otherwise. It compares two graphs through their common walks of length n.
- The random walk kernel is obtained with $\lambda_G(w) = P_G(w)$, where P_G is a Markov random walk on G. In that case we have:

$$K(G_1, G_2) = P(label(W_1) = label(W_2)) \,,$$

where W_1 and W_2 are two independant random walks on G_1 and G_2, respectively (Kashima et al., 2003).

Walk kernel examples

Examples

- The nth-order walk kernel is the walk kernel with $\lambda_G(w) = 1$ if the length of w is n, 0 otherwise. It compares two graphs through their common walks of length n.
- The random walk kernel is obtained with $\lambda_G(w) = P_G(w)$, where P_G is a Markov random walk on G. In that case we have:

$$K(G_1, G_2) = P(label(W_1) = label(W_2)),$$

  where W_1 and W_2 are two independant random walks on G_1 and G_2, respectively (Kashima et al., 2003).
- The geometric walk kernel is obtained (when it converges) with $\lambda_G(w) = \beta^{length(w)}$, for $\beta > 0$. In that case the feature space is of infinite dimension (Gärtner et al., 2003).

Computation of walk kernels

Proposition

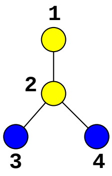
These three kernels (nth-order, random and geometric walk kernels) can be computed efficiently in polynomial time.

Product graph

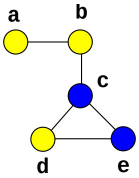
Definition

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs with labeled vertices. The product graph $G = G_1 \times G_2$ is the graph $G = (V, E)$ with:

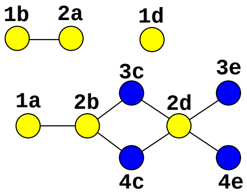
1. $V = \{(v_1, v_2) \in V_1 \times V_2 \ : \ v_1 \text{ and } v_2 \text{ have the same label}\}$,
2. $E = \{((v_1, v_2), (v_1', v_2')) \in V \times V \ : \ (v_1, v_1') \in E_1 \text{ and } (v_2, v_2') \in E_2\}$.



G1



G2



G1 x G2

Walk kernel and product graph

Lemma

There is a bijection between:

1. The pairs of walks $w_1 \in \mathcal{W}_n(G_1)$ and $w_2 \in \mathcal{W}_n(G_2)$ with the same label sequences,
2. The walks on the product graph $w \in \mathcal{W}_n(G_1 \times G_2)$.

Walk kernel and product graph

Lemma

There is a bijection between:

1. The pairs of walks $w_1 \in \mathcal{W}_n(G_1)$ and $w_2 \in \mathcal{W}_n(G_2)$ with the same label sequences,
2. The walks on the product graph $w \in \mathcal{W}_n(G_1 \times G_2)$.

Corollary

$$\begin{aligned}
K_{walk}(G_1, G_2) &= \sum_{s \in \mathcal{S}} \Phi_s(G_1)\Phi_s(G_2) \\
&= \sum_{(w_1,w_2)\in\mathcal{W}(G_1)\times\mathcal{W}(G_1)} \lambda_{G_1}(w_1)\lambda_{G_2}(w_2)\mathbf{1}(l(w_1) = l(w_2)) \\
&= \sum_{w\in\mathcal{W}(G_1\times G_2)} \lambda_{G_1\times G_2}(w)\,.
\end{aligned}$$

Computation of the nth-order walk kernel

- For the nth-order walk kernel we have $\lambda_{G_1 \times G_2}(w) = 1$ if the length of w is n, 0 otherwise.
- Therefore:
$$K_{nth-order}(G_1, G_2) = \sum_{w \in \mathcal{W}_n(G_1 \times G_2)} 1\,.$$
- Let A be the adjacency matrix of $G_1 \times G_2$. Then we get:
$$K_{nth-order}(G_1, G_2) = \sum_{i,j} [A^n]_{i,j} = \mathbf{1}^\top A^n \mathbf{1}\,.$$
- Computation in $O(n|G_1||G_2|d_1 d_2)$, where d_i is the maximum degree of G_i.

Computation of random and geometric walk kernels

- In both cases $\lambda_G(w)$ for a walk $w = v_1 \dots v_n$ can be decomposed as:

$$\lambda_G(v_1 \dots v_n) = \lambda^i(v_1) \prod_{i=2}^{n} \lambda^t(v_{i-1}, v_i) \,.$$

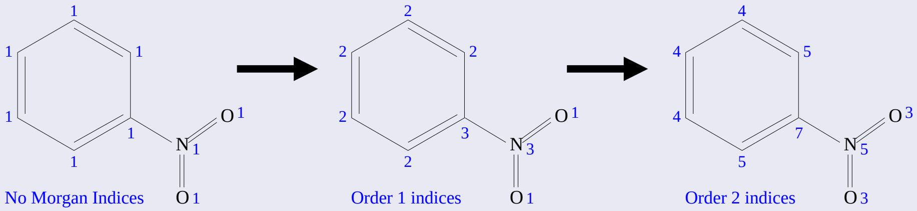
- Let Λ_i be the vector of $\lambda^i(v)$ and Λ_t be the matrix of $\lambda^t(v, v')$:

$$\begin{aligned} K_{walk}(G_1, G_2) &= \sum_{n=1}^{\infty} \sum_{w \in \mathcal{W}_n(G_1 \times G_2)} \lambda^i(v_1) \prod_{i=2}^{n} \lambda^t(v_{i-1}, v_i) \\ &= \sum_{n=0}^{\infty} \Lambda_i \Lambda_t^n \mathbf{1} \\ &= \Lambda_i \left(I - \Lambda_t\right)^{-1} \mathbf{1} \end{aligned}$$

- Computation in $O(|G_1|^3 |G_2|^3)$

Extensions 1: label enrichment

Atom relabebling with the Morgan index

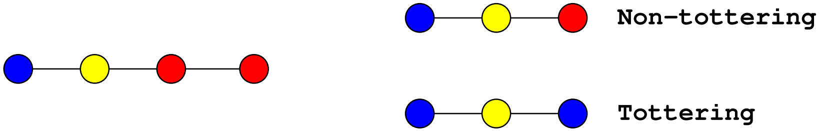


- Compromise between fingerprints and structural keys features.
- Other relabeling schemes are possible (graph coloring).
- Faster computation with more labels (less matches implies a smaller product graph).

Extension 2: Non-tottering walk kernel

Tottering walks

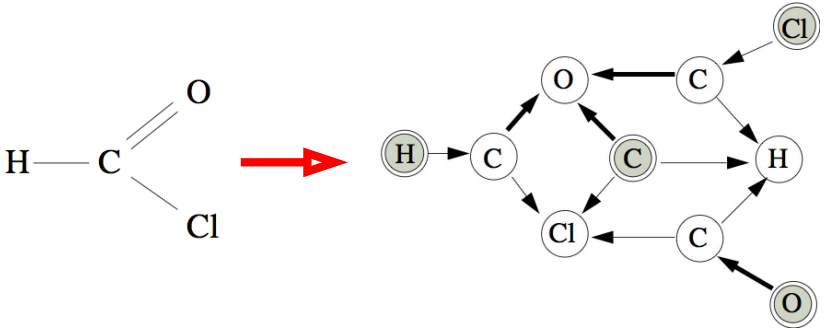
A tottering walk is a walk $w = v_1 \ldots v_n$ with $v_i = v_{i+2}$ for some i.



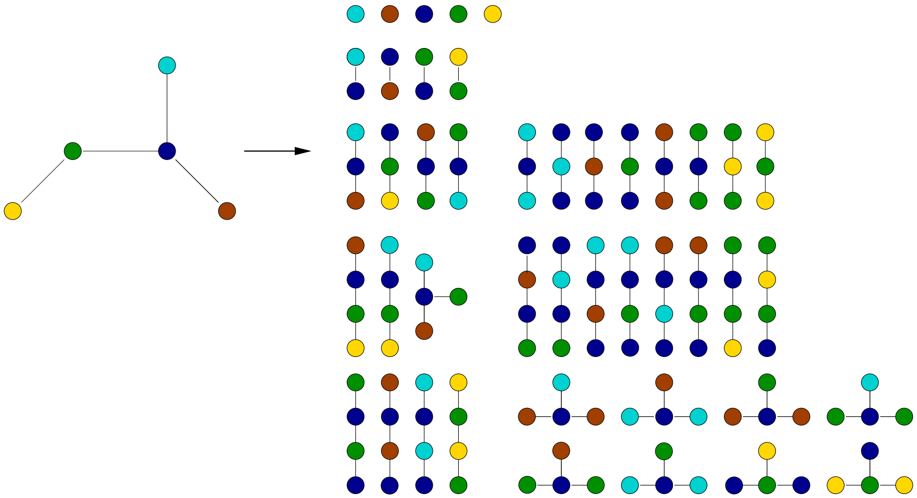
- Tottering walks seem irrelevant for many applications
- Focusing on non-tottering walks is a way to get closer to the path kernel (e.g., equivalent on trees).

Computation of the non-tottering walk kernel (Mahé et al., 2005)

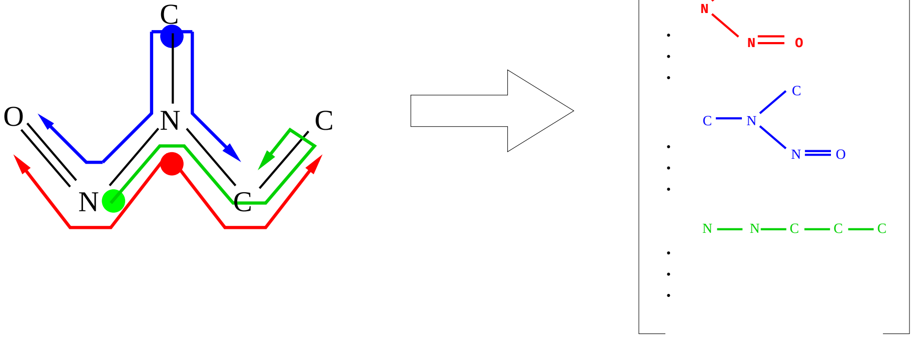
- Second-order Markov random walk to prevent tottering walks
- Written as a first-order Markov random walk on an augmented graph
- Normal walk kernel on the augmented graph (which is always a directed graph).



Extension 3: Subtree kernels



Example: Tree-like fragments of molecules



Computation of the subtree kernel

- Like the walk kernel, amounts to compute the (weighted) number of subtrees in the product graph.
- Recursion: if $\mathcal{T}(v,n)$ denotes the weighted number of subtrees of depth n rooted at the vertex v, then:

$$\mathcal{T}(v,n+1) = \sum_{R \subset \mathcal{N}(v)} \prod_{v' \in R} \lambda_t(v,v')\mathcal{T}(v',n)\,,$$

  where $\mathcal{N}(v)$ is the set of neighbors of v.
- Can be combined with the non-tottering graph transformation as preprocessing to obtain the non-tottering subtree kernel.

Outline

Application in chemoinformatics (Mahé et al., 2004)

MUTAG dataset

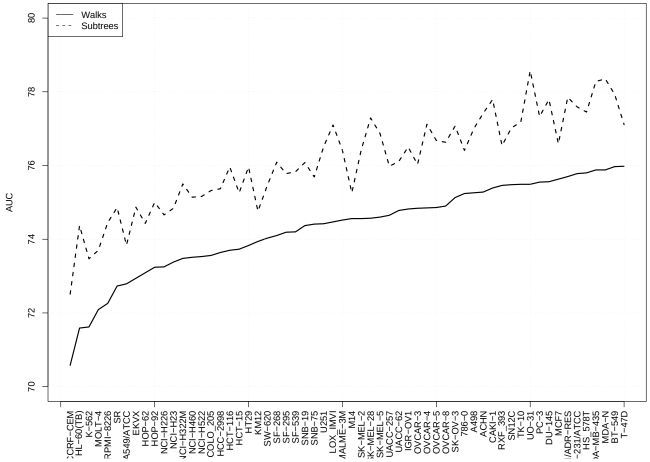
- aromatic/hetero-aromatic compounds
- high mutagenic activity /no mutagenic activity, assayed in *Salmonella typhimurium*.
- 188 compouunds: 125 + / 63 -

Results

10-fold cross-validation accuracy

Method	Accuracy
Progol1	81.4%
2D kernel	91.2%

2D Subtree vs walk kernels

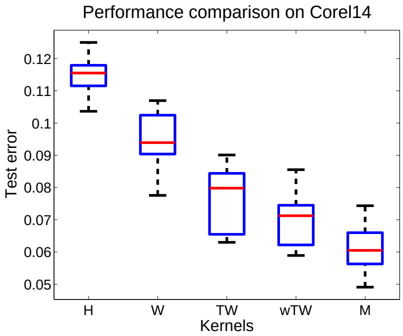


Screening of inhibitors for 60 cancer cell lines.

Image classification (Harchaoui and Bach, 2007)

COREL14 dataset

- 1400 natural images in 14 classes
- Compare kernel between histograms (H), walk kernel (W), subtree kernel (TW), weighted subtree kernel (wTW), and a combination (M).



Summary: graph kernels

What we saw

- Kernels do not allow to overcome the NP-hardness of subgraph patterns
- They allow to work with approximate subgraphs (walks, subtrees), in infinite dimension, thanks to the kernel trick
- However: using kernels makes it difficult to come back to patterns after the learning stage

Outline

Outline

Motivation

- In feature selection, we look for a linear function $f(\mathbf{x}) = \mathbf{x}^\top \beta$, where only a limited number of coefficients in β are non-zero.
- Motivations
  - Accuracy: by imposing a constraint on β, we increase the bias but decrease the variance. This should be helpful in particular in high dimension.
  - Interpretation: simpler to understand and communicate a sparse model.
  - Implementation: a device based on a few markers can be cheaper and faster.

Of course, this is particularly relevant if we believe that there exist good predictors which are sparse (prior knowledge).

Best subset selection

$$\Omega(\beta) = \|\beta\|_0 = \text{number of non-zero coefficients}$$

- In best subset selection, we must solve the problem:

$$\min R(f_\beta) \quad \text{s.t.} \quad \| \beta \|_0 \leq k$$

  for $k = 1, \ldots, p$.
- The state-of-the-art is branch-and-bound optimization, known as *leaps and bound* for least squares (Furnival and Wilson, 1974).
- This is usually a NP-hard problem, feasible for p as large as 30 or 40

Efficient feature selection

To work with more variables, we must use different methods. The state-of-the-art is split among

- Filter methods : the predictors are preprocessed and ranked from the most relevant to the less relevant. The subsets are then obtained from this list, starting from the top.
- Wrapper method: here the feature selection is iterative, and uses the ERM algorithm in the inner loop
- Embedded methods : here the feature selection is part of the ERM algorithm itself (see later the shrinkage estimators).

Filter methods

- Associate a score $S(i)$ to each feature i, then rank the features by decreasing score.
- Many scores / criteria can be used
  - Loss of the ERM trained on a single feature
  - Statistical tests (Fisher, T-test)
  - Other performance criteria of the ERM restricted to a single feature (AUC, ...)
  - Information theoretical criteria (mutual information...)

Filter methods

- Associate a score $S(i)$ to each feature i, then rank the features by decreasing score.
- Many scores / criteria can be used
  - Loss of the ERM trained on a single feature
  - Statistical tests (Fisher, T-test)
  - Other performance criteria of the ERM restricted to a single feature (AUC, ...)
  - Information theoretical criteria (mutual information...)

Pros

Simple, scalable, good empirical success

Filter methods

- Associate a score $S(i)$ to each feature i, then rank the features by decreasing score.
- Many scores / criteria can be used
    - Loss of the ERM trained on a single feature
    - Statistical tests (Fisher, T-test)
    - Other performance criteria of the ERM restricted to a single feature (AUC, ...)
    - Information theoretical criteria (mutual information...)

Pros

Simple, scalable, good empirical success

Cons

- Selection of redundant features
- Some variables useless alone can become useful together

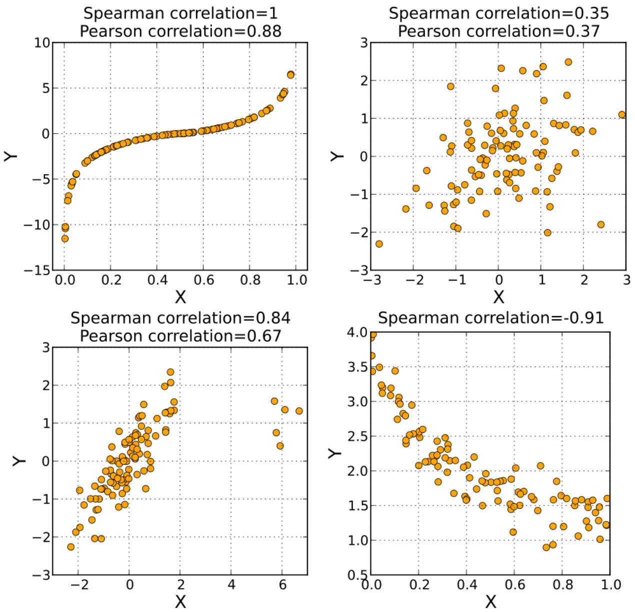
Measuring dependency: correlation coefficients

- Assume X and Y take continuous values
- $(X_1, Y_1), \ldots, (X_n, Y_n)$ the n expression values of both genes
- Pearson correlation:

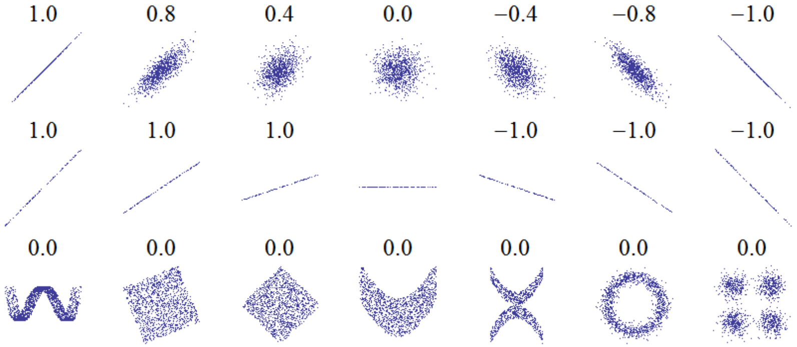
$$\rho = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i (X_i - \bar{X})^2}\sqrt{\sum_i (Y_i - \bar{Y})^2}}$$

- Spearman correlation: similar but replace X_i by its rank.

