Machine learning in post-genomic

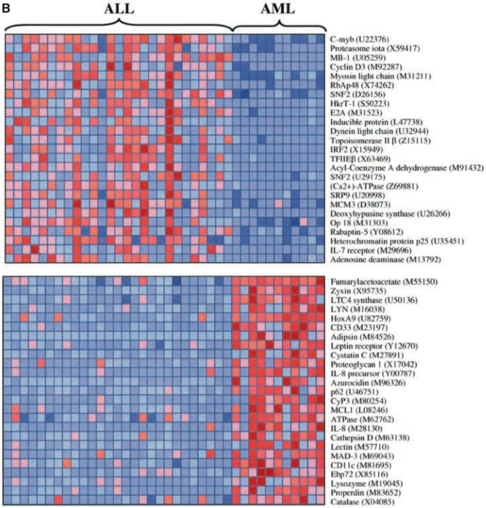
Jean-Philippe Vert

Jean-Philippe.Vert@ensmp.fr

Mines ParisTech / Institut Curie / Inserm

Atelier Statistique de la SFDS, Paris, November 27, 2008.

Tissue classification from microarray data



Goal

- Design a classifier to automatically assign a class to future samples from their expression profile
- Interpret biologically the differences between the classes

Supervised sequence classification

Data (training)

- Secreted proteins:

  MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNIEA...

  MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...

  MALHTVLIMLSLLPMLEAQNPEHANITIGEPITNETLGWL...

  ...
- Non-secreted proteins:

  MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVNLGVG...

  MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...

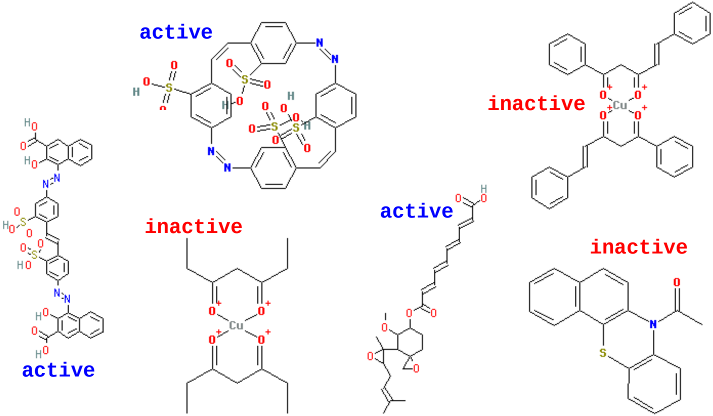
  MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..

  ...

Goal

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

Ligand-Based Virtual Screening and QSAR



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

Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

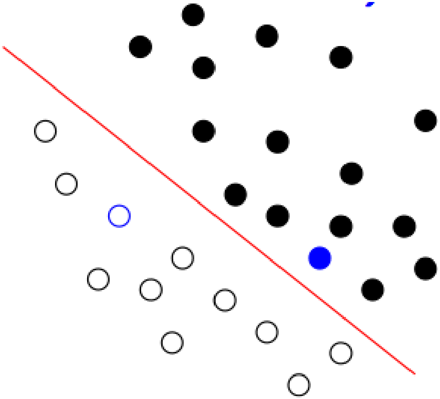
Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

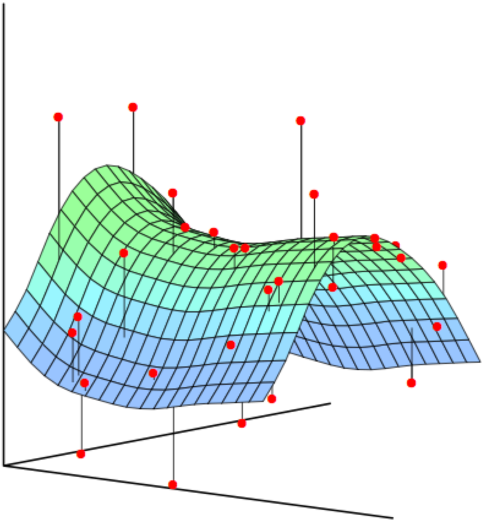
Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Pattern recognition, *aka* supervised classification



Regression



From Hastie et al. (2001) *The elements of statistical learning*

Formalization

Input

- $\mathcal{X}$ the space of patterns (typically, $\mathcal{X} = \mathbb{R}^p$)
- $\mathcal{Y}$ the space of response or labels
  - Regression : $\mathcal{Y} = \mathbb{R}$
  - Pattern recognition : $\mathcal{Y} = \{-1, 1\}$
- $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ 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 $f(x)$

Examples

- Least-square regression
- Nearest neighbors
- Decision trees
- Neural networks
- Logistic regression
- PLS
- SVM