Illustration



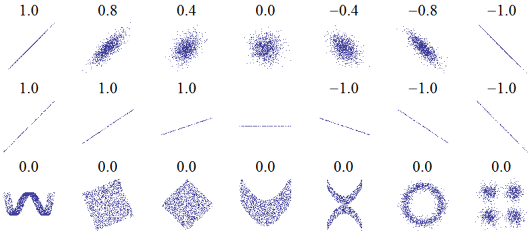
Limit of correlations



Mutual information

$$I(X;Y) = \int_Y \int_X p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) dxdy$$

- $I(X;Y) \geq 0$
- $I(X;Y) = 0$ if and only if X and Y are independent



Wrapper methods

The idea

- A greedy approach to

$$\min R(f_\beta) \quad \text{s.t.} \quad \| \beta \|_0 \leq k$$

- For a given set of seleted features, we know how to minimize $R(f)$
- We iteratively try to find a good set of features, by adding/removing features which contribute most to decrease the risk (using ERM as an internal loop)

Two flavors of wrapper methods

Forward stepwise selection

- Start from no features
- Sequentially add into the model the feature that most improves the fit

Two flavors of wrapper methods

Forward stepwise selection

- Start from no features
- Sequentially add into the model the feature that most improves the fit

Backward stepwise selection (if n¿p)

- Start from all features
- Sequentially removes from the model the feature that least degrades the fit

Two flavors of wrapper methods

Forward stepwise selection

- Start from no features
- Sequentially add into the model the feature that most improves the fit

Backward stepwise selection (if n¿p)

- Start from all features
- Sequentially removes from the model the feature that least degrades the fit

Other variants

Hybrid stepwise selection strategies that consider both forward and backward moves at each stage, and make the "best" move

Outline

The idea

- The following problem is NP-hard:

$$\min R(f_\beta) \quad \text{s.t.} \quad \| \beta \|_0 \leq k$$

- As a proxy we can consider the more general problem:

$$\min R(f_\beta) \quad \text{s.t.} \quad \Omega(\beta) \leq \gamma$$

 where $\Omega(\beta)$ is a penalty function that leads to sparse solutions and to computationally efficient algorithms.

LASSO regression (Tibshirani, 1996) Basis Pursuit (Chen et al., 1998)

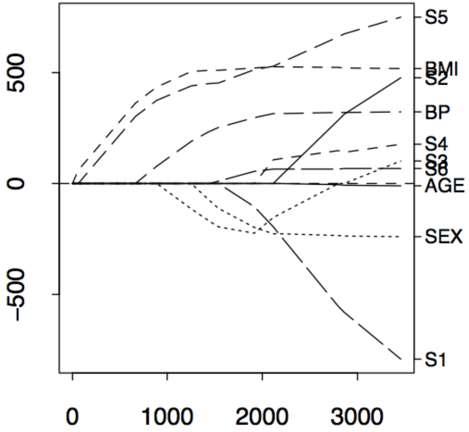
$$\Omega(\beta) = \| \beta \|_1 = \sum_{i=1}^{p} | \beta_i |$$

- LASSO or BP:

$$\min_{\beta} R(f_\beta) = \sum_{i=1}^{n} \left(f_\beta \left(\mathbf{x}_i \right) - \mathbf{y}_i \right)^2 + \lambda \sum_{i=1}^{p} | \beta_i | \qquad (3)$$

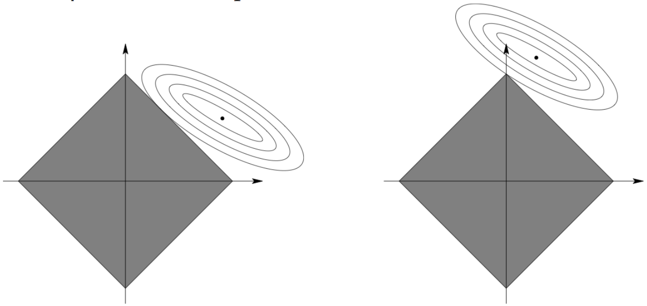
- No explicit solution, but this is just a quadratic program.
- LARS (Efron et al., 2004) provides a fast algorithm to compute the solution for all λ's simultaneously (regularization path)

LASSO regression example



Why LASSO leads to sparse solutions

Geometric interpretation with $p = 2$



Generalization: Atomic Norm [Chandrasekaran et al., 2012]

Definition

Given a set of atoms $\mathcal{A}$, the associated atomic norm is

$$\|x\|_{\mathcal{A}} = \inf\{t > 0 \mid x \in t\,\mathrm{conv}(\mathcal{A})\}.$$

NB: This is really a norm if $\mathcal{A}$ is centrally symmetric and spans $\mathbb{R}^p$

Primal and dual form of the norm

$$\begin{aligned} \|x\|_{\mathcal{A}} &= \inf\left\{ \sum_{a\in\mathcal{A}} c_a \;\middle|\; x = \sum_{a\in\mathcal{A}} c_a\, a, \quad c_a > 0,\ \forall a \in \mathcal{A} \right\} \\ \|x\|_{\mathcal{A}}^{*} &= \sup_{a\in\mathcal{A}} \langle a, x\rangle \end{aligned}$$

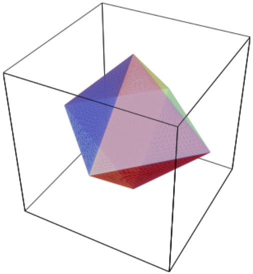
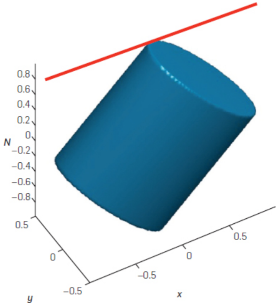
Examples

- Vector ℓ_1-norm: $x \in \mathbb{R}^p \mapsto \|x\|_1$

$$\mathcal{A} = \{\pm e_k \mid 1 \leq k \leq p\}$$

- Matrix trace norm: $Z \in \mathbb{R}^{m_1 \times m_2} \mapsto \|Z\|_*$ (sum of singular value)

$$\mathcal{A} = \left\{ab^\top \ : \ a \in \mathbb{R}^{m_1}, b \in \mathbb{R}^{m_2}, \| a \|_2 = \| b \|_2 = 1\right\}$$

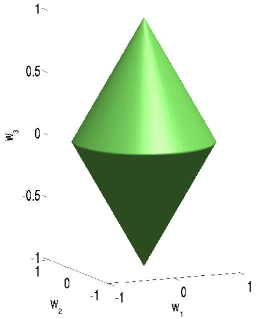
Group lasso [Yuan and Lin, 2006]

For $x \in \mathbb{R}^p$ and $\mathcal{G} = \{g_1, \dots, g_G\}$ a partition of $[1, p]$:

$$\| x \|_{1,2} = \sum_{g \in \mathcal{G}} \| x_g \|_2$$

is the atomic norm associated to the set of atoms

$$\mathcal{A}_{\mathcal{G}} = \bigcup_{g \in \mathcal{G}} \{u \in \mathbb{R}^p \ : \ \mathrm{supp}(u) = g, \| u \|_2 = 1\}$$



$$\begin{aligned}\mathcal{G} &= \{\{1, 2\}, \{3\}\} \\ \| x \|_{1,2} &= \|(x_1, x_2)^\top\|_2 + \|x_3\|_2 \\ &= \sqrt{x_1^2 + x_2^2} + \sqrt{x_3^2}\end{aligned}$$

Group lasso with overlaps

How to generalize the group lasso when the groups overlap?

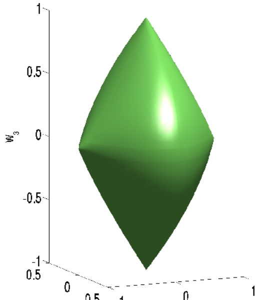
- Set features to zero by groups [Jenatton et al., 2011]

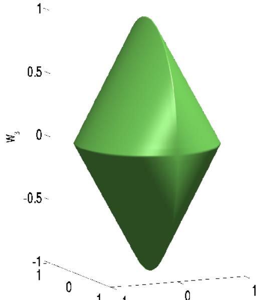
$$\| x \|_{1,2} = \sum_{g \in \mathcal{G}} \| x_g \|_2$$

- Select support as a union of groups [Jacob et al., 2009]

$$\| x \|_{\mathcal{A}_{\mathcal{G}}},$$

see also MKL [Bach et al., 2004]



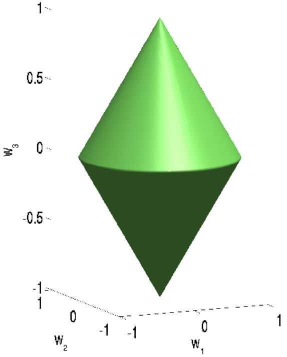


$$\mathcal{G} = \{\{1,2\}, \{2,3\}\}$$

Extension to other loss functions

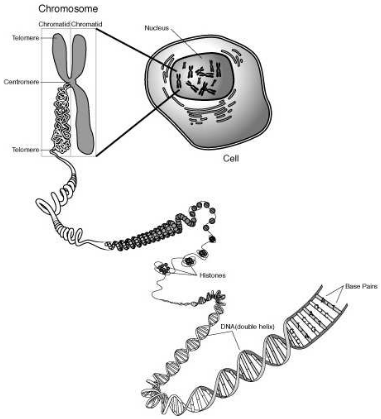
Of course we can learn sparse or group-sparse linear models with any different (smoothly convex) loss function:

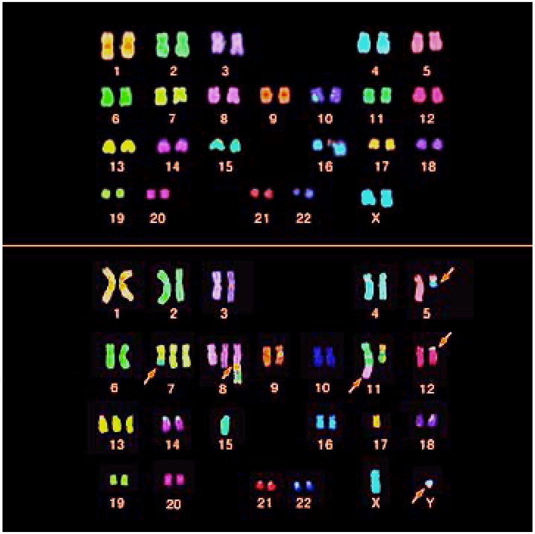
$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f_\beta\left(\mathbf{x}_i\right), \mathbf{y}_i\right) + \lambda \|\beta\|_1 \text{ or } \|\beta\|_{1,2}$$



Outline

Chromosomic aberrations in cancer

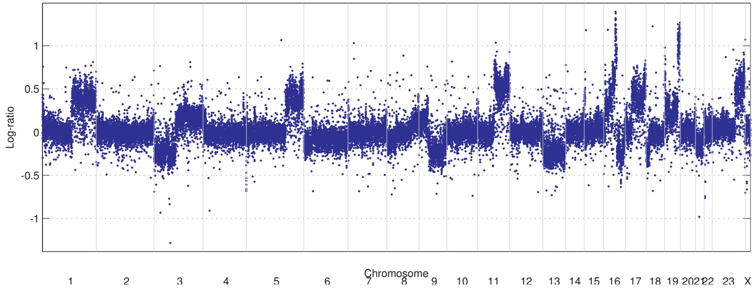




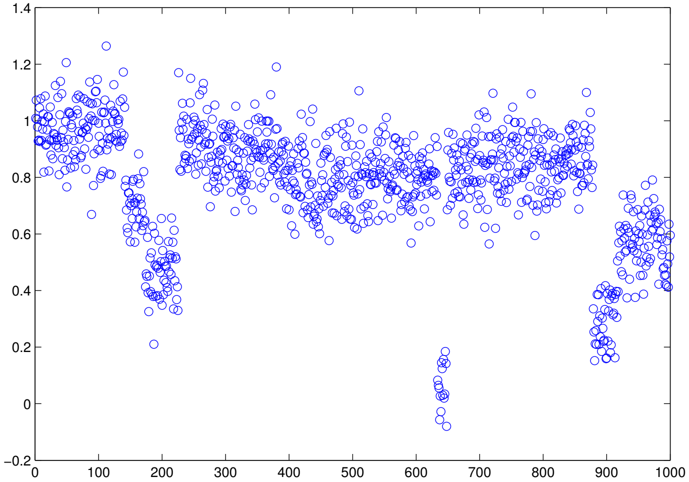
Comparative Genomic Hybridization (CGH)

Motivation

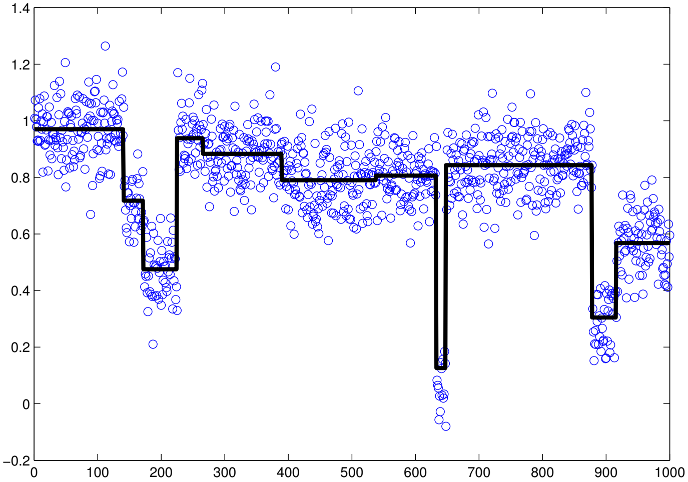
- Comparative genomic hybridization (CGH) data measure the DNA copy number along the genome
- Very useful, in particular in cancer research to observe systematically variants in DNA content



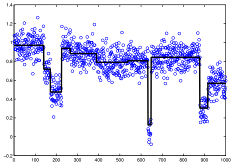
Where are the breakpoints?



Where are the breakpoints?



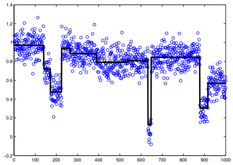
Optimal breakpoint detection



- Let $Y \in \mathbb{R}^p$ the signal. We search a smooth profile $\beta \in \mathbb{R}^p$ with at most k change-points by solving

$$\min_{\beta \in \mathbb{R}^p} \| Y - \beta \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} \mathbf{1}(\beta_{i+1} \neq \beta_i) \leq k$$

Optimal breakpoint detection

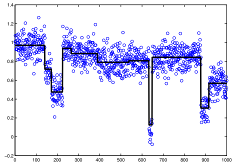


- Let $Y \in \mathbb{R}^p$ the signal. We search a smooth profile $\beta \in \mathbb{R}^p$ with at most k change-points by solving

$$\min_{\beta \in \mathbb{R}^p} \| Y - \beta \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} \mathbf{1}(\beta_{i+1} \neq \beta_i) \leq k$$

- This is an optimization problem over the $\binom{p}{k}$ partitions...

Optimal breakpoint detection

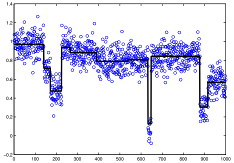


- Let $Y \in \mathbb{R}^p$ the signal. We search a smooth profile $\beta \in \mathbb{R}^p$ with at most k change-points by solving

$$\min_{\beta \in \mathbb{R}^p} \| Y - \beta \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} \mathbf{1}(\beta_{i+1} \neq \beta_i) \leq k$$

- This is an optimization problem over the $\binom{p}{k}$ partitions...
- Dynamic programming finds the solution in $O(p^2 k)$ in time and $O(p^2)$ in memory

Optimal breakpoint detection



- Let $Y \in \mathbb{R}^p$ the signal. We search a smooth profile $\beta \in \mathbb{R}^p$ with at most k change-points by solving

$$\min_{\beta \in \mathbb{R}^p} \| Y - \beta \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} \mathbf{1}\left(\beta_{i+1} \neq \beta_i\right) \leq k$$

- This is an optimization problem over the $\binom{p}{k}$ partitions...
- Dynamic programming finds the solution in $O(p^2 k)$ in time and $O(p^2)$ in memory
- But: does not scale to $p = 10^6 \sim 10^9$...

Promoting piecewise constant profiles

$$\Omega(\beta) = \|\beta\|_{TV} = \sum_{i=1}^{p-1} |\beta_{i+1} - \beta_i|$$

The total variation / variable fusion penalty

If $R(\beta)$ is convex and "smooth", the solution of

$$\min_{\beta \in \mathbb{R}^p} R(\beta) + \lambda \sum_{i=1}^{p-1} |\beta_{i+1} - \beta_i|$$

is usually piecewise constant (Rudin et al., 1992; Land and Friedman, 1996).