Outline

1. Pattern recognition and regression
   - Empirical risk minimization
   - Feature selection
   - Shrinkage methods

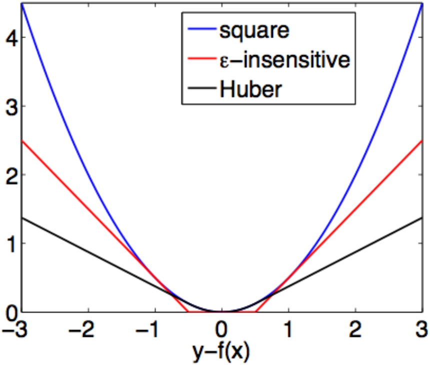
Probabilistic formalism

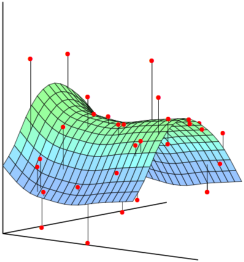
Risk

- P an (unknown) distribution on $\mathcal{X} \times \mathcal{Y}$.
- Observation: $\mathcal{S}_n = (X_i, Y_i)_{i=1,\ldots,n}$ i.i.d. random variables according to P.
- Loss function $\ell\left(f\left(\mathbf{x}\right), \mathbf{y}\right) \in \mathbb{R}$ small when $f\left(\mathbf{x}\right)$ is a good predictor for y
- Risk: $R(f) = \mathbf{E}l\left(f\left(X\right), Y\right)$.
- Estimator $\hat{f}_n : \mathcal{X} \rightarrow \mathcal{Y}$.
- Goal: small risk $R\left(\hat{f}_n\right)$.

Loss for regression

- Square loss : $\ell\left(f\left(\mathbf{x}\right),\mathbf{y}\right) = \left(f\left(\mathbf{x}\right) - \mathbf{y}\right)^2$
- ϵ-insensitive loss : $\ell\left(f\left(\mathbf{x}\right),\mathbf{y}\right) = \left(\mid f\left(\mathbf{x}\right) - \mathbf{y}\mid - \epsilon\right)_+$
- Huber loss : mixed quadratic/linear

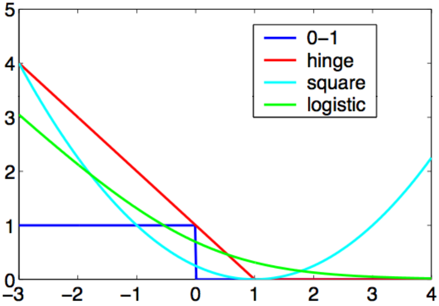


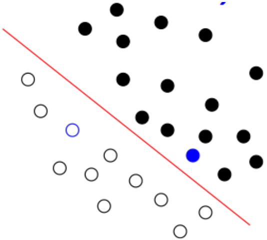


Loss for pattern recognition

Large margin classifiers

- For pattern recognition $\mathcal{Y} = \{-1, 1\}$
- Estimate a function $f : \mathcal{X} \to \mathbb{R}$.
- The margin of the function f for a pair $(\mathbf{x}, \mathbf{y})$ is: $\mathbf{y} f(\mathbf{x})$.
- The loss function is usually a decreasing function of the margin : $\ell\left(f\left(\mathbf{x}\right), \mathbf{y}\right) = \phi\left(\mathbf{y} f\left(\mathbf{x}\right)\right)$,





Empirical risk minimization (ERM)

ERM estimator

- $\mathcal{F}$ a class of candidate functions (e.g., linear functions)
- The empirical risk is:

$$R^n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(f\left(X_i\right), Y_i\right) .$$

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

$$\hat{f}_n = \arg\min_{f \in \mathcal{F}} R^n(f) .$$

Example: least squares linear regression

- $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \mathbb{R}$
- X the $n \times p$ matrix of patterns, y the $n \times 1$ vector of outputs
- Linear estimator:

$$f_\beta(\mathbf{x}) = \beta_0 + \sum_{i=1}^{p} x_i \beta_i$$

- ERM estimator for the square loss:

$$\begin{aligned} \min_{\beta \in \mathbb{R}^{p+1}} R^n(f_\beta) &= \sum_{i=1}^{n} \left(f_\beta\left(\mathbf{x}_i\right) - \mathbf{y}_i\right)^2 \\ &= (y - X\beta)^\top (y - X\beta) \end{aligned} \tag{1}$$

- Explicit solution:

$$\hat{\beta} = \left(X^\top X\right)^{-1} X^\top y \,.$$

Example: least squares linear regression

- $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \mathbb{R}$
- X the $n \times p$ matrix of patterns, y the $n \times 1$ vector of outputs
- Linear estimator:

$$f_\beta(\mathbf{x}) = \beta_0 + \sum_{i=1}^{p} x_i \beta_i$$

- ERM estimator for the square loss:

$$\begin{aligned} \min_{\beta \in \mathbb{R}^{p+1}} R^n(f_\beta) &= \sum_{i=1}^{n} \left(f_\beta\left(\mathbf{x}_i\right) - \mathbf{y}_i\right)^2 \\ &= (y - X\beta)^\top (y - X\beta) \end{aligned} \tag{1}$$

- Explicit solution:

$$\hat{\beta} = \left(X^\top X\right)^{-1} X^\top y \,.$$

Example: least squares linear regression

- $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \mathbb{R}$
- X the $n \times p$ matrix of patterns, y the $n \times 1$ vector of outputs
- Linear estimator:

$$f_\beta(\mathbf{x}) = \beta_0 + \sum_{i=1}^{p} x_i \beta_i$$

- ERM estimator for the square loss:

$$\begin{aligned} \min_{\beta \in \mathbb{R}^{p+1}} R^n(f_\beta) &= \sum_{i=1}^{n} \left(f_\beta\left(\mathbf{x}_i\right) - \mathbf{y}_i\right)^2 \\ &= (y - X\beta)^\top (y - X\beta) \end{aligned} \tag{1}$$