Proof:

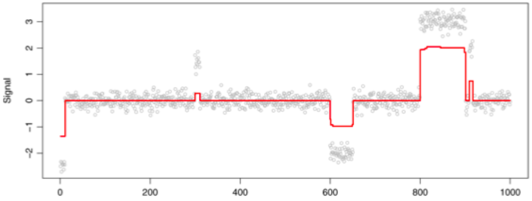
- Change of variable $u_i = \beta_{i+1} - \beta_i$, $u_0 = \beta_1$
- We obtain a Lasso problem in $u \in \mathbb{R}^{p-1}$
- u sparse means β piecewise constant

TV signal approximator

$$\min_{\beta \in \mathbb{R}^p} \| Y - \beta \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} | \beta_{i+1} - \beta_i | \leq \mu$$

Adding additional constraints does not change the change-points:

- $\sum_{i=1}^{p} | \beta_i | \leq \nu$ (Tibshirani et al., 2005; Tibshirani and Wang, 2008)
- $\sum_{i=1}^{p} \beta_i^2 \leq \nu$ (Mairal et al. 2010)



Solving TV signal approximator

$$\min_{\beta \in \mathbb{R}^p} \| Y - \beta \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} | \beta_{i+1} - \beta_i | \leq \mu$$

- QP with sparse linear constraints in $O(p^2)$ -¿ 135 min for $p = 10^5$ (Tibshirani and Wang, 2008)
- Coordinate descent-like method $O(p)$? -¿ 3s s for $p = 10^5$ (Friedman et al., 2007)
- For all μ with the LARS in $O(pK)$ (Harchaoui and Levy-Leduc, 2008)
- For all μ in $O(p \ln p)$ (Hoefling, 2009)
- For the first K change-points in $O(p \ln K)$ (Bleakley and V., 2010)

TV signal approximator as dichotomic segmentation

Algorithm 1 Greedy dichotomic segmentation

```
Require: $k$ number of intervals, $\gamma(I)$ gain function to split an interval $I$ into $I_L(I), I_R(I)$
1: $I_0$ represents the interval $[1, n]$
2: $\mathcal{P} = \{I_0\}$
3: for $i = 1$ to $k$ do
4:     $I^* \leftarrow \arg\max_{I \in \mathcal{P}} \gamma(I^*)$
5:     $\mathcal{P} \leftarrow \mathcal{P} \backslash \{I^*\}$
6:     $\mathcal{P} \leftarrow \mathcal{P} \cup \{I_L(I^*), I_R(I^*)\}$
7: end for
8: return $\mathcal{P}$
```

Theorem (V. and Bleakley, 2010; see also Hoefling, 2009)

TV signal approximator performs "greedy" dichotomic segmentation

TV signal approximator as dichotomic segmentation

Algorithm 1 Greedy dichotomic segmentation

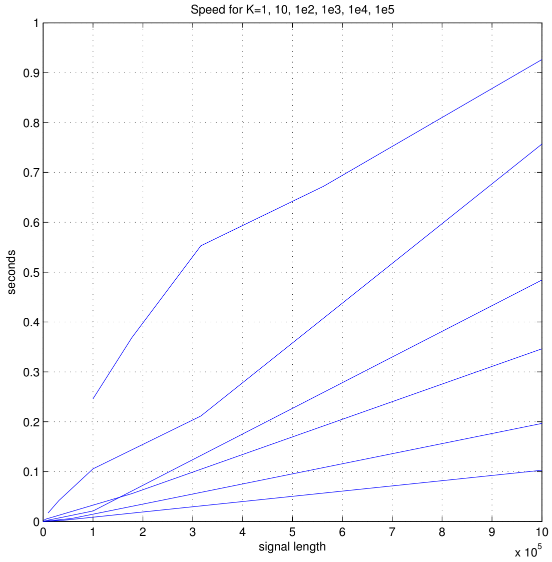
```
Require: $k$ number of intervals, $\gamma(I)$ gain function to split an interval $I$ into $I_L(I), I_R(I)$
1: $I_0$ represents the interval $[1, n]$
2: $\mathcal{P} = \{I_0\}$
3: for $i = 1$ to $k$ do
4:     $I^* \leftarrow \arg\max_{I \in \mathcal{P}} \gamma(I^*)$
5:     $\mathcal{P} \leftarrow \mathcal{P} \backslash \{I^*\}$
6:     $\mathcal{P} \leftarrow \mathcal{P} \cup \{I_L(I^*), I_R(I^*)\}$
7: end for
8: return $\mathcal{P}$
```

Theorem (V. and Bleakley, 2010; see also Hoefling, 2009)

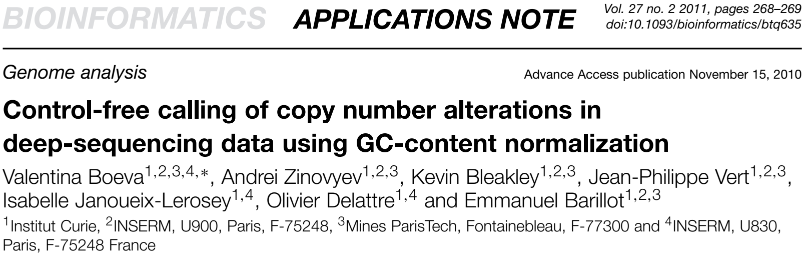
TV signal approximator performs "greedy" dichotomic segmentation

Apparently greedy algorithm finds the global optimum!

Speed trial : 2 s. for $K = 100$, $p = 10^7$



Applications


BIOINFORMATICS APPLICATIONS NOTE

Vol. 27 no. 2 2011, pages 268–269
doi:10.1093/bioinformatics/btq635

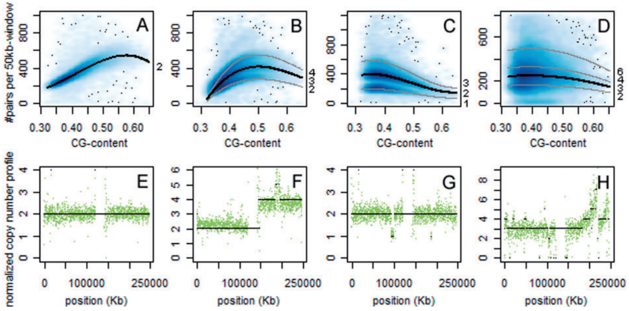
Genome analysis

Advance Access publication November 15, 2010

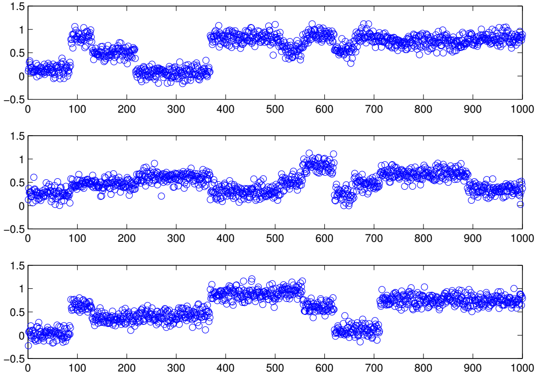
Control-free calling of copy number alterations in deep-sequencing data using GC-content normalization

Valentina Boeva[1,2,3,4,*], Andrei Zinovyev[1,2,3], Kevin Bleakley[1,2,3], Jean-Philippe Vert[1,2,3], Isabelle Janoueix-Lerosey[1,4], Olivier Delattre[1,4] and Emmanuel Barillot[1,2,3]

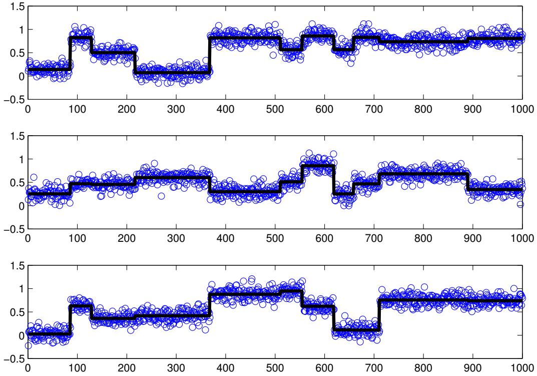
[1]Institut Curie, [2]INSERM, U900, Paris, F-75248, [3]Mines ParisTech, Fontainebleau, F-77300 and [4]INSERM, U830, Paris, F-75248 France



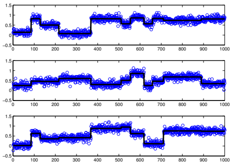
Extension 1: finding multiple change points shared by several profiles



Extension 1: finding multiple change points shared by several profiles



"Optimal" segmentation by dynamic programming



- Define the "optimal" piecewise constant approximation $\hat{U} \in \mathbb{R}^{p \times n}$ of Y as the solution of

$$\min_{U \in \mathbb{R}^{p \times n}} \| Y - U \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} \mathbf{1}\left(U_{i+1,\bullet} \neq U_{i,\bullet}\right) \leq k$$

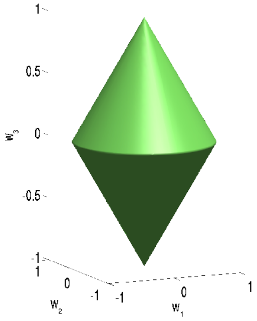
- DP finds the solution in $O(p^2 kn)$ in time and $O(p^2)$ in memory
- But: does not scale to $p = 10^6 \sim 10^9$...

Selecting pre-defined groups of variables

Group lasso (Yuan & Lin, 2006)

If groups of covariates are likely to be selected together, the ℓ_1/ℓ_2-norm induces sparse solutions *at the group level*:

$$\Omega_{group}(w) = \sum_g \|w_g\|_2$$



$$\begin{aligned}\Omega(w_1, w_2, w_3) &= \|(w_1, w_2)\|_2 + \|w_3\|_2 \\ &= \sqrt{w_1^2 + w_2^2} + \sqrt{w_3^2}\end{aligned}$$

GFLseg (Bleakley and V., 2011)

Replace

$$\min_{U \in \mathbb{R}^{p \times n}} \| Y - U \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} \mathbf{1}\left(U_{i+1,\bullet} \neq U_{i,\bullet}\right) \leq k$$

by

$$\min_{U \in \mathbb{R}^{p \times n}} \| Y - U \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} w_i \| U_{i+1,\bullet} - U_{i,\bullet} \| \leq \mu$$

GFLseg = Group Fused Lasso segmentation

GFLseg (Bleakley and V., 2011)

Replace

$$\min_{U\in\mathbb{R}^{p\times n}} \| Y - U \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} \mathbf{1}\left(U_{i+1,\bullet} \neq U_{i,\bullet}\right) \leq k$$

by

$$\min_{U\in\mathbb{R}^{p\times n}} \| Y - U \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} w_i \| U_{i+1,\bullet} - U_{i,\bullet} \| \leq \mu$$

GFLseg = Group Fused Lasso segmentation

Questions

- Practice: can we solve it efficiently?
- Theory: does it recover the correct segmentation?

TV approximator implementation

$$\min_{U \in \mathbb{R}^{p \times n}} \| Y - U \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} w_i \| U_{i+1,\bullet} - U_{i,\bullet} \| \leq \mu$$

Theorem

The TV approximator can be solved efficiently:

- approximately with the group LARS in $O(npk)$ in time and $O(np)$ in memory
- exactly with a block coordinate descent + active set method in $O(np)$ in memory

Speed trial

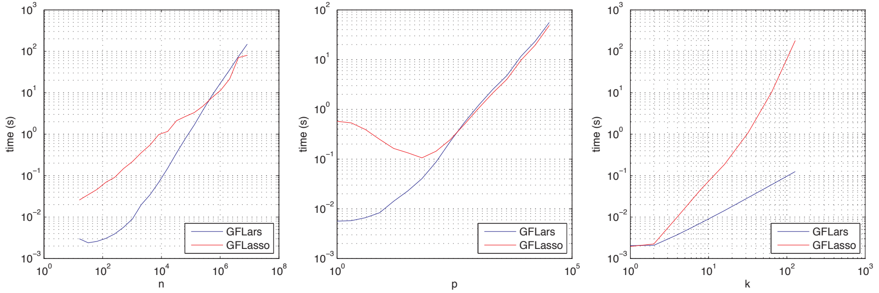
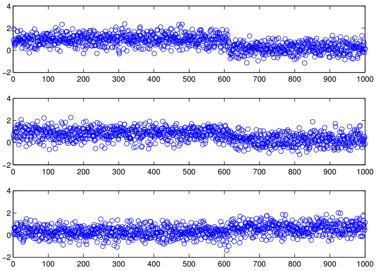


Figure 2: **Speed trials for group fused LARS (top row) and Lasso (bottom row).** *Left column:* varying n, with fixed $p = 10$ and $k = 10$; *center column:* varying p, with fixed $n = 1000$ and $k = 10$; *right column:* varying k, with fixed $n = 1000$ and $p = 10$. Figure axes are log-log. Results are averaged over 100 trials.

Consistency

Suppose a single change-point:

- at position $u = \alpha p$
- with increments $(\beta_i)_{i=1,\ldots,n}$ s.t. $\bar{\beta}^2 = \lim_{k\to\infty} \frac{1}{n} \sum_{i=1}^{n} \beta_i^2$
- corrupted by i.i.d. Gaussian noise of variance σ^2



Does the TV approximator correctly estimate the first change-point as p increases?

Consistency of the weighted TV approximator

$$\min_{U \in \mathbb{R}^{p \times n}} \| Y - U \|^2 \quad \text{such that} \quad \sum_{i=1}^{p-1} w_i \| U_{i+1,\bullet} - U_{i,\bullet} \| \leq \mu$$

Theorem

The weighted TV approximator with weights

$$\forall i \in [1, p-1] \,, \quad w_i = \sqrt{\frac{i(p-i)}{p}}$$

correctly finds the first change-point with probability tending to 1 as $n \to +\infty$.

- we see the benefit of increasing n
- we see the benefit of adding weights to the TV penalty

Consistency for a single change-point

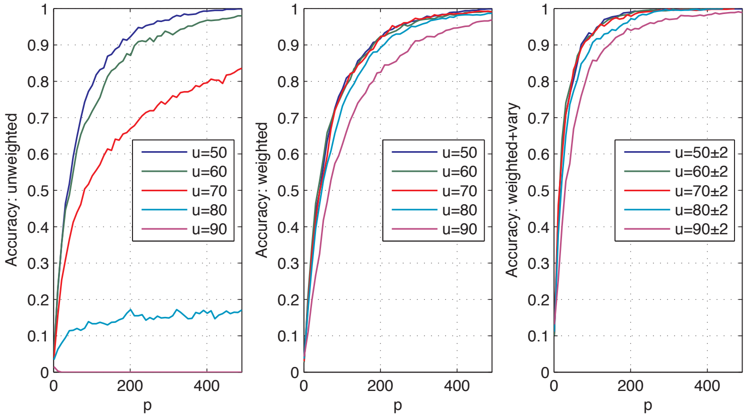


Figure 3: **Single change-point accuracy for the group fused Lasso.** Accuracy as a function of the number of profiles p when the change-point is placed in a variety of positions $u = 50$ to $u = 90$ (left and centre plots, resp. unweighted and weighted group fused Lasso), or: $u = 50 \pm 2$ to $u = 90 \pm 2$ (right plot, weighted with varying change-point location), for a signal of length 100.

Estimation of several change-points

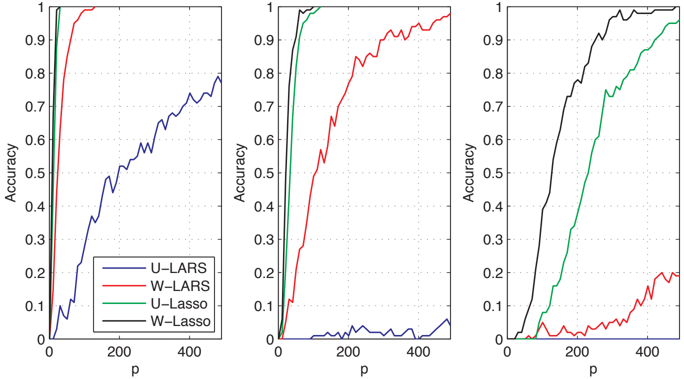
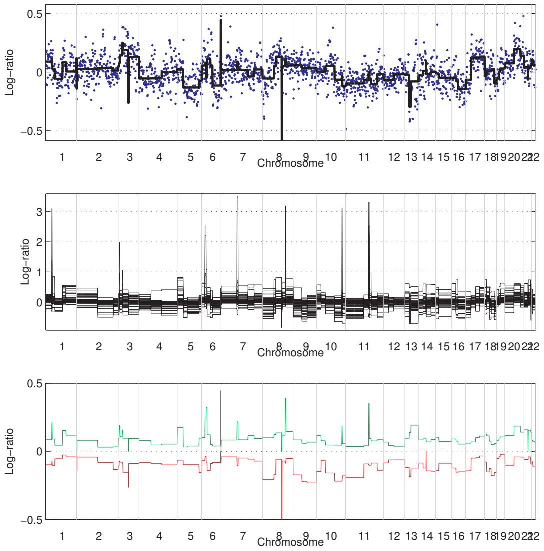
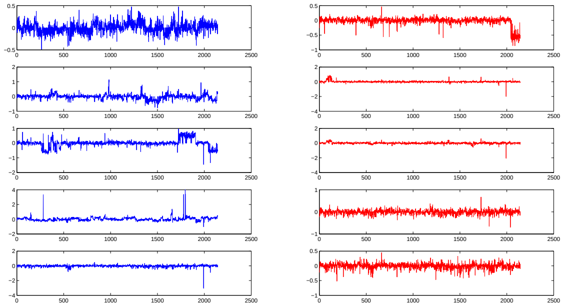


Figure 4: **Multiple change-point accuracy.** Accuracy as a function of the number of profiles p when change-points are placed at the nine positions $\{10, 20, \dots, 90\}$ and the variance σ^2 of the centered Gaussian noise is either 0.05 (left), 0.2 (center) and 1 (right). The profile length is 100.

Application: detection of frequent abnormalities



Extension 2: Supervised classification of genomic profiles

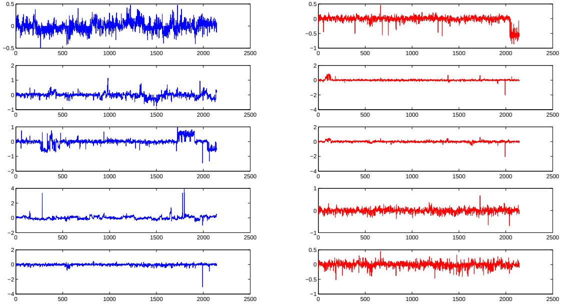


- $x_1, \ldots, x_n \in \mathbb{R}^p$ the n profiles of length p
- $y_1, \ldots, y_n \in [-1, 1]$ the labels
- We want to learn a function $f : \mathbb{R}^p \to [-1, 1]$

Prior knowledge

We expect β to be

- sparse : not all positions should be discriminative, and we want to identify the predictive region (presence of oncogenes or tumor suppressor genes?)
- piecewise constant : within a selected region, all probes should contribute equally



Fused lasso for supervised classification (Rapaport et al., 2008)

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \ell\left(y_i, \beta^\top x_i\right) + \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{i=1}^{p-1} |\beta_{i+1} - \beta_i| \, .$$

where ℓ is, e.g., the hinge loss $\ell(y, t) = max(1 - yt, 0)$.

Fused lasso for supervised classification (Rapaport et al., 2008)

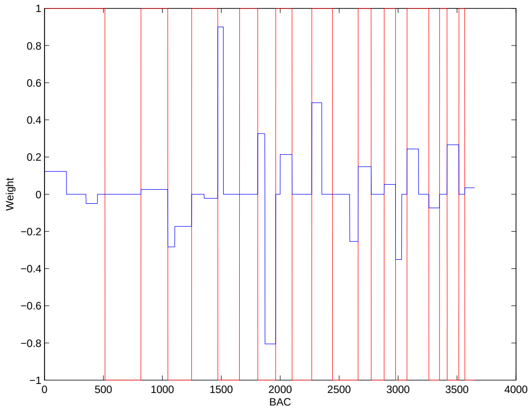
$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \ell\left(y_i, \beta^\top x_i\right) + \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{i=1}^{p-1} |\beta_{i+1} - \beta_i| \,.$$

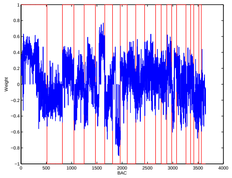
where ℓ is, e.g., the hinge loss $\ell(y,t) = max(1 - yt, 0)$.

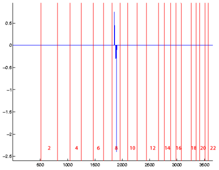
Implementation

- When ℓ is the hinge loss (fused SVM), this is a linear program -¿ up to $p = 10^3 \sim 10^4$
- When ℓ is convex and smooth (logistic, quadratic), efficient implementation with proximal methods -¿ up to $p = 10^8 \sim 10^9$

Example: predicting metastasis in melanoma





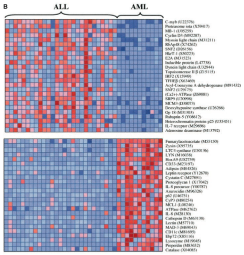


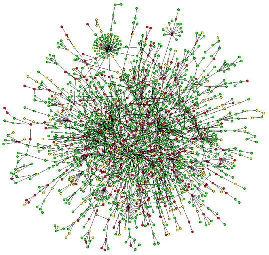
Outline

Gene networks and expression data

Motivation

- Basic biological functions usually involve the coordinated action of several proteins:
  - Formation of protein complexes
  - Activation of metabolic, signalling or regulatory pathways
- Many pathways and protein-protein interactions are already known
- Hypothesis: the weights of the classifier should be "coherent" with respect to this prior knowledge





Graph-based penalty

$$\min_{\beta} R(\beta) + \lambda \Omega_G(\beta)$$

Hypothesis

We would like to design penalties $\Omega_G(\beta)$ to promote one of the following hypothesis:

- Hypothesis 1: genes near each other on the graph should have similar weights (but we do not try to select only a few genes), i.e., the classifier should be smooth on the graph
- Hypothesis 2: genes selected in the signature should be connected to each other, or be in a few known functional groups, without necessarily having similar weights.

Graph based penalty with kernels

Prior hypothesis

Genes near each other on the graph should have similar weigths.

Graph based penalty with kernels

Prior hypothesis

Genes near each other on the graph should have similar weigths.

Network kernel (Rapaport et al., 2007)

$$\Omega_{spectral}(\beta) = \sum_{i \sim j} (\beta_i - \beta_j)^2 \,,$$

$$\min_{\beta \in \mathbb{R}^p} R(\beta) + \lambda \sum_{i \sim j} (\beta_i - \beta_j)^2 \,.$$

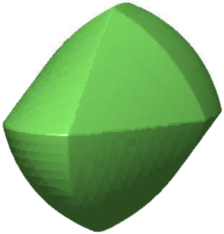
Other penalties without kernels

- Gene selection + Piecewise constant on the graph

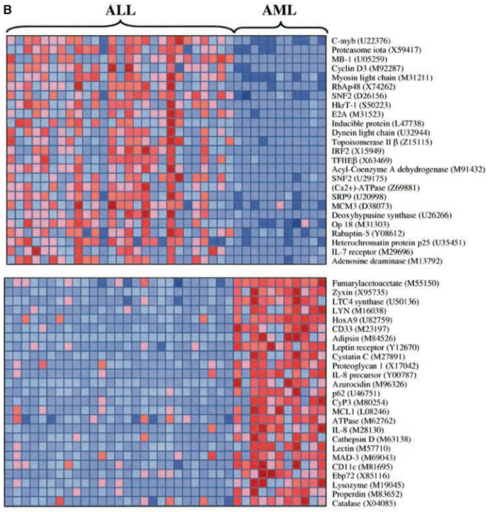
$$\Omega(\beta) = \sum_{i \sim j} |\,\beta_i - \beta_j\,| + \sum_{i=1}^{p} |\,\beta_i\,|$$

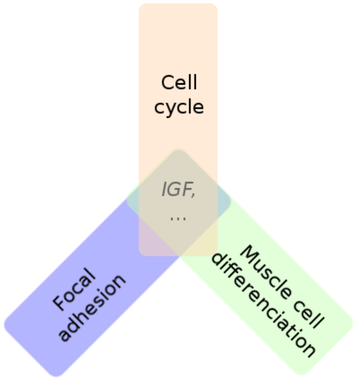
- Gene selection + smooth on the graph

$$\Omega(\beta) = \sum_{i \sim j} (\beta_i - \beta_j)^2 + \sum_{i=1}^{p} |\,\beta_i\,|$$



How to select jointly genes belonging to predefined pathways?



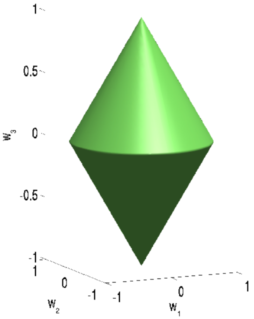


Selecting pre-defined groups of variables

Group lasso (Yuan & Lin, 2006)

If groups of covariates are likely to be selected together, the ℓ_1/ℓ_2-norm induces sparse solutions *at the group level*:

$$\Omega_{group}(w) = \sum_g \|w_g\|_2$$

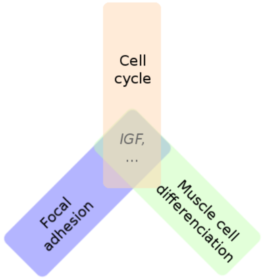


$$\Omega(w_1, w_2, w_3) = \|(w_1, w_2)\|_2 + \|w_3\|_2$$

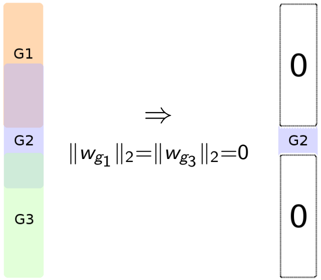
What if a gene belongs to several groups?

Issue of using the group-lasso

- $\Omega_{group}(w) = \sum_g \|w_g\|_2$ sets groups to 0.
- One variable is selected $\Leftrightarrow$ all the groups to which it belongs are selected.



IGF selection $\Rightarrow$ selection of unwanted groups



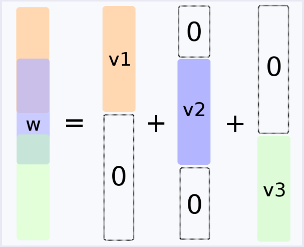
Removal of *any* group containing a gene $\Rightarrow$ the weight of the gene is 0.

Latent group lasso (Jacob et al., 2009)

An idea

Introduce latent variables v_g:

$$\begin{cases} \min\limits_{w,v} L(w) + \lambda \sum\limits_{g \in \mathcal{G}} \|v_g\|_2 \\ w = \sum_{g \in \mathcal{G}} v_g \\ \operatorname{supp}(v_g) \subseteq g. \end{cases}$$



Properties

- Resulting support is a *union* of groups in $\mathcal{G}$.
- Possible to select one variable without selecting all the groups containing it.
- Equivalent to group lasso when there is no overlap

A new norm

Overlap norm

$$\begin{cases} \min\limits_{w,v} L(w) + \lambda \sum\limits_{g \in \mathcal{G}} \|v_g\|_2 \\ w = \sum_{g \in \mathcal{G}} v_g \\ \operatorname{supp}(v_g) \subseteq g. \end{cases} = \min_{w} L(w) + \lambda \Omega_{overlap}(w)$$

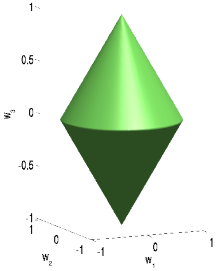
with

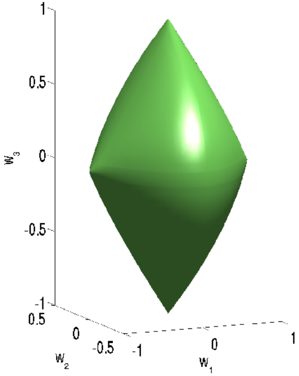
$$\Omega_{overlap}(w) \triangleq \begin{cases} \min\limits_{v} \sum\limits_{g \in \mathcal{G}} \|v_g\|_2 \\ w = \sum_{g \in \mathcal{G}} v_g \\ \operatorname{supp}(v_g) \subseteq g. \end{cases} \qquad (*)$$

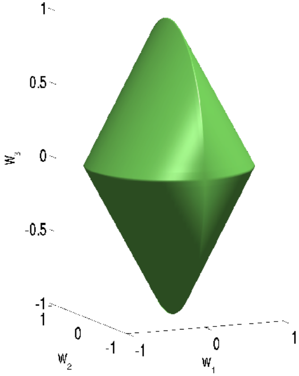
Property

- $\Omega_{overlap}(w)$ is a norm of w.
- $\Omega_{overlap}(.)$ associates to w a specific (not necessarily unique) decomposition $(v_g)_{g \in \mathcal{G}}$ which is the argmin of $(*)$.

Overlap and group unity balls







Balls for $\Omega^{\mathcal{G}}_{\text{group}}(\cdot)$ (middle) and $\Omega^{\mathcal{G}}_{\text{overlap}}(\cdot)$ (right) for the groups $\mathcal{G} = \{\{1,2\},\{2,3\}\}$ where w_2 is represented as the vertical coordinate. Left: group-lasso ($\mathcal{G} = \{\{1,2\},\{3\}\}$), for comparison.

Theoretical results

Consistency in group support (Jacob et al., 2009)

- Let $\bar{w}$ be the true parameter vector.
- Assume that there exists a unique decomposition $\bar{v}_g$ such that $\bar{w} = \sum_g \bar{v}_g$ and $\Omega^{\mathcal{G}}_{\text{overlap}}(\bar{w}) = \sum \|\bar{v}_g\|_2$.
- Consider the regularized empirical risk minimization problem $L(w) + \lambda \Omega^{\mathcal{G}}_{\text{overlap}}(w)$.

Theoretical results

Consistency in group support (Jacob et al., 2009)

- Let $\bar{w}$ be the true parameter vector.
- Assume that there exists a unique decomposition $\bar{v}_g$ such that $\bar{w} = \sum_g \bar{v}_g$ and $\Omega^{\mathcal{G}}_{\text{overlap}}(\bar{w}) = \sum \|\bar{v}_g\|_2$.
- Consider the regularized empirical risk minimization problem $L(w) + \lambda \Omega^{\mathcal{G}}_{\text{overlap}}(w)$.

Then

- under appropriate mutual incoherence conditions on X,
- as $n \to \infty$,
- with very high probability,

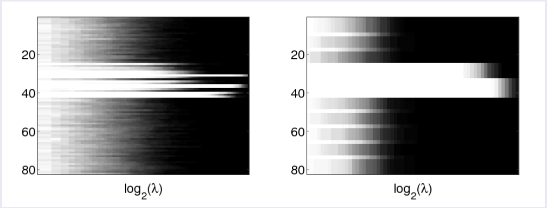
the optimal solution $\hat{w}$ admits a unique decomposition $(\hat{v}_g)_{g \in \mathcal{G}}$ such that

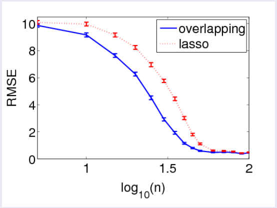
$$\{g \in \mathcal{G} | \hat{v}_g \neq 0\} = \{g \in \mathcal{G} | \bar{v}_g \neq 0\}.$$

Experiments

Synthetic data: overlapping groups

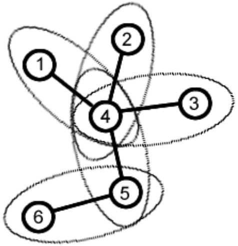
- 10 groups of 10 variables with 2 variables of overlap between two successive groups :$\{1, \ldots, 10\}, \{9, \ldots, 18\}, \ldots, \{73, \ldots, 82\}$.
- Support: union of $4th$ and $5th$ groups.
- Learn from 100 training points.





Frequency of selection of each variable with the lasso (left) and $\Omega^{\mathcal{G}}_{\text{overlap}}(.)$ (middle), comparison of the RMSE of both methods (right).

Graph lasso



Two solutions

$$\Omega_{intersection}(\beta) = \sum_{i \sim j} \sqrt{\beta_i^2 + \beta_j^2}\,,$$

$$\Omega_{union}(\beta) = \sup_{\alpha \in \mathbb{R}^p : \forall i \sim j, \|\alpha_i^2 + \alpha_j^2\| \leq 1} \alpha^\top \beta\,.$$

Graph lasso vs kernel on graph

- Graph lasso:

$$\Omega_{\text{graph lasso}}(w) = \sum_{i \sim j} \sqrt{w_i^2 + w_j^2}\,.$$

  constrains the sparsity, not the values

- Graph kernel

$$\Omega_{\text{graph kernel}}(w) = \sum_{i \sim j} (w_i - w_j)^2\,.$$

  constrains the values (smoothness), not the sparsity

Preliminary results

Breast cancer data

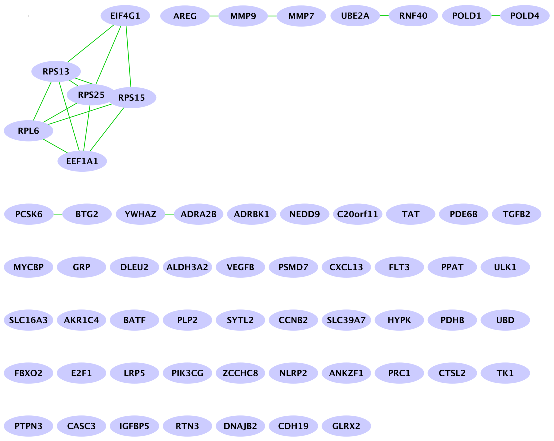
- Gene expression data for $8,141$ genes in 295 breast cancer tumors.
- Canonical pathways from MSigDB containing 639 groups of genes, 637 of which involve genes from our study.

METHOD	ℓ_1	$\Omega^{\mathcal{G}}_{\text{OVERLAP}}(.)$
ERROR	0.38 ± 0.04	0.36 ± 0.03
MEAN $\sharp$ PATH.	130	30

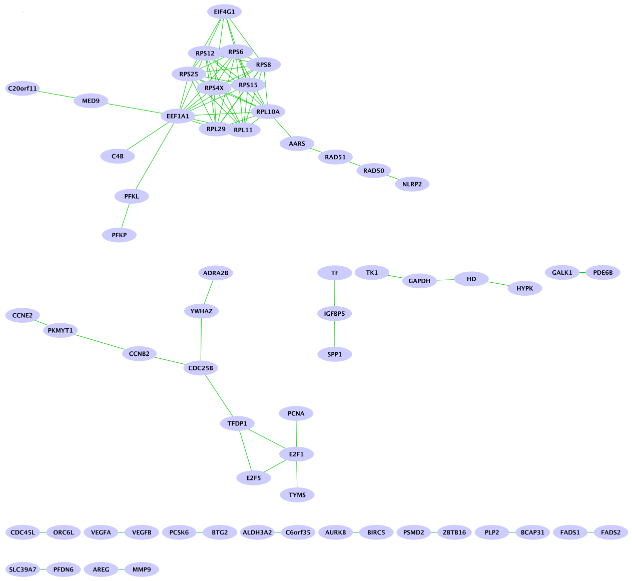
- Graph on the genes.

METHOD	ℓ_1	$\Omega_{graph}(.)$
ERROR	0.39 ± 0.04	0.36 ± 0.01
AV. SIZE C.C.	1.03	1.30

Lasso signature



Graph Lasso signature



Outline

Outline

Gene expression

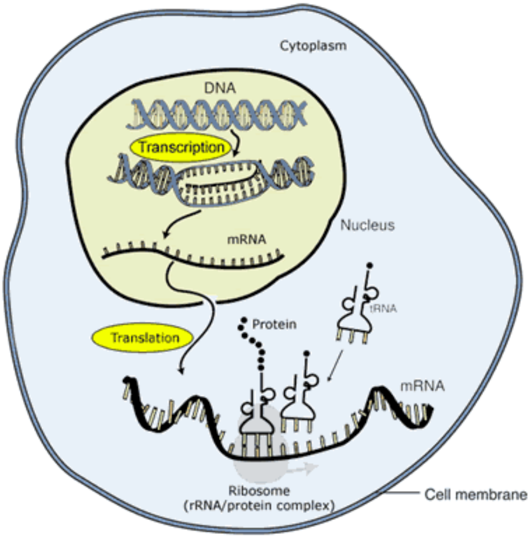
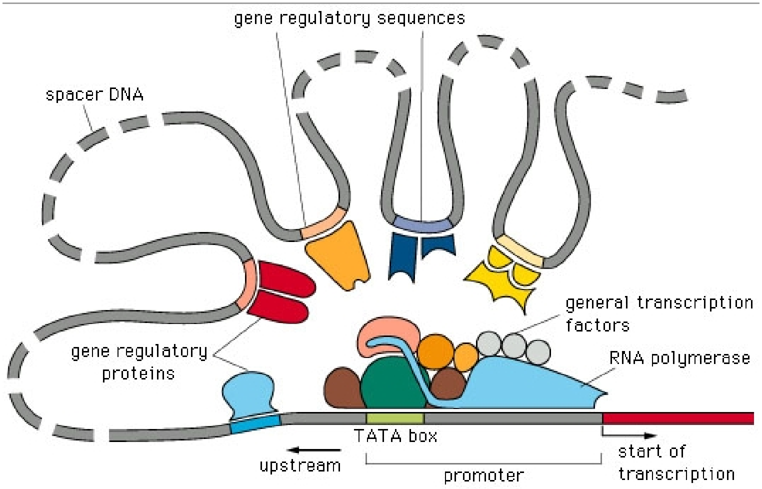
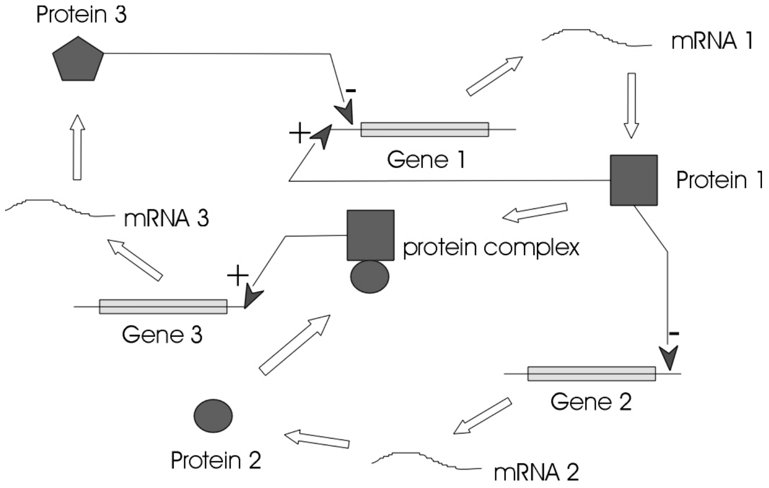


Image adapted from: National Human Genome Research Institute.