- Explicit solution:

$$\hat{\beta} = \left(X^\top X\right)^{-1} X^\top y \,.$$

Example: pattern recognition with the hinge loss

- $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \{-1, 1\}$
- Linear estimator:

$$f_\beta(\mathbf{x}) = sign(\mathbf{x}^\top \beta)$$

- ERM estimator for the hinge loss:

$$\min_{\beta \in \mathbb{R}^p} R^n(f_\beta) = \sum_{i=1}^{n} \max\left(0, 1 - \mathbf{y}_i f(\mathbf{x}_i)\right)$$

- Equivalent to the linear program

$$\begin{aligned} &\min \sum_{i=1}^{n} \xi_i \\ &\text{subject to} \quad \xi_i \geq 0, \xi_i \geq 1 - \mathbf{y}_i \mathbf{x}_i^\top \beta \end{aligned} \qquad (2)$$

Example: pattern recognition with the hinge loss

- $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \{-1, 1\}$
- Linear estimator:

$$f_\beta(\mathbf{x}) = sign(\mathbf{x}^\top \beta)$$

- ERM estimator for the hinge loss:

$$\min_{\beta \in \mathbb{R}^p} R^n(f_\beta) = \sum_{i=1}^{n} \max\left(0, 1 - \mathbf{y}_i f(\mathbf{x}_i)\right)$$

- Equivalent to the linear program

$$\begin{aligned} &\min \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad &\xi_i \geq 0, \xi_i \geq 1 - \mathbf{y}_i \mathbf{x}_i^\top \beta \end{aligned} \qquad (2)$$

Example: pattern recognition with the hinge loss

- $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \{-1, 1\}$
- Linear estimator:

$$f_\beta(\mathbf{x}) = sign(\mathbf{x}^\top \beta)$$

- ERM estimator for the hinge loss:

$$\min_{\beta \in \mathbb{R}^p} R^n(f_\beta) = \sum_{i=1}^n \max\left(0, 1 - \mathbf{y}_i f(\mathbf{x}_i)\right)$$

- Equivalent to the linear program

$$\min \sum_{i=1}^n \xi_i \qquad (2)$$

$$\text{subject to} \quad \xi_i \geq 0, \xi_i \geq 1 - \mathbf{y}_i \mathbf{x}_i^\top \beta$$

Other ERM methods : convex optimization

- For other losses, there is generally no explicit analytical formula for the solution
- However, if the loss function is convex in f, then we end up with a convex optimization problem that can usually be solved efficiently

Limits of ERM

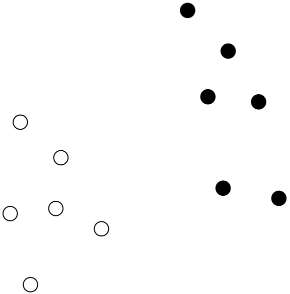
Unfortunately, the ERM estimator can be:

- ill-posed
- not statistically consistent (i.e., bad accuracy)

This is particularly the case in high dimension...

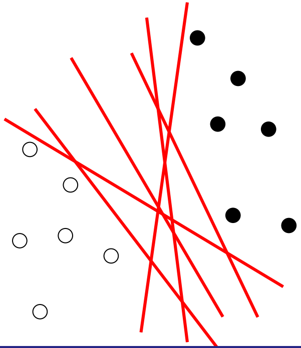
ERM is ill-posed in high dimension

- Suppose $n < p$
- Then $X^\top X$ is not invertible, so the least-square estimatore $\left(X^\top X\right)^{-1} X^\top y$ is not defined.
- More precisely, there are an infinite number of solution that minimize the empirical risk to 0.



ERM is ill-posed in high dimension

- Suppose $n < p$
- Then $X^\top X$ is not invertible, so the least-square estimatore $\left(X^\top X\right)^{-1} X^\top y$ is not defined.
- More precisely, there are an infinite number of solution that minimize the empirical risk to 0.

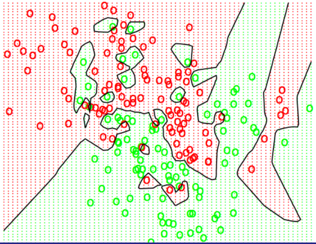


ERM is not consistent

- From the law of large numbers, for any $f \in \mathcal{F}$, the empirical risk converges to the true risk when the sample size increases:

$$\forall f \in \mathcal{F}, \quad R^n(f) \underset{n \to \infty}{\to} R(f)$$

- This suggest that minimizing $R^n(f)$ should give a good estimator of the minimizer of $R(f)$, but...
- Unfortunately it is not so simple! Vapnik in particular showed that this is only true if the "capacity" of $\mathcal{F}$ is not too large



Solution

Restrict the space of hypothesis

- A solution to work in high dimension is to restrict the space of functions $\mathcal{F}$ over which ERM is applied:

$$\min_{f \in \mathcal{F}} R^n(f)$$

- We will focus on linear functions $f(\mathbf{x}) = \mathbf{x}^\top \beta$, and put various constraints on β
  - Restrict the number of non-zero components (feature selection)
  - Restrict the size of β, for some norm (shrinkage methods)

The bias / variance trade-off

- When $\mathcal{F}$ is small, the ERM principle is efficient to find a good solution among $\mathcal{F}$, i.e.:

$$R(\hat{f}) \sim \inf_{f \in \mathcal{F}} R(f)$$

We say that the variance is small.