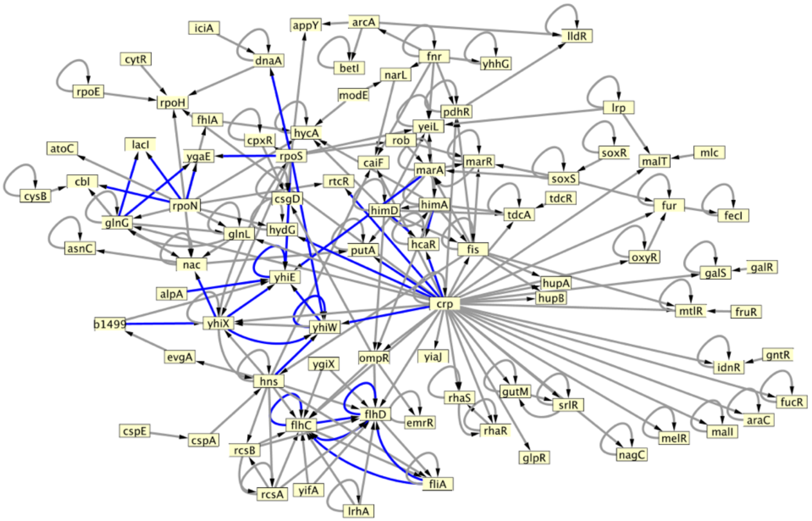
Gene expression regulation



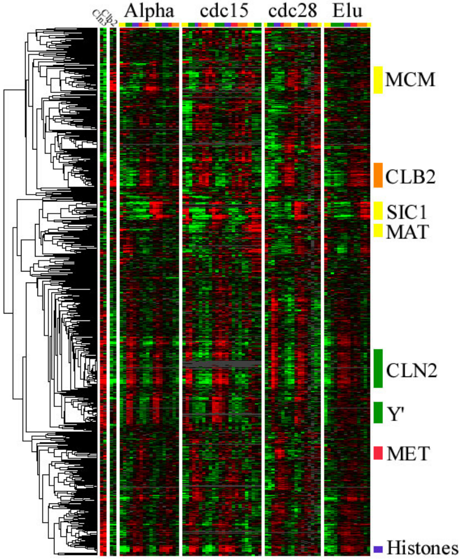
Gene regulatory network



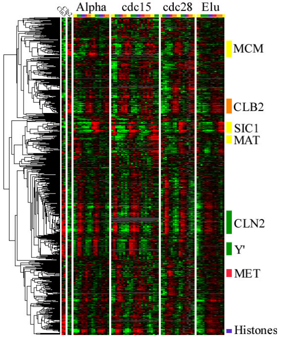
Gene regulatory network of E. coli

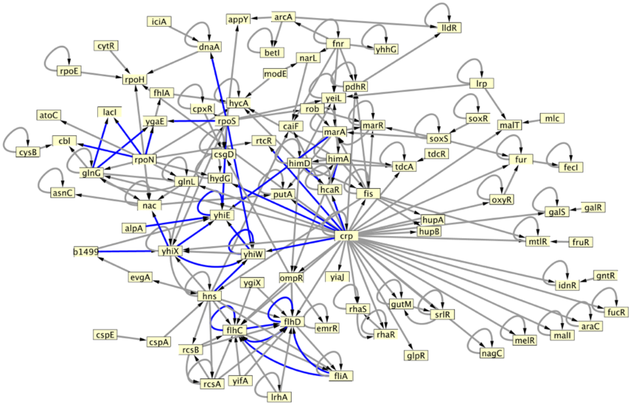


Gene expression data

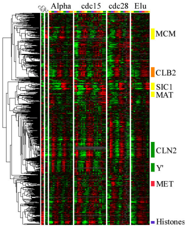


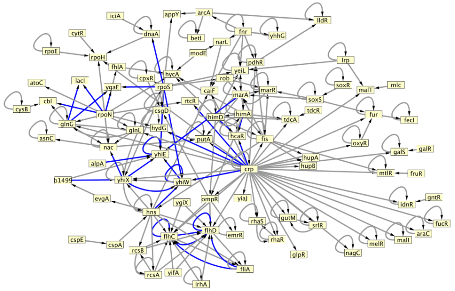
Reconstruction of gene regulatory network





Two flavours: *de novo* or supervised





De novo inference

Given a matrix of expression data, infer regulations

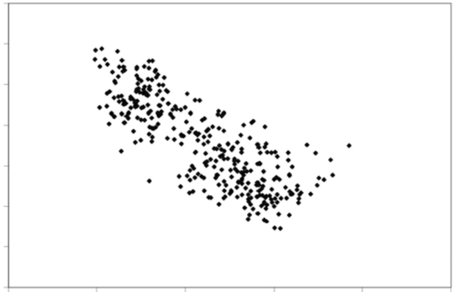
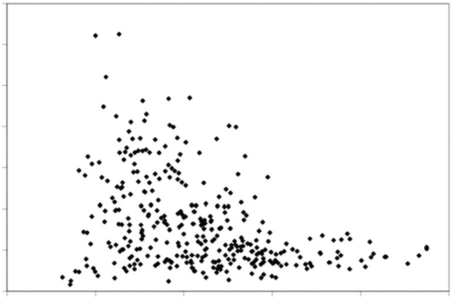
Supervised inference

Given a matrix of expression data and a set of knows regulations, infer *other unknown* regulations

Outline

The idea

If A regulates B, then we should expect some form of "correlation" between the expression levels of A and B across different experiments.

We can therefore try to detect these correlations to infer regulation.

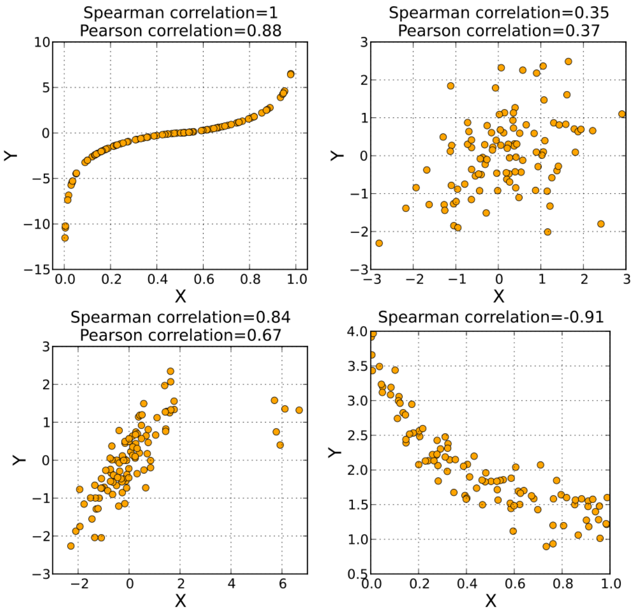
Measuring dependency: correlation coefficients

- $(X_1, Y_1), \ldots, (X_n, Y_n)$ the n expression values of both genes
- Pearson correlation:

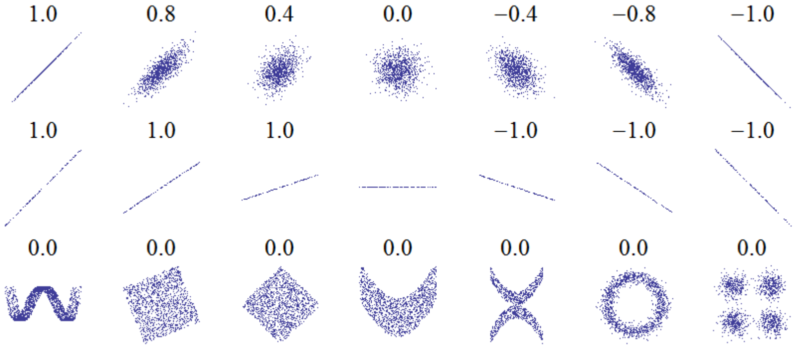
$$\rho = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i (X_i - \bar{X})^2}\sqrt{\sum_i (Y_i - \bar{Y})^2}}$$

- Spearman correlation: similar but replace X_i by its rank.

Illustration



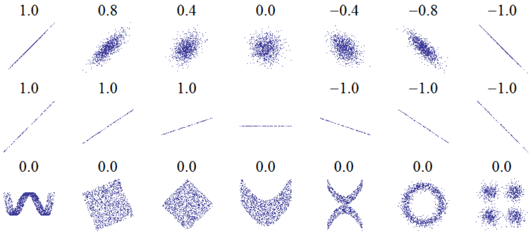
Limit of correlations



Mutual information

$$I(X;Y) = \int_Y \int_X p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) dxdy$$

- $I(X;Y) \geq 0$
- $I(X;Y) = 0$ if and only if X and Y are independent



Outline

The idea

- The dynamic equation of the mRNA concentration of a gene is of the form:
  $$\frac{dX}{dt} = f(X, R)$$
  where R represent the set of concentrations of transcription factors that regulate X.
- At steady state, $dX/dt = 0 = f(X, R)$
- If we linearize $f(X, R) = 0$ we get linear relation of the form
  $$X = \sum_{i \in R} \beta_i X_i$$
- This suggests to look for sets of transcription factors whose concentration is sufficient to explain the level of X across different experiments.

Predicting regulation by sparse regression

Let Y the expression of a gene, and $X_1, \ldots, X_p$ the expression of all TFs.
We look for a model

$$Y = \sum_{i=1}^{p} \beta_i X_i + \text{noise}$$

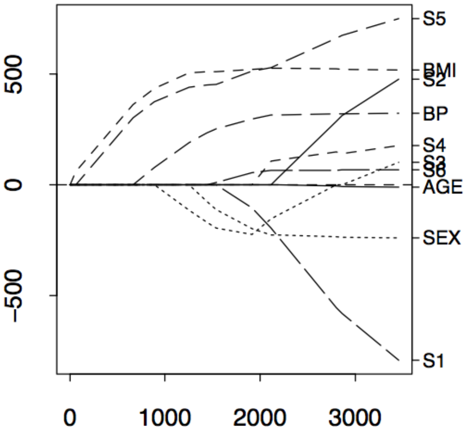
where β is sparse, i.e., only a few β_i are non-zero.
We can estimate the sparse regression model from a matrix of expression data.
Non-zero β_i's correspond to predicted regulators.

Example: sparse regression with the Lasso

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \left(Y_i - \sum_{j=1}^{p} X_i, j\beta_j \right)^2 \text{ such that } \sum_{i=1}^{p} | \beta_i | \leq t$$

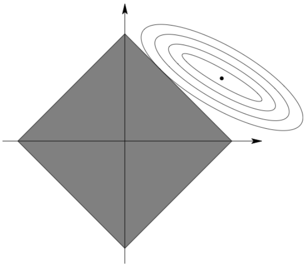
- No explicit solution, but this is just a quadratic program.
- LARS (Efron et al., 2004) provides a fast algorithm to compute the solution for all t's simultaneously (regularization path)
- When t is not too large, the solution will usually be sparse

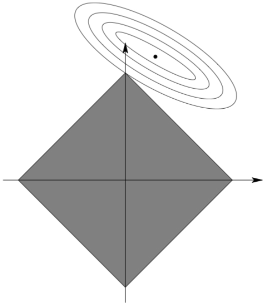
LASSO regression example



Why LASSO leads to sparse solutions

Geometric interpretation with $p = 2$

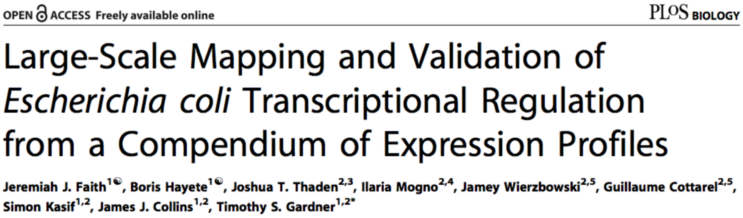




Improved feature selection with stability selection

- For $t = 1$ to T do
  - Bootstrap a random sample S_t from the training set
  - Randomly reweight each feature
  - Select M features, e.g., with the Lassp
- The score of a feature is the number of times it was selected among the T repeats
- Rank features by decreasing score.
- See Meinshausen and Bühlmann (2009).

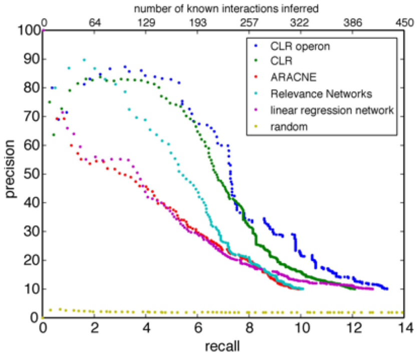
Examples of *de novo* methods



OPEN ACCESS Freely available online PLOS BIOLOGY

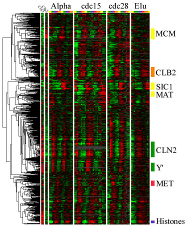
Large-Scale Mapping and Validation of *Escherichia coli* Transcriptional Regulation from a Compendium of Expression Profiles

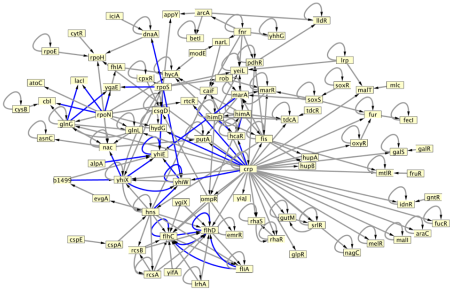
Jeremiah J. Faith[1,☯], Boris Hayete[1,☯], Joshua T. Thaden[2,3], Ilaria Mogno[2,4], Jamey Wierzbowski[2,5], Guillaume Cottarel[2,5], Simon Kasif[1,2], James J. Collins[1,2], Timothy S. Gardner[1,2*]



Outline

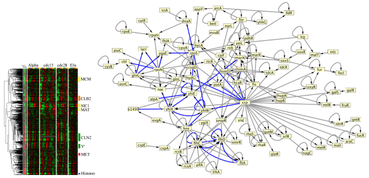
Motivations





- In many cases, we already know quite a few regulations.
- Can we use them, in addition to expression data, to *predict unknown regulations*?

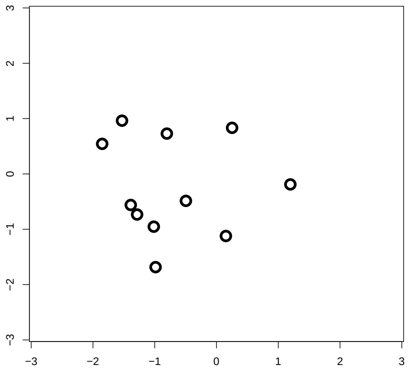
Using expression data for supervised inference



- If a gene has an expression profile similar to other genes known to be regulated by a TF, then it is likely to be regulated by the TF itself
- Underlying hypothesis: genes regulated by the same TF have similar expression variations
- Note that this is very different from *de novo* inference, where we compare the expression profile of the gene to that of the TF
- This is only possible if we already have a list of known regulations.

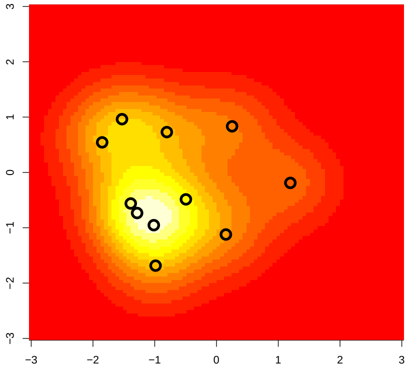
The idea

- For a given TF, let $P \subset [1, n]$ be the set of genes known to be regulated by it



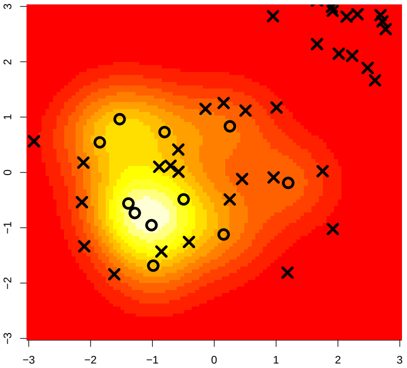
The idea

- For a given TF, let $P \subset [1, n]$ be the set of genes known to be regulated by it
- From the expression profiles $(X_i)_{i \in P}$, estimate a score $s(X)$ to assess which expression profiles X are similar

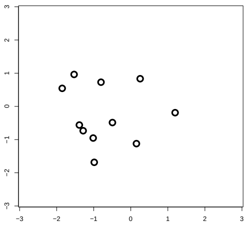


The idea

- For a given TF, let $P \subset [1, n]$ be the set of genes known to be regulated by it
- From the expression profiles $(X_i)_{i \in P}$, estimate a score $s(X)$ to assess which expression profiles X are similar
- Then classify the genes not in P by decreasing score



Estimating the scoring function: examples



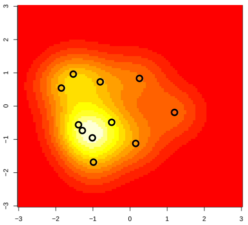
- Kernel density estimation

$$s(X) = \sum_{i \in P} \exp\left(-\gamma \| X - X_i \|^2\right)$$

- One-class SVM

$$s(X) = \sum_{i \in P} \alpha_i \exp\left(-\gamma \| X - X_i \|^2\right)$$

Estimating the scoring function: examples



- Kernel density estimation

$$s(X) = \sum_{i \in P} \exp\left(-\gamma \| X - X_i \|^2\right)$$

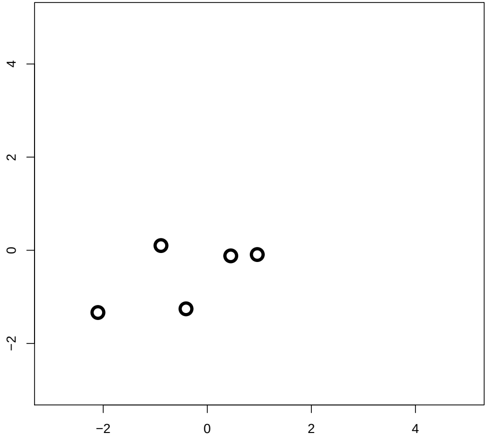
- One-class SVM

$$s(X) = \sum_{i \in P} \alpha_i \exp\left(-\gamma \| X - X_i \|^2\right)$$

Outline

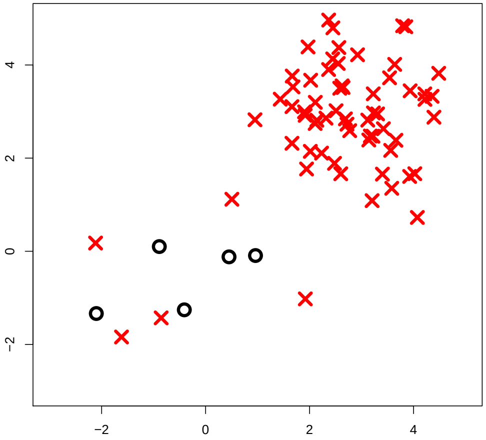
The idea

Since we know in advance all genes, can we use them instead of relying only on genes in P to estimate the scoring function?

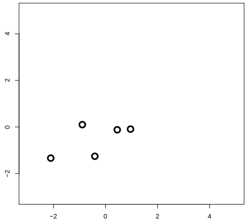


The idea

Since we know in advance all genes, can we use them instead of relying only on genes in P to estimate the scoring function?