- When $\mathcal{F}$ is large, then the best solution in $\mathcal{F}$ is close to the best solution possible:

$$\inf_{f \in \mathcal{F}} R(f) \sim \inf_{f} R(f)$$

We say that the bias is small.

- A good estimator should have a small bias and small variance
- Therefore it is important to put prior knowledge on the design of $\mathcal{F}$, to make it as small as possible (small variance) but make sure it contains good functions (small bias)

Outline

1. Pattern recognition and regression
   - Empirical risk minimization
   - Feature selection
   - Shrinkage methods

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 restricting $\mathcal{F}$, we increase the bias but decrease the variance. This should be helpful in particular in high dimension, where bias is low and variance is large.
  - Interpretation: with a large number of predictors, we often would like to determine a smaller subset that exhibit the strongest effects.
- Of course, this is particularly relevant if we believe that there exist good predictors which are sparse (prior knowledge).

Best subset selection

- 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...)

Pros

Simple, scalable, good empirical success

Cons

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

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

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

Wrapper methods

The idea

- A greedy approach to

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

- For a given set of seleted features, we know how to minimize $R^n(f)$
- 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

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

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

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

1. Pattern recognition and regression
   - Empirical risk minimization
   - Feature selection
   - Shrinkage methods

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.

Motivation

- Accuracy: as for feature selection, we reduce $\mathcal{F}$, hence reduce variance
- Inclusion of prior knowledge: $\Omega(\beta)$ is the place to put your prior knowledge to reduce the bias
- Computational efficiency: if $R(f)$ and $\Omega(\beta)$ are convex, then we obtain a convex optimization problem that can often be solved exactly and efficiently. It is then equivalent to:

$$\min R(f_\beta) + \lambda\Omega(\beta)$$

Ridge regression

- Take $\Omega(\beta) = \sum_{i=1}^{p} \beta_i^2 = \| \beta \|_2^2$.
- Constrained least-square:

$$\begin{aligned} \min_{\beta \in \mathbb{R}^{p+1}} R^n(f_\beta) &= \sum_{i=1}^{n} \left(f_\beta\left(\mathbf{x}_i\right) - \mathbf{y}_i \right)^2 + \lambda \sum_{i=1}^{p} \beta_i^2 \\ &= (y - X\beta)^\top (y - X\beta) + \lambda \beta^\top \beta \,. \end{aligned} \tag{3}$$

- Explicit solution:

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

Ridge regression

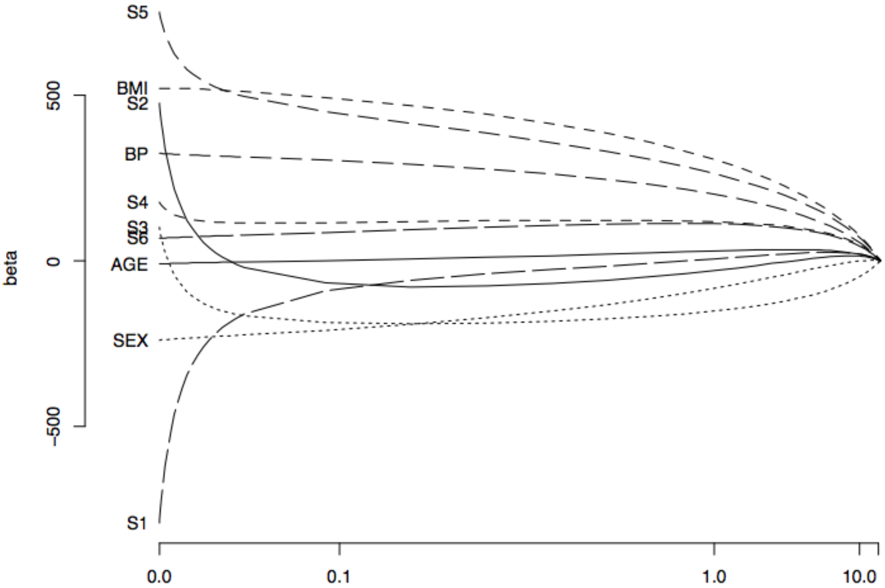
- Take $\Omega(\beta) = \sum_{i=1}^{p} \beta_i^2 = \| \beta \|_2^2$.
- Constrained least-square:

$$\begin{aligned} \min_{\beta \in \mathbb{R}^{p+1}} R^n(f_\beta) &= \sum_{i=1}^{n} \left(f_\beta\left(\mathbf{x}_i\right) - \mathbf{y}_i\right)^2 + \lambda \sum_{i=1}^{p} \beta_i^2 \\ &= (y - X\beta)^\top (y - X\beta) + \lambda \beta^\top \beta \,. \end{aligned} \tag{3}$$

- Explicit solution:

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

Ridge regression example



LASSO regression

- Take $\Omega(\beta) = \sum_{i=1}^{p} |\, \beta_i \,| = \|\, \beta \,\|_1$.
- Constrained least-square:

$$\min_{\beta \in \mathbb{R}^{p+1}} R^n(f_\beta) = \sum_{i=1}^{n} \left(f_\beta\left(\mathbf{x}_i\right) - \mathbf{y}_i\right)^2 + \lambda \sum_{i=1}^{p} |\, \beta_i \,| \tag{4}$$

- 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)

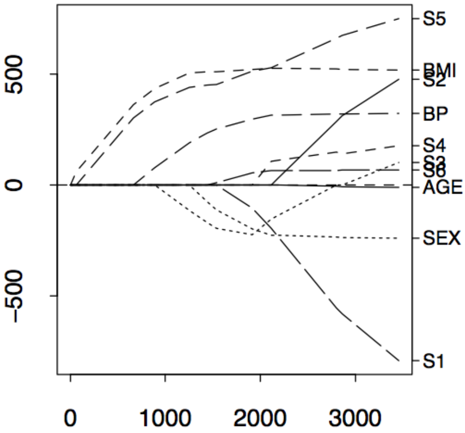
LASSO regression

- Take $\Omega(\beta) = \sum_{i=1}^{p} \mid \beta_i \mid = \| \beta \|_1$.
- Constrained least-square:

$$\min_{\beta \in \mathbb{R}^{p+1}} R^n(f_\beta) = \sum_{i=1}^{n} \left(f_\beta \left(\mathbf{x}_i \right) - \mathbf{y}_i \right)^2 + \lambda \sum_{i=1}^{p} \mid \beta_i \mid \tag{4}$$

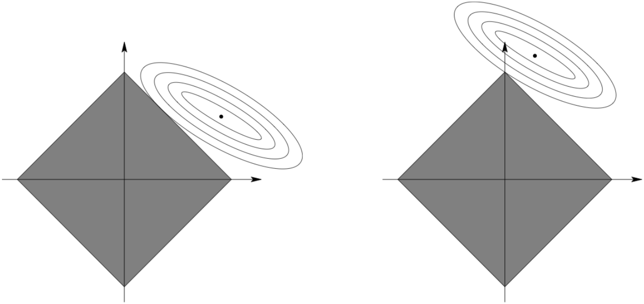
- No explicit solution, but this is just a quadratic program.
- LARS (Efron et al., 2004) provides a fast algorithm to compute the solution for all λ's simultaneously (regularization path)

LASSO regression example



Why LASSO leads to sparse solutions

Geometric interpretation with $p = 2$



Summary

- ERM is a popular induction principle, which underlies many algorithms for regression and pattern recognition
- In high dimension we must be careful
- Constrained ERM provides a coherent and nice framework

$$\min_f R^n(f) \quad \text{s.t.} \quad \Omega(f) < \gamma$$

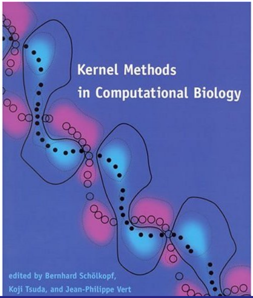
- A strong constraint (γ small) reduces the variance but increases the bias
- The key idea to learn in high dimension is to use prior knowledge to design $\Omega(f)$ to ensure a small bias.

Outline

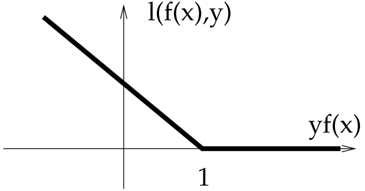
1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Motivation

- SVM is just a particular constrained ERM algorithm
- It became extremely popular in many applied fields over the last 10 years
- It allows to extend considerably the hypothesis space $\mathcal{F}$ beyond linear functions, thanks to the use of positive definite kernels (
- It also allows to extend most linear methods to structured objects, e.g., strings and graphs.


Kernel Methods
in Computational Biology
edited by Bernhard Schölkopf,
Koji Tsuda, and Jean-Philippe Vert

Linear SVM for pattern recognition



- $\mathcal{X} = \mathbb{R}^p, \mathcal{Y} = \{-1, 1\}$
- Linear classifiers:

$$f_\beta(\mathbf{x}) = \mathbf{x}^\top \beta .$$

- The loss function is the hinge loss:

$$\phi_{\text{hinge}}(u) = \max(1-u, 0) = \begin{cases} 0 & \text{if } u \geq 1, \\ 1-u & \text{otherwise.} \end{cases}$$

Linear SVM for pattern recognition

- SVM solve the problem:

$$\min_{f_\beta \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \phi_{\text{hinge}}\left(\mathbf{y}_i f_\beta\left(\mathbf{x}_i\right)\right) \quad \text{s.t.} \quad \| \beta \|_2^2 \leq \gamma \,.$$

- Equivalently

$$\min_{f_\beta \in \mathcal{F}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \phi_{\text{hinge}}\left(\mathbf{y}_i f_\beta\left(\mathbf{x}_i\right)\right) + \lambda \| \beta \|_2^2 \right\} \,.$$