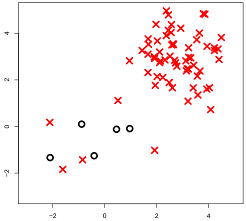


From one-class to PU learning



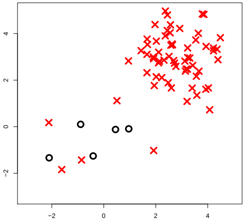
- One class: given genes in P, estimate the function $s(X)$

From one-class to PU learning



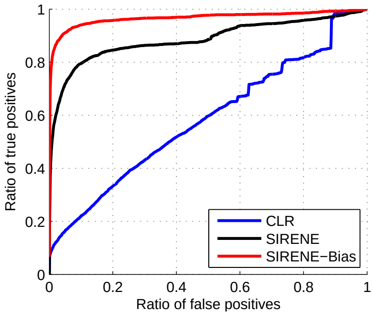
- <span>One class</span>: given genes in P, estimate the function $s(X)$
- PU learning: given genes in P and the set of unlabeled genes U, estimate the scores $s(X_j)$ for $j \in U$

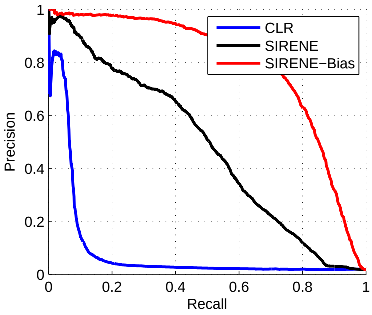
PU learning in practice (Mordelet and V., 2014)



1. Train a classifier to discriminate P from U (eg, SVM or random forest)
2. Rank genes in U by decreasing training score

Example: E. coli regulatory network

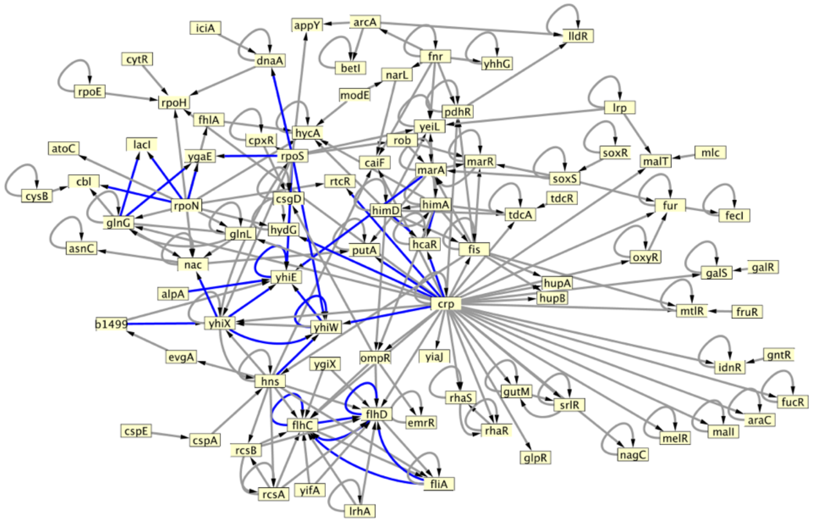




Method	Recall at 60%	Recall at 80%
SIRENE	**44.5%**	**17.6%**
CLR	7.5%	5.5%
Relevance networks	4.7%	3.3%
ARACNe	1%	0%
Bayesian network	1%	0%

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

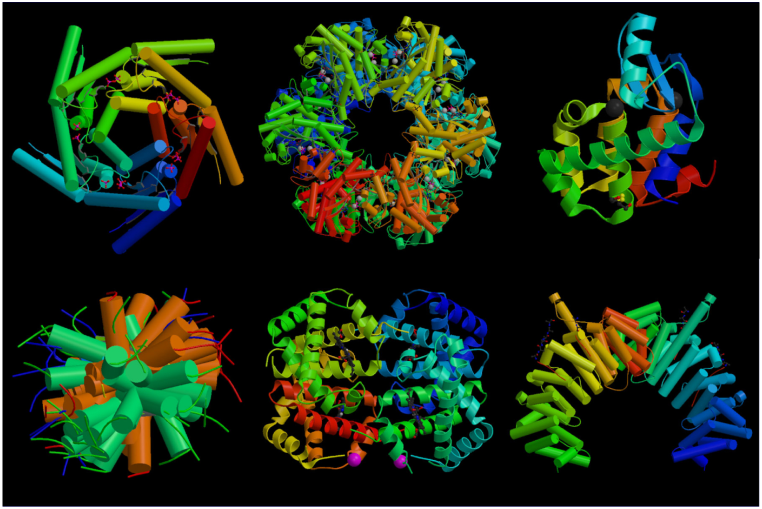
Application: predicted regulatory network (E. coli)



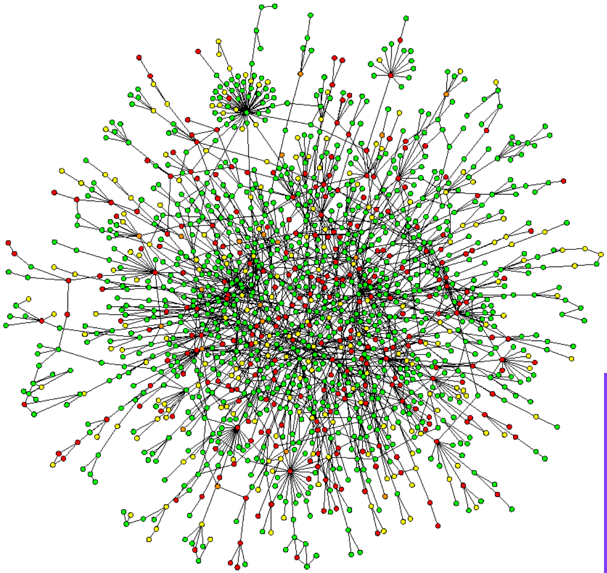
Outline

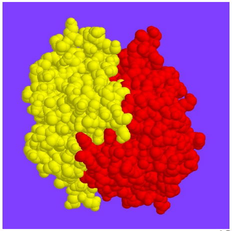
Outline

Proteins

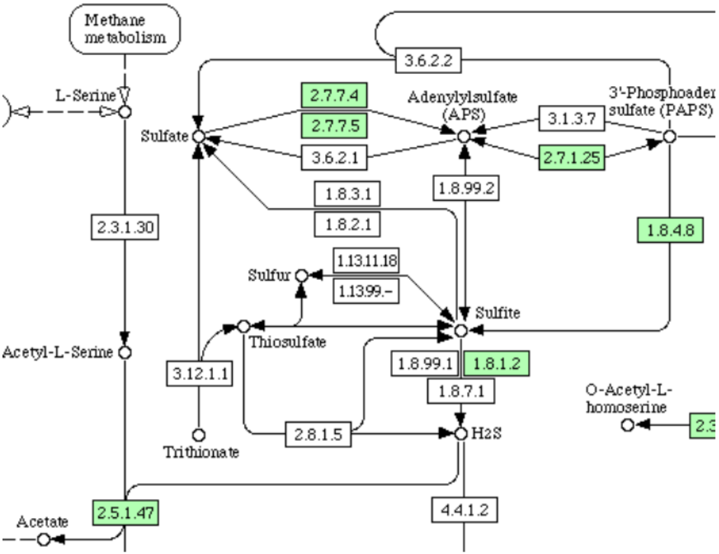


Network 1: protein-protein interaction

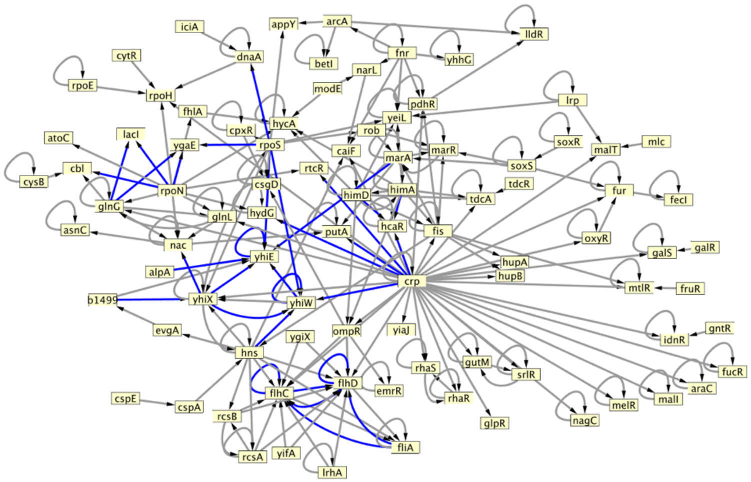




Network 2: metabolic network



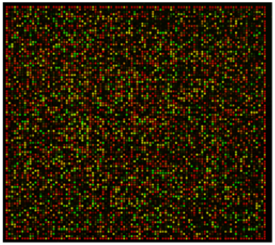
Network 3: gene regulatory network

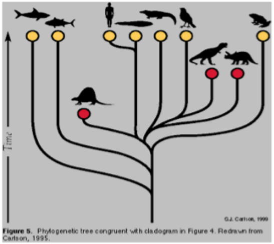


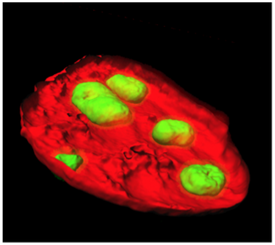
Data available

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

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

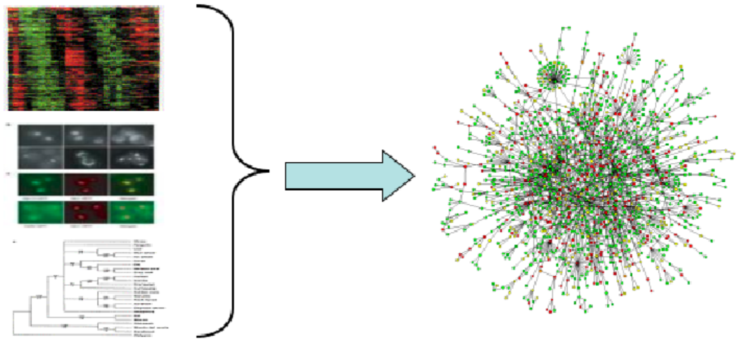



Time
G.J. Carlson, 1999
Figure 5. Phylogenetic tree congruent with cladogram in Figure 4. Redrawn from Carlson, 1995.



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

Our goal



Data

- Gene expression,
- Gene sequence,
- Protein localization, ...

Graph

- Protein-protein interactions,
- Metabolic pathways,
- Signaling pathways, ...

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 $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. We focus on undirected graphs.

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 $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. We focus on undirected graphs.

"De novo" inference

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

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 $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. We focus on undirected graphs.

"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 $\mathcal{E}_{test} = \mathcal{E} \backslash \mathcal{E}_{train}$

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)

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

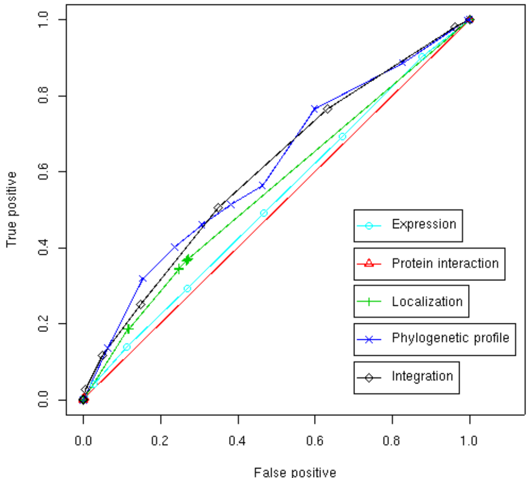
- Excellent approach if the model is correct and enough data are available
- Interpretability of the model
- Inclusion of prior knowledge

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).

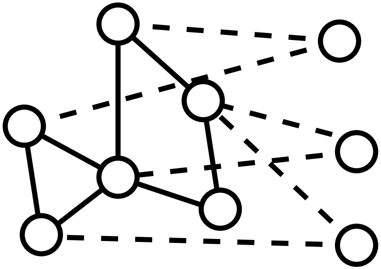


Supervised methods

Motivation

In actual applications,

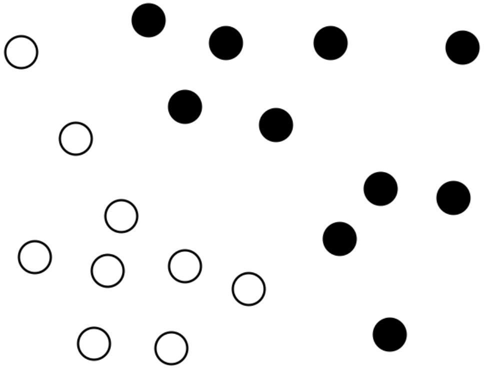
- we know in advance parts of the network to be inferred
- the problem is to add/remove nodes and edges using genomic data as side information



Supervised method

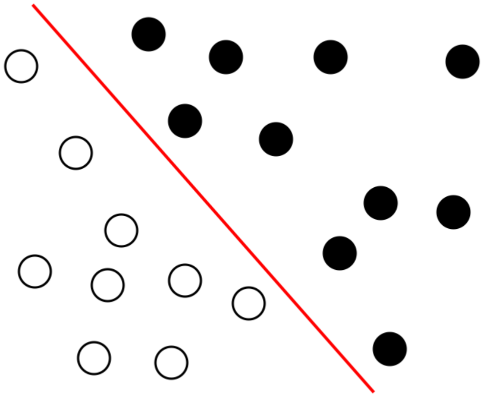
- Given genomic data and the currently known network...
- Infer missing edges between current nodes and additional nodes.

Pattern recognition



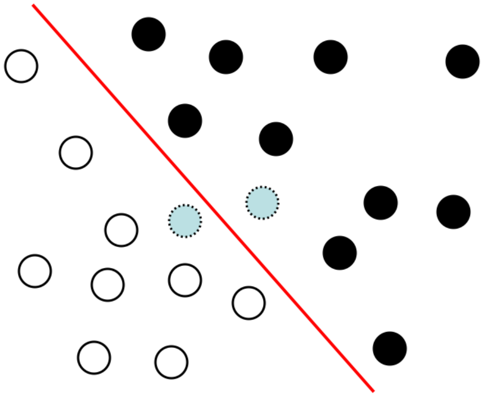
- Given a training set of patterns in two classes, learn to discriminate them
- Many algorithms (ANN, SVM, Decision tress, ...)

Pattern recognition



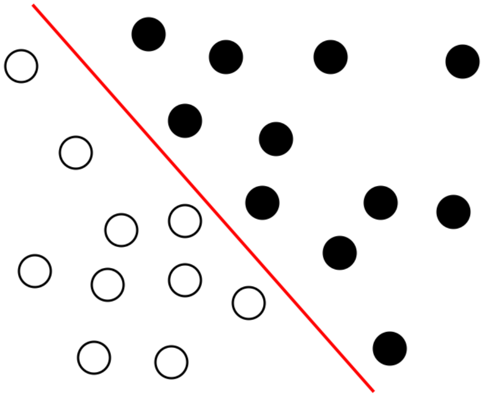
- Given a training set of patterns in two classes, learn to discriminate them
- Many algorithms (ANN, SVM, Decision tress, ...)

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)

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)

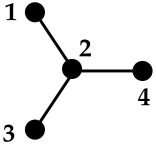
Two solutions

- 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

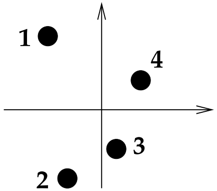
Outline

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!



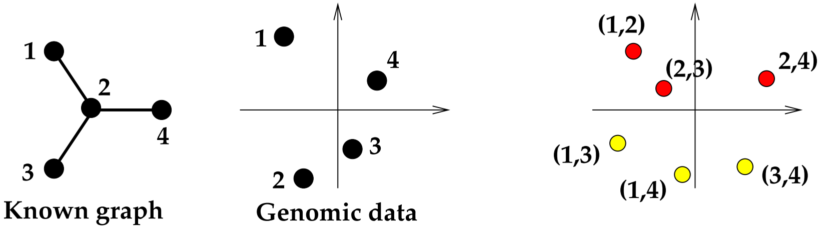
Known graph



Genomic data

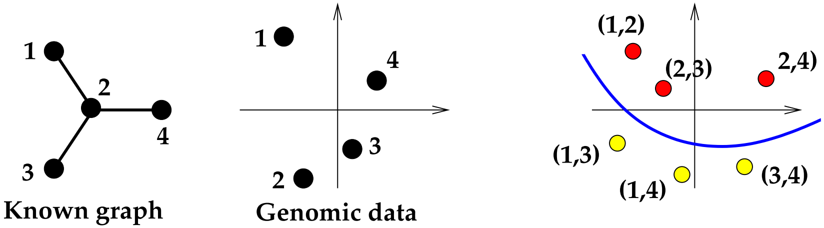
Pattern recognition for pairs: basic issue

- 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):

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

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:

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

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...

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...):

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

Representing an unordered pair

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

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

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:

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

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, 2005):

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

Another idea: metric learning

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

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

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?

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

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

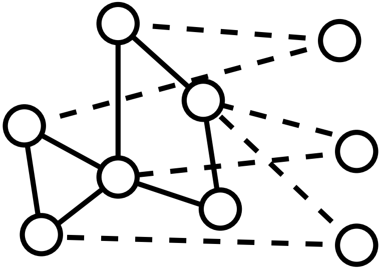
  trained to discriminate connected from non-connected pairs, solves this metric learning problem without the constraint $M \geq 0$.
- Equivalently, train the SVM over pairs with the metric learning pairwise kernel:

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

Outline

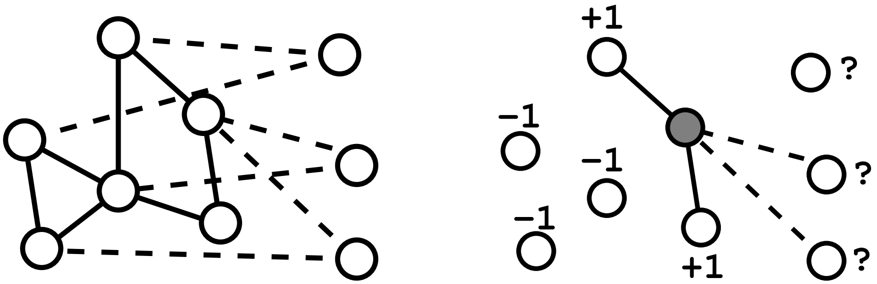
The idea (Bleakley et al., 2007)

- 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.