Dual problem

- This is a convex optimization problem. It is equivalent to the following dual problem (good exercice to derive it):

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^d} 2 \sum_{i=1}^{n} \alpha_i \mathbf{y}_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j \,,$$

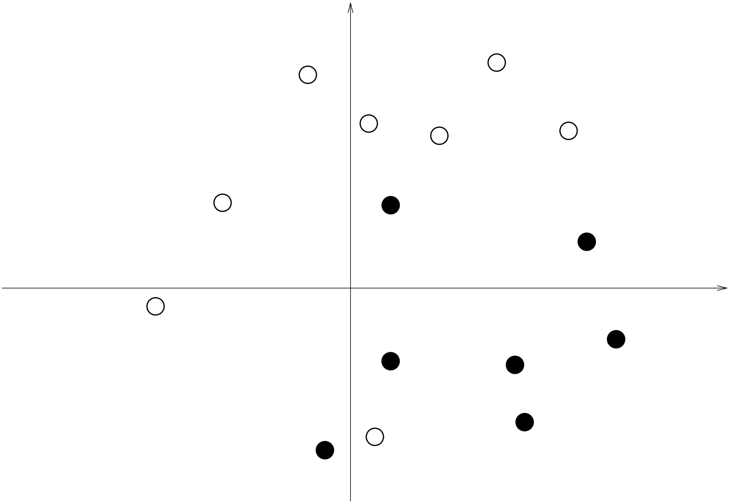
subject to:

$$0 \leq y_i \alpha_i \leq \frac{1}{2\lambda n}, \quad \text{for } i = 1, \ldots, n \,.$$

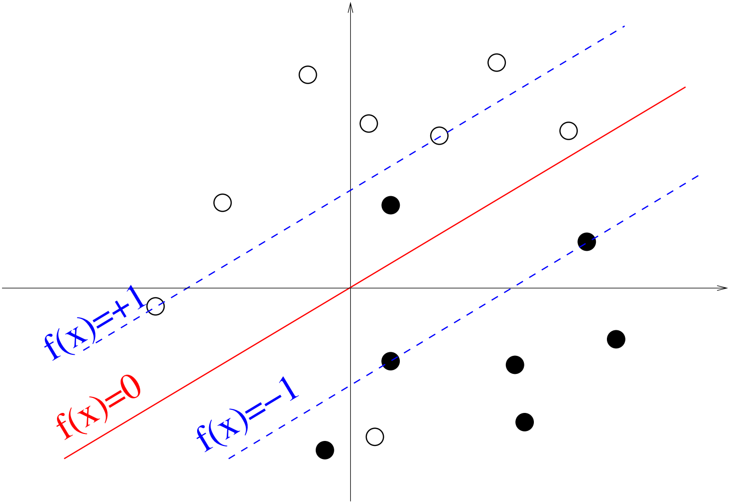
- If α solves this problem, we recover the solution of the primal problem by:

$$f_\beta \left(\mathbf{x} \right) = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^\top \mathbf{x} \,.$$

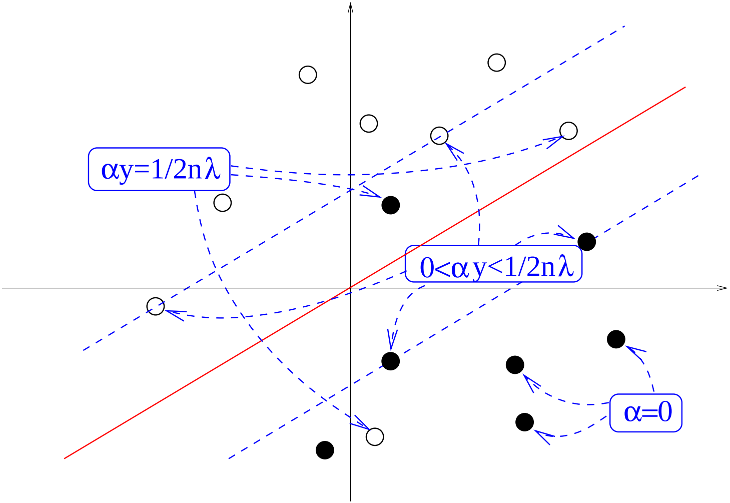
$f_\beta(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^\top \mathbf{x}$



$f_\beta(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^\top \mathbf{x}$



$f_\beta(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^\top \mathbf{x}$



Support vectors

Consequence of KKT conditions

- The training points with $\alpha_i \neq 0$ are called support vectors.
- Only support vectors are important for the classification of new points:

$$\forall \mathbf{x} \in \mathcal{X}, \quad f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^\top \mathbf{x} = \sum_{i \in SV} \alpha_i \mathbf{x}_i^\top \mathbf{x} \,,$$

where SV is the set of support vectors.

Consequences

- The solution is sparse in α, leading to fast algorithms for training (use of decomposition methods).
- The classification of a new point only involves kernel evaluations with support vectors (fast).

An important remark

- Training a SVM means finding $\boldsymbol{\alpha} \in \mathbb{R}^n$ which solves:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^d} 2 \sum_{i=1}^{n} \alpha_i \mathbf{y}_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j \,,$$

subject to:

$$0 \leq y_i \alpha_i \leq \frac{1}{2\lambda n}, \quad \text{for } i = 1, \ldots, n \,.$$

- The prediction for a new point $\mathbf{x}$ is the sign of

$$f\left(\mathbf{x}\right) = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^\top \mathbf{x} \,.$$

Kernels

- Let the kernel function:

$$K\left(\mathbf{x},\mathbf{x}'\right)=\mathbf{x}^\top\mathbf{x}'\,.$$

- Training a SVM means finding $\boldsymbol{\alpha}\in\mathbb{R}^n$ which solves:

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^d} 2\sum_{i=1}^n \alpha_i \mathbf{y}_i - \sum_{i,j=1}^n \alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j)\,,$$

subject to:

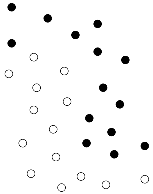
$$0\leq y_i\alpha_i\leq\frac{1}{2\lambda n},\quad \text{for } i=1,\ldots,n\,.$$

- The prediction for a new point **x** is the sign of

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

Extension

- Let $\mathcal{X}$ be any set, and
$$\Phi : \mathcal{X} \rightarrow \mathcal{H}$$
an embedding in a Hilbert space ($\mathcal{H} = \mathbb{R}^p$ with p finite or infinite)
- Then we can train and use a SVM implicitly in $\mathcal{H}$ if we are able to compute the kernel:
$$K\left(\mathbf{x}, \mathbf{x}'\right) = \phi(\mathbf{x})^\top \Phi(\mathbf{x}') \,.$$



Extension

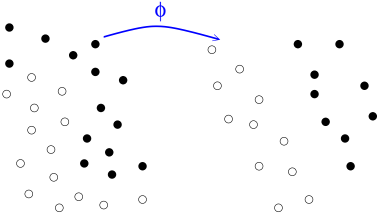
- Let $\mathcal{X}$ be any set, and

$$\Phi : \mathcal{X} \rightarrow \mathcal{H}$$

an embedding in a Hilbert space ($\mathcal{H} = \mathbb{R}^p$ with p finite or infinite)

- Then we can train and use a SVM implicitly in $\mathcal{H}$ if we are able to compute the kernel:

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \phi(\mathbf{x})^\top \Phi(\mathbf{x}')\,.$$



Extension

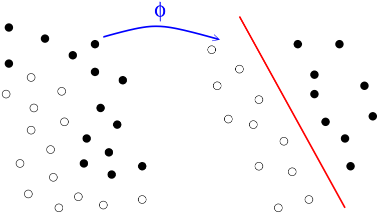
- Let $\mathcal{X}$ be any set, and

$$\Phi : \mathcal{X} \to \mathcal{H}$$

an embedding in a Hilbert space ($\mathcal{H} = \mathbb{R}^p$ with p finite or infinite)

- Then we can train and use a SVM implicitly in $\mathcal{H}$ if we are able to compute the kernel:

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \phi(\mathbf{x})^\top \Phi(\mathbf{x}') \,.$$



Extension

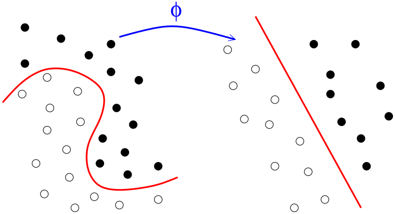
- Let $\mathcal{X}$ be any set, and

$$\Phi : \mathcal{X} \to \mathcal{H}$$

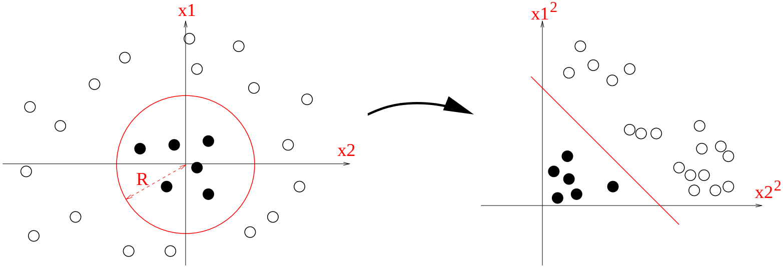
an embedding in a Hilbert space ($\mathcal{H} = \mathbb{R}^p$ with p finite or infinite)

- Then we can train and use a SVM implicitly in $\mathcal{H}$ if we are able to compute the kernel:

$$K\left(\mathbf{x},\mathbf{x}'\right) = \phi(\mathbf{x})^\top \Phi(\mathbf{x}')\,.$$



Example: polynomial kernel



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

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

Positive Definite (p.d.) Kernels

Definition

A positive definite (p.d.) kernel 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 $(a_1, a_2, \ldots, a_N) \in \mathbb{R}^N$:

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

Characterization of inner products

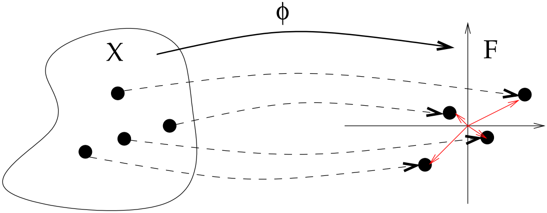
Theorem (Aronszajn, 1950)