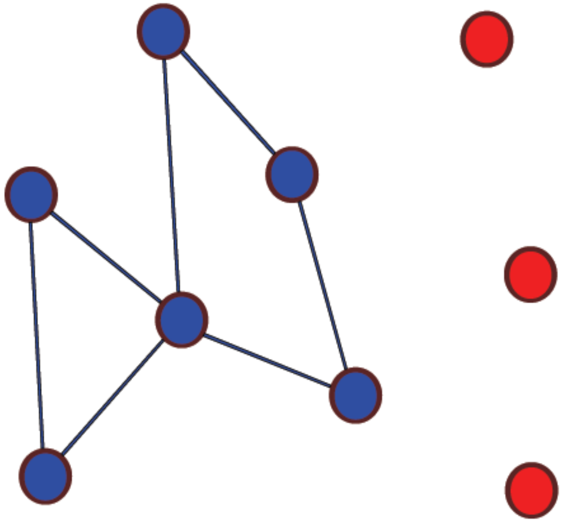


The idea (Bleakley et al., 2007)

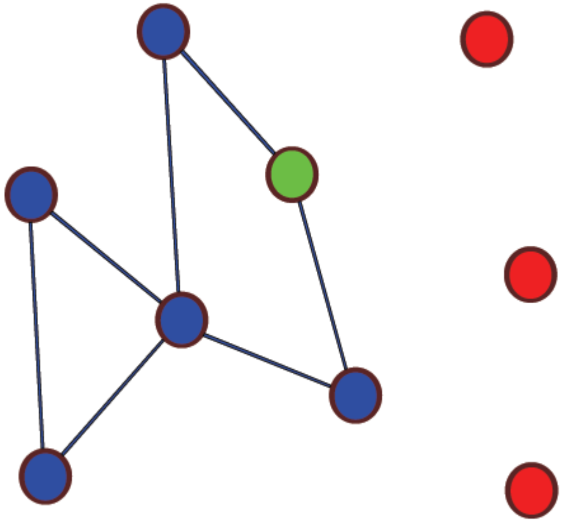
- Motivation: define specific models for each target node to discriminate between its neighbors and the others
- Treat each node independently from the other. Then combine predictions for ranking candidate edges.



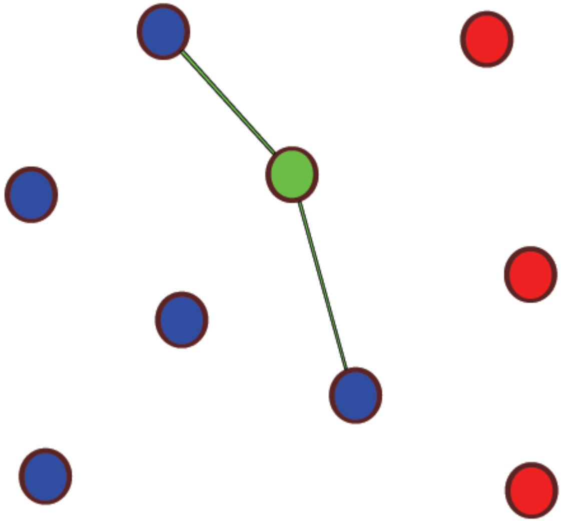
The LOCAL model



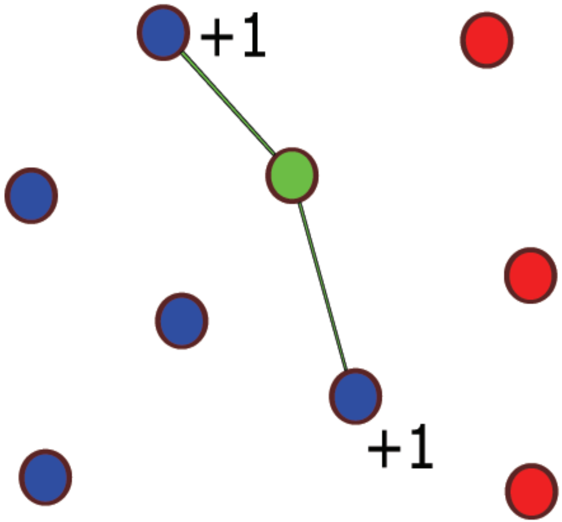
The LOCAL model



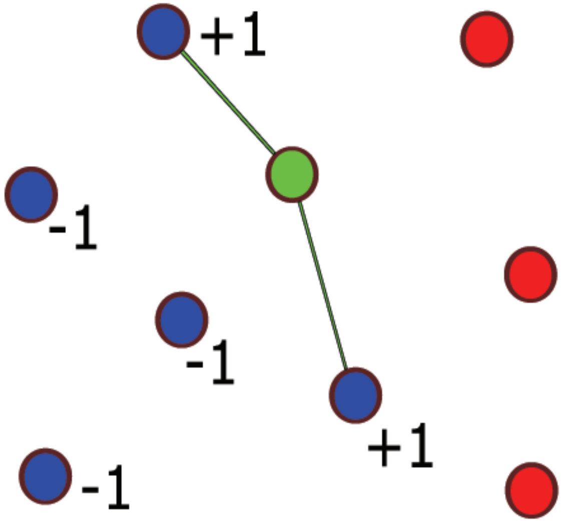
The LOCAL model



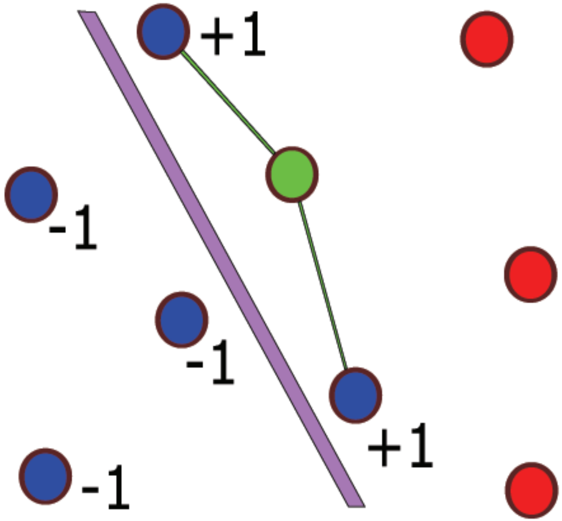
The LOCAL model



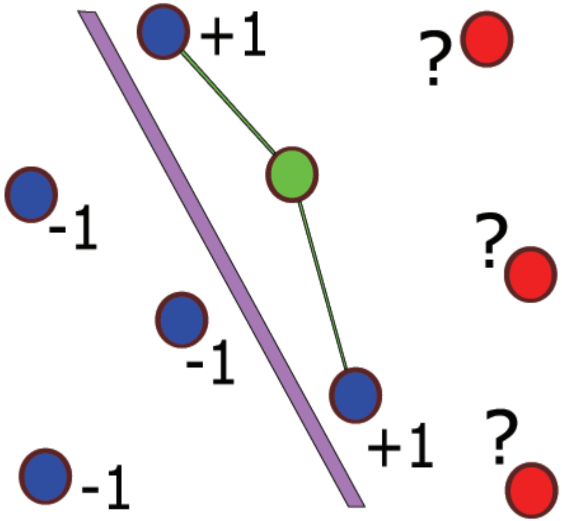
The LOCAL model



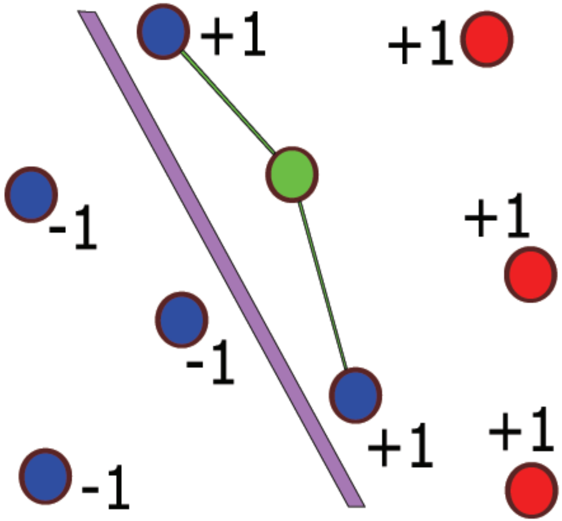
The LOCAL model



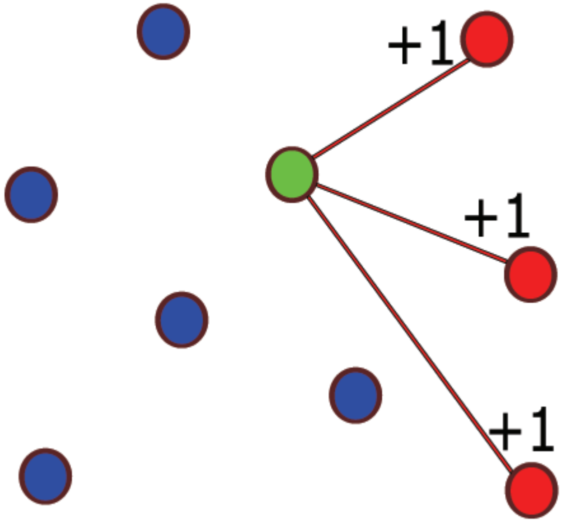
The LOCAL model



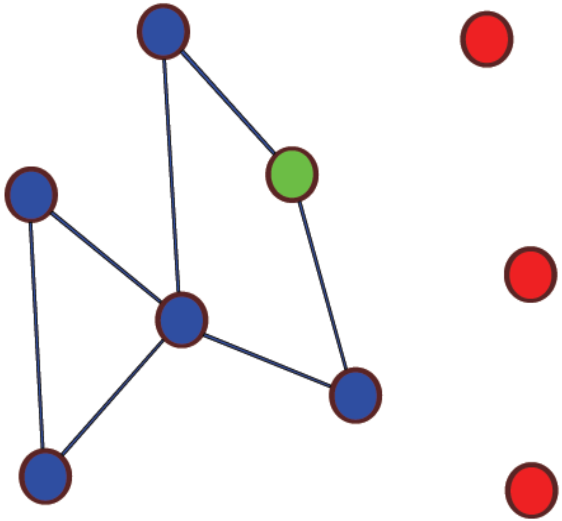
The LOCAL model



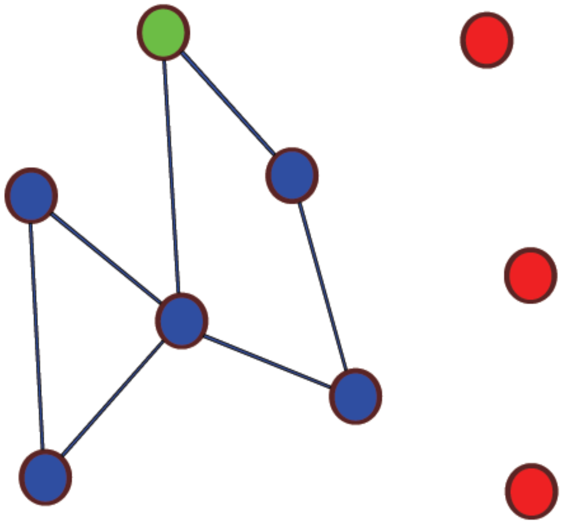
The LOCAL model



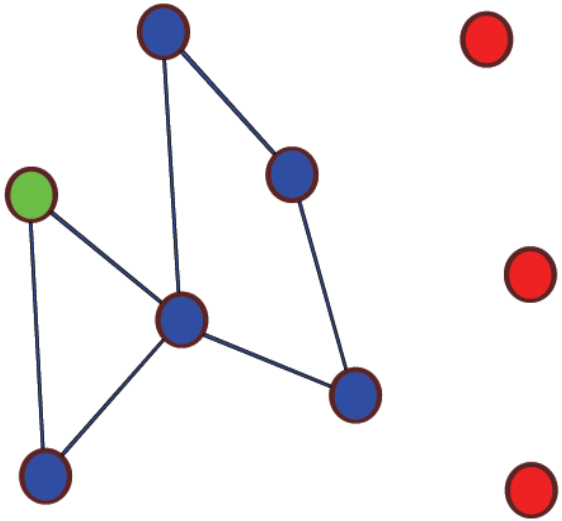
The LOCAL model



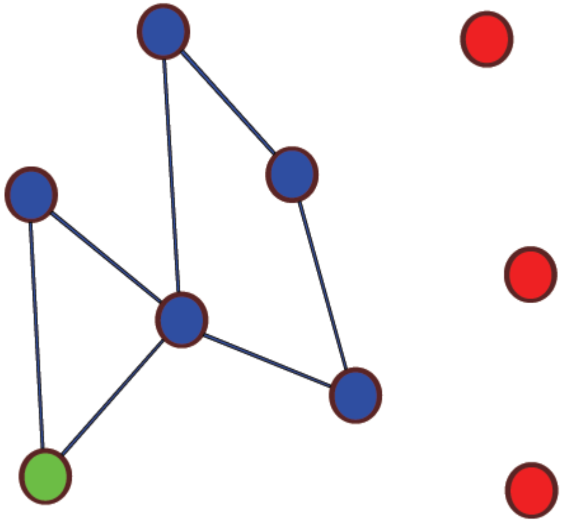
The LOCAL model



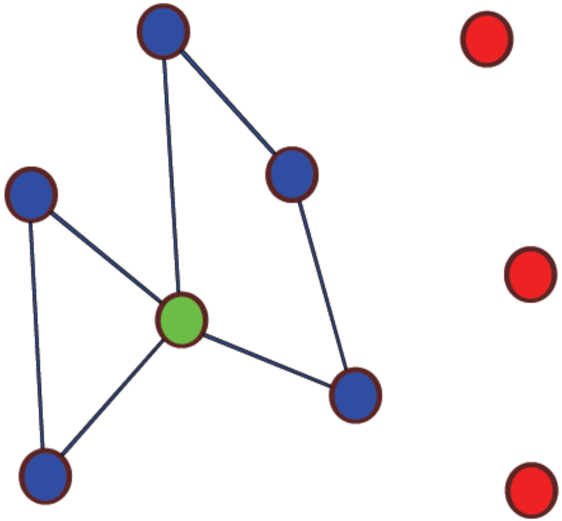
The LOCAL model



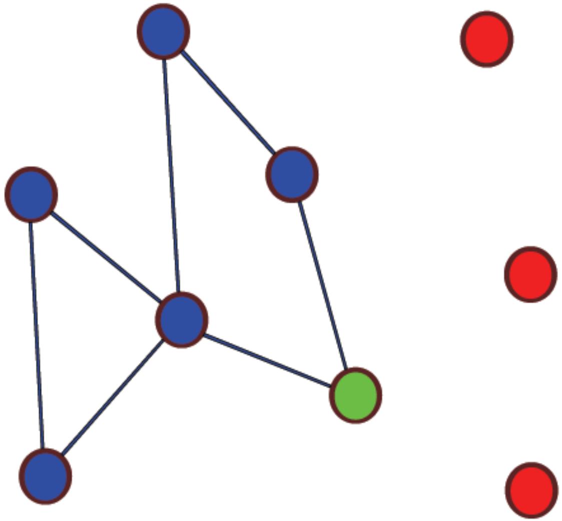
The LOCAL model



The LOCAL model



The LOCAL model



A few remarks

- 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 $\{A, B\}$

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.

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.

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

Outline

Motivation

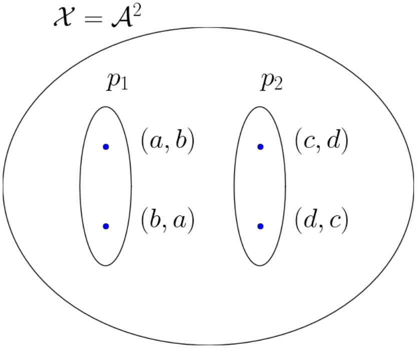
In the case of unordered pairs $\{A, B\}$, pairwise kernels such as the TPPK and local models look very different:

- 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 $K_{\mathcal{X}}$:

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

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

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

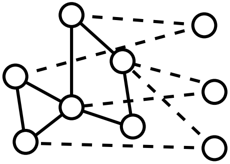
Let $\mathcal{X} = \mathcal{A}^2$ be endowed with the p.d. kernel:

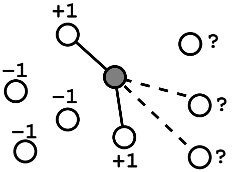
$$K_{\mathcal{X}}\left((a,b),(c,d)\right) = 2K_{\mathcal{A}}(a,c)K_{\mathcal{A}}(b,d)\,. \tag{4}$$

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 δ 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 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

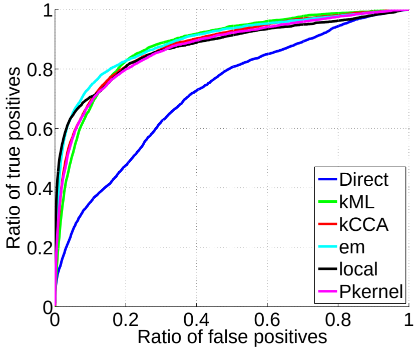
Interpolation:

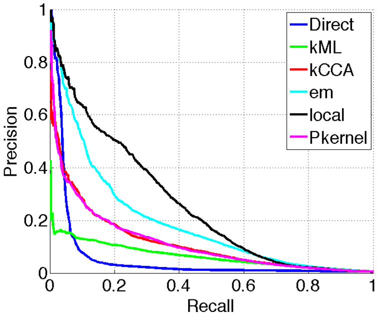
$$K_{\mathcal{X}} = \left((1-\lambda)K_{\mathcal{A}} + \lambda\delta\right) \otimes K_{\mathcal{A}}$$

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

Outline

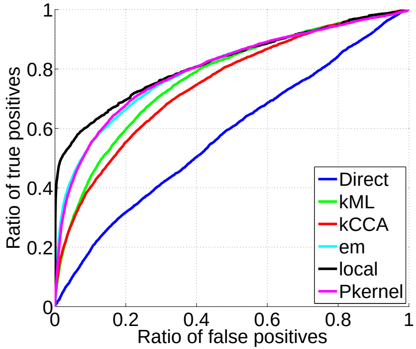
Results: protein-protein interaction (yeast)

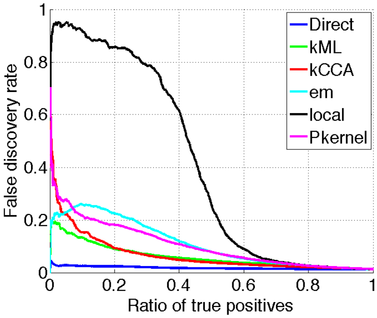




(from Bleakley et al., 2007)

Results: metabolic gene network (yeast)





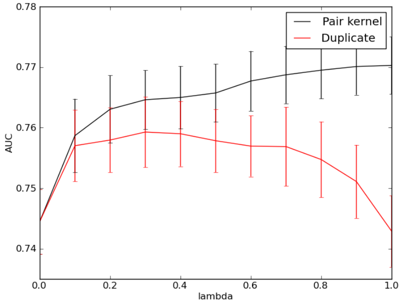
(from Bleakley et al., 2007)

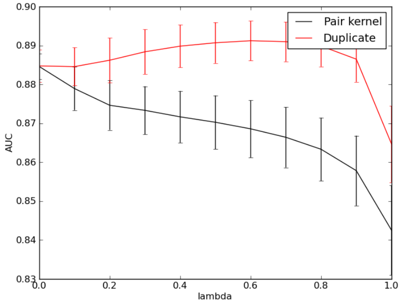
Interpolation kernel

Table: Strategy and kernel realizing the maximum mean AUC for nine metabolic and protein-protein interaction networks experiments, with the kernel K^λ 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

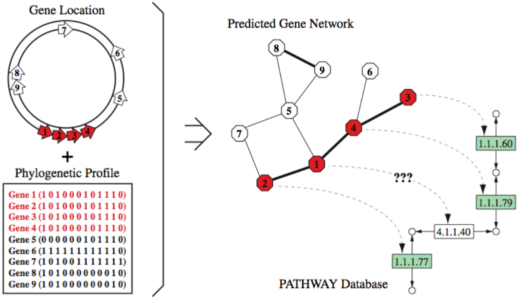
the FEBS Journal

Prediction of missing enzyme genes in a bacterial metabolic network

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

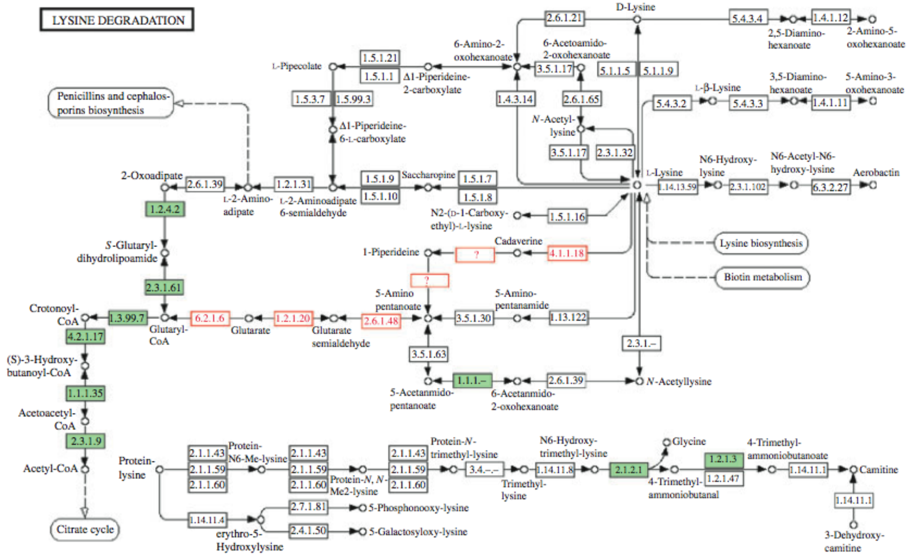
Yoshihiro Yamanishi[1], Hisaaki Mihara[2], Motoharu Osaki[2], Hisashi Muramatsu[3], Nobuyoshi Esaki[2], Tetsuya Sato[1], Yoshiyuki Hizukuri[1], Susumu Goto[1] and Minoru Kanehisa[1]

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

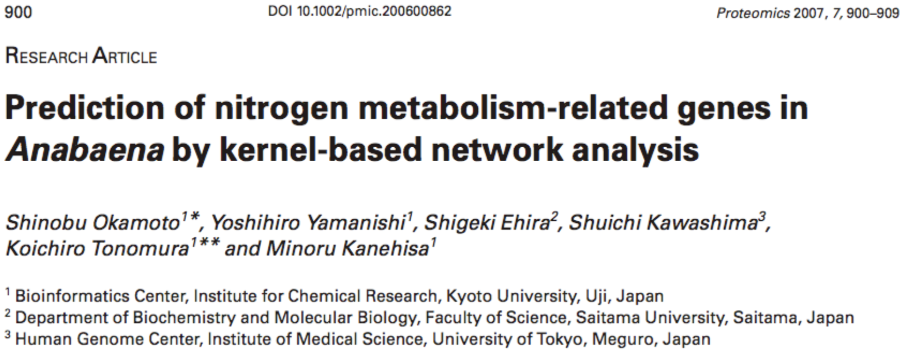


Gene 1 (1 0 1 0 0 0 1 0 1 1 1 0)
Gene 2 (1 0 1 0 0 0 1 0 1 1 1 0)
Gene 3 (1 0 1 0 0 0 1 0 1 1 1 0)
Gene 4 (1 0 1 0 0 0 1 0 1 1 1 0)
Gene 5 (0 0 0 0 0 0 1 0 1 1 1 0)
Gene 6 (1 1 1 1 1 1 1 1 1 1 1 0)
Gene 7 (1 0 1 0 0 1 1 1 1 1 1 1)
Gene 8 (1 0 1 0 0 0 0 0 0 0 1 0)
Gene 9 (1 0 1 0 0 0 0 0 0 0 1 0)

Applications: missing enzyme prediction



Applications: missing enzyme prediction



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

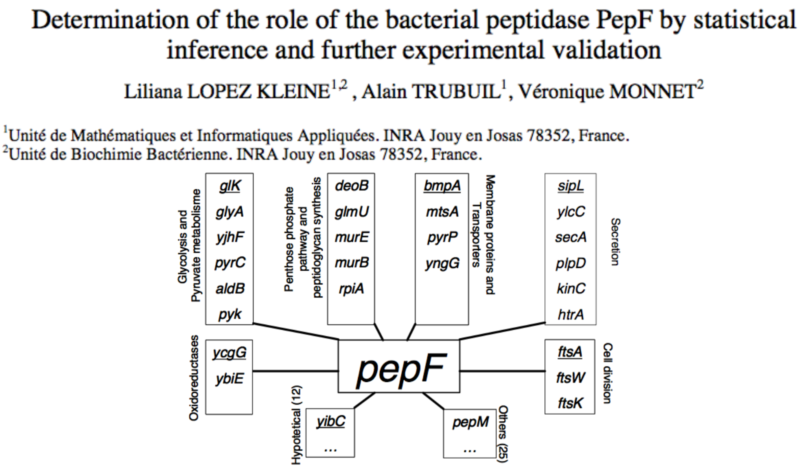
RESEARCH ARTICLE

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.

Glycolysis and Pyruvate metabolisme
glK
glyA
yjhF
pyrC
aldB
pyk
Penthose phosphate pathway and peptidoglycan synthesis
deoB
glmU
murE
murB
rpiA
bmpA
mtsA
pyrP
yngG
Membrane proteins and Transporters
sipL
ylcC
secA
plpD
kinC
htrA
Secretion
Oxidoreductases
ycgG
ybiE
pepF
ftsA
ftsW
ftsK
Cell division
Hypotetical (12)
yibC
...
pepM
...
Others (25)

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?

Outline

Machine learning in computational and systems biology

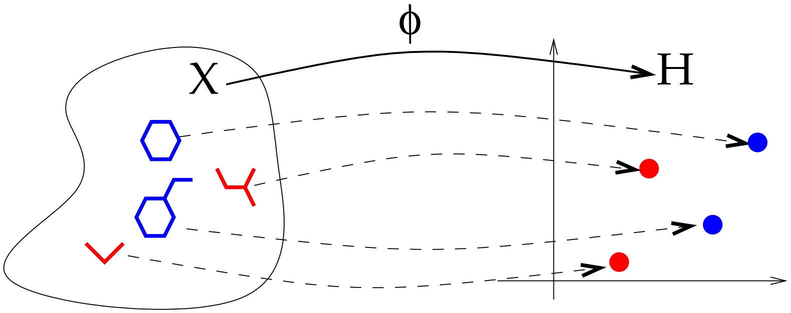
- Biology faces a flood of data following the development of high-throughput technologies (sequencing, DNA chips, ...)
- Many problems can be formalized in the framework of machine learning, e.g.:
  - Protein annotation
  - Drug discovery, virtual screening
  - Gene network inference
- These data have often complex structures (strings, graphs, high-dimensional vectors) and often require dedicated algorithms.

Support vector machines (SVM)

- A general-purpose algorithm for pattern recognition
- Based on the principle of large margin *("séparateur à vaste marge")*
- Linear or nonlinear with the kernel trick
- Control of the regularization / data fitting trade-off with the C parameter
- State-of-the-art performance on many applications

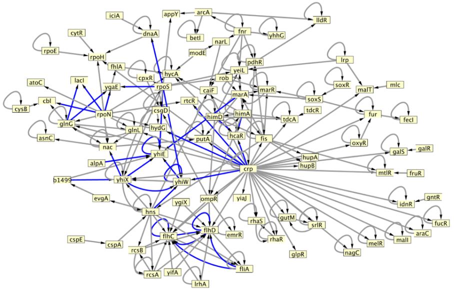
Kernels

- A central ingredient of SVM
- Allows nonlinearity
- Allows to work implicitly in a high-dimensional feature space
- Allows to work with structured data (e.g., graphs)



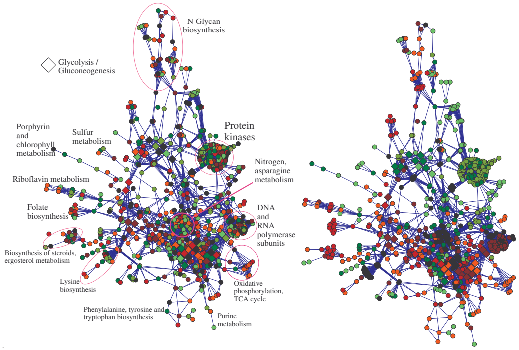
Gene network inference

- *Ab initio* reconstruction of regulatory network can be formulated as feature selection, and solved, e.g., by the Lasso or random forests
- Supervised reconstruction is more powerful when edges (e.g., regulations) are already known
- PU learning is more powerful than one-class learning in this setting, and can be solved by SVM
- Predicting edges requires learning over pairs with specific kernels in the case of SVM



Using gene networks

- Gene networks can be used as prior knowledge to analyze gene expression data
- Spectral graph analysis and graph kernels are useful tools
- It allows to capture pathways or protein complexes instead of individual genes


N Glycan biosynthesis
Glycolysis / Gluconeogenesis
Porphyrin and chlorophyll metabolism
Sulfur metabolism
Protein kinases
Nitrogen, asparagine metabolism
Riboflavin metabolism
Folate biosynthesis
DNA and RNA polymerase subunits
Biosynthesis of steroids, ergosterol metabolism
Lysine biosynthesis
Oxidative phosphorylation, TCA cycle
Phenylalanine, tyrosine and tryptophan biosynthesis
Purine metabolism

References

N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68:337 – 404, 1950. URL http://www.jstor.org/stable/1990404.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 6, New York, NY, USA, 2004. ACM. doi: 10.1145/1015330.1015424. URL http://doi.acm.org/10.1145/1015330.1015424.

P. Bartlett, M. Jordan, and J. McAuliffe. Convexity, classification and risk bounds. Technical Report 638, UC Berkeley Statistics, 2003.

A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl. 1):i38–i46, Jun 2005. doi: 10.1093/bioinformatics/bti1016. URL http://dx.doi.org/10.1093/bioinformatics/bti1016.

K. Bleakley, G. Biau, and J.-P. Vert. Supervised reconstruction of biological networks with local models. *Bioinformatics*, 23(13):i57–i65, Jul 2007. doi: 10.1093/bioinformatics/btm204. URL http://dx.doi.org/10.1093/bioinformatics/btm204.

V. Boeva, A. Zinovyev, K. Bleakley, J.-P. Vert, I. Janoueix-Lerosey, O. Delattre, and E. Barillot. Control-free calling of copy number alterations in deep-sequencing data using GC-content normalization. *Bioinformatics*, 27(2):268–269, Jan 2011. doi: 10.1093/bioinformatics/btq635. URL http://dx.doi.org/10.1093/bioinformatics/btq635.

References (cont.)

V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. The convex geometry of linear inverse problems. *Found. Comput. Math.*, 12(6):805–849, 2012. doi: 10.1007/s10208-012-9135-7. URL http://dx.doi.org/10.1007/s10208-012-9135-7.

S. S. Chen, D. L. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, 1998. doi: 10.1137/S1064827596304010. URL http://dx.doi.org/10.1137/S1064827596304010.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Stat.*, 32(2): 407–499, 2004. doi: 10.1214/009053604000000067. URL http://dx.doi.org/10.1214/009053604000000067.

J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner. Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol.*, 5(1):e8, Jan 2007. doi: 10.1371/journal.pbio.0050008. URL http://dx.doi.org/10.1371/journal.pbio.0050008.

J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Ann. Appl. Statist.*, 1(1):302–332, 2007. doi: 10.1214/07-AOAS131. URL http://dx.doi.org/10.1214/07-AOAS131.

G. Furnival and R. Wilson. Regressions by leaps and bounds. *Technometrics*, 16(4):499–511, 1974. URL http://www.jstor.org/stable/1267601.

References (cont.)

T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: hardness results and efficient alternatives. In B. Schölkopf and M. Warmuth, editors, *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory and the Seventh Annual Workshop on Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143, Heidelberg, 2003. Springer. doi: 10.1007/b12006. URL `http://dx.doi.org/10.1007/b12006`.

Z. Harchaoui and C. Levy-Leduc. Multiple change-point estimation with a total variation penalty. *J. Am. Stat. Assoc.*, 105(492):1480–1493, 2010. doi: 10.1198/jasa.2010.tm09181. URL `http://dx.doi.org/10.1198/jasa.2010.tm09181`.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2001.

A. E. Hoerl and R. W. Kennard. Ridge regression : biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

M. Hue and J.-P. Vert. On learning with kernels for unordered pairs. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 463–470. Omnipress, 2010. URL `http://www.icml2010.org/papers/520.pdf`.

L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553431. URL `http://dx.doi.org/10.1145/1553374.1553431`.

References (cont.)

R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. *J. Mach. Learn. Res.*, 12:2777–2824, 2011. URL http://www.jmlr.org/papers/volume12/jenatton11b/jenatton11b.pdf.

G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004a. URL http://www.jmlr.org/papers/v5/lanckriet04a.html.

G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004b. doi: 10.1093/bioinformatics/bth294. URL http://bioinformatics.oupjournals.org/cgi/content/abstract/20/16/2626.

S. R. Land and J. H. Friedman. Variable fusion: A new adaptive signal regression method. Technical Report 656, Department of Statistics, Carnegie Mellon University Pittsburgh, 1997. URL http://www.stat.cmu.edu/tr/tr656/tr656.html.

S. Le Cessie and J. C. van Houwelingen. Ridge estimators in logistic regression. *Appl. Statist.*, 41(1):191–201, 1992. URL http://www.jstor.org/stable/2347628.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11:19–60, 2010. URL http://jmlr.csail.mit.edu/papers/v11/mairal10a.html.

N. Meinshausen and P. Bühlmann. Stability selection. *J. R. Stat. Soc. Ser. B*, 72(4):417–473, 2010. doi: 10.1111/j.1467-9868.2010.00740.x. URL http://dx.doi.org/10.1111/j.1467-9868.2010.00740.x.

References (cont.)

F. Mordelet and J.-P. Vert. SIRENE: Supervised inference of regulatory networks. *Bioinformatics*, 24(16):i76–i82, 2008. doi: 10.1093/bioinformatics/btn273. URL `http://dx.doi.org/10.1093/bioinformatics/btn273`.

F. Mordelet and J.-P. Vert. A bagging SVM to learn from positive and unlabeled examples. *Pattern Recognition Lett.*, 37:201–209, 2014. doi: 10.1016/j.patrec.2013.06.010. URL `http://dx.doi.org/10.1016/j.patrec.2013.06.010`.

S. Okamoto, Y. Yamanishi, S. Ehira, S. Kawashima, K. Tonomura, and M. Kanehisa. Prediction of nitrogen metabolism-related genes in anabaena by kernel-based network analysis. *Proteomics*, 7(6):900–909, Mar 2007. doi: 10.1002/pmic.200600862. URL `http://dx.doi.org/10.1002/pmic.200600862`.

F. Rapaport, A. Zinovyev, M. Dutreix, E. Barillot, and J.-P. Vert. Classification of microarray data using gene networks. *BMC Bioinformatics*, 8:35, 2007. doi: 10.1186/1471-2105-8-35. URL `http://dx.doi.org/10.1186/1471-2105-8-35`.

F. Rapaport, E. Barillot, and J.-P. Vert. Classification of arrayCGH data using fused SVM. *Bioinformatics*, 24(13):i375–i382, Jul 2008. doi: 10.1093/bioinformatics/btn188. URL `http://dx.doi.org/10.1093/bioinformatics/btn188`.

L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992. doi: 10.1016/0167-2789(92)90242-F. URL `http://dx.doi.org/10.1016/0167-2789(92)90242-F`.

References (cont.)

H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004. URL http://bioinformatics.oupjournals.org/cgi/content/abstract/20/11/1682.

M. R. Stratton, P. J. Campbell, and P. A. Futreal. The cancer genome. *Nature*, 458(7239): 719–724, Apr 2009. doi: 10.1038/nature07943. URL http://dx.doi.org/10.1038/nature07943.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B*, 58(1): 267–288, 1996. URL http://www.jstor.org/stable/2346178.

R. Tibshirani and P. Wang. Spatial smoothing and hot spot detection for cgh data using the fused lasso. *Biostatistics (Oxford, England)*, 9(1):18–29, January 2008. ISSN 1465-4644. doi: 10.1093/biostatistics/kxm013. URL http://dx.doi.org/10.1093/biostatistics/kxm013.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(1):91–108, 2005. URL http://ideas.repec.org/a/bla/jorssb/v67y2005i1p91-108.html.

J.-P. Vert and K. Bleakley. Fast detection of multiple change-points shared by many signals using group LARS. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Adv. Neural. Inform. Process Syst.*, volume 22, pages 2343–2352, 2010. URL http://books.nips.cc/papers/files/nips23/NIPS2010_1131.pdf.

J.-P. Vert, J. Qiu, and W. S. Noble. A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, 8 Suppl 10:S8, 2007. doi: 10.1186/1471-2105-8-S10-S8. URL http://dx.doi.org/10.1186/1471-2105-8-S10-S8.

References (cont.)

J. D. Watson and F. H. C. Crick. A Structure for Deoxyribose Nucleic Acid. *Nature*, 171:737, 1953. URL `http://www.nature.com/genomics/human/watson-crick/index.html`.

Y. Yamanishi, J.-P. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20:i363–i370, 2004. URL `http://bioinformatics.oupjournals.org/cgi/reprint/19/suppl_1/i323`.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. Ser. B*, 68(1):49–67, 2006. doi: 10.1111/j.1467-9868.2005.00532.x. URL `http://dx.doi.org/10.1111/j.1467-9868.2005.00532.x`.