K is a p.d. kernel on the set $\mathcal{X}$ if and only if there exists a Hilbert space $\mathcal{H}$ and a mapping

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

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

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



Examples

Classical kernels for vectors ($\mathcal{X} = \mathbb{R}^p$) include:

- The linear kernel

$$K_{lin}\left(\mathbf{x},\mathbf{x}'\right) = \mathbf{x}^\top\mathbf{x}' \ .$$

- The polynomial kernel

$$K_{poly}\left(\mathbf{x},\mathbf{x}'\right) = \left(\mathbf{x}^\top\mathbf{x}' + a\right)^d \ .$$

- The Gaussian RBF kernel:

$$K_{Gaussian}\left(\mathbf{x},\mathbf{x}'\right) = \exp\left(-\frac{\parallel \mathbf{x} - \mathbf{x}' \parallel^2}{2\sigma^2}\right) \ .$$

Summary

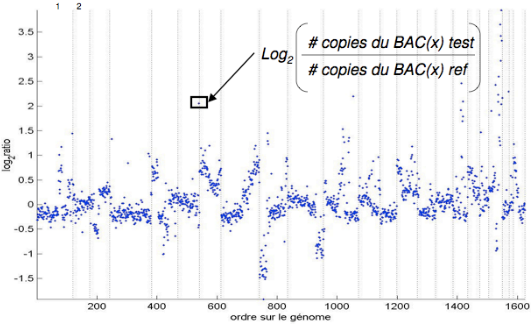
- Kernels allow to apply linear methods in a much larger space ($\mathcal{F}$ increases, bias decreases) without changing the algorithm
- This can be generalized to any ERM constrained by the Euclidean norm (kernel ridge regression ...)
- Allows to infer nonlinear functions
- Allows to work with non-vector space (see later: strings, graphs, ...)
- Include prior knowledge in the kernel

Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Motivation

- Comparative genomic hybridization (CGH) data measure the DNA copy number along the genome
- Very useful, in particular in cancer research
- Can we classify CGH arrays for diagnosis or prognosis purpose?



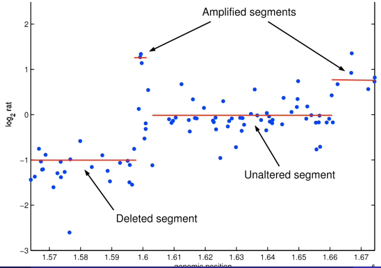
Jain et al. Genome research 2002 12:325-332

Prior knowledge

- Let **x** be a CGH profile
- We focus on linear classifiers, i.e., the sign of :

$$f(\mathbf{x}) = \mathbf{x}^\top \beta \, .$$

- We expect β to be
  - sparse : only a few positions should be discriminative
  - piecewise constant : within a region, all probes should contribute equally

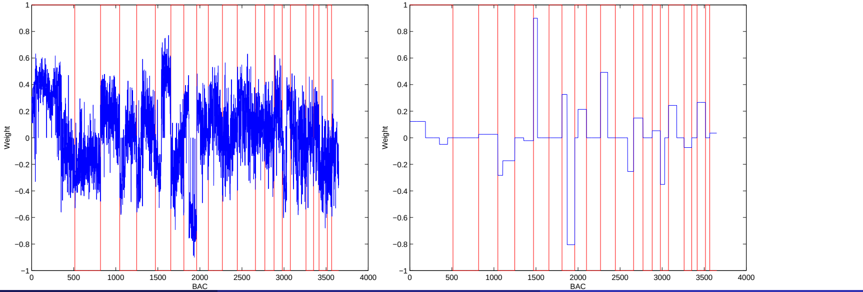


Example: CGH array classification

A solution (Rapaport et al., 2008)

$$\Omega_{fusedlasso}(\beta) = \sum_{i} |\beta_i| + \sum_{i \sim j} |\beta_i - \beta_j| .$$

- Good performance on diagnosis for bladder cancer, and prognosis for melanoma.
- More interpretable classifiers



Outline

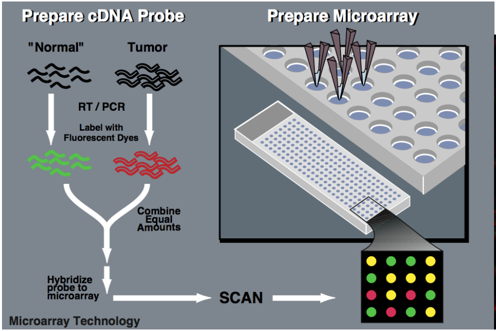
1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. **Classification of expression data**
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

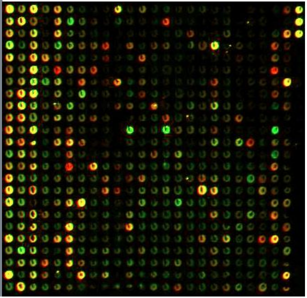
Outline

4 Classification of expression data

- Motivation
- Using gene networks as prior knowledge
- Application

Tissue profiling with DNA chips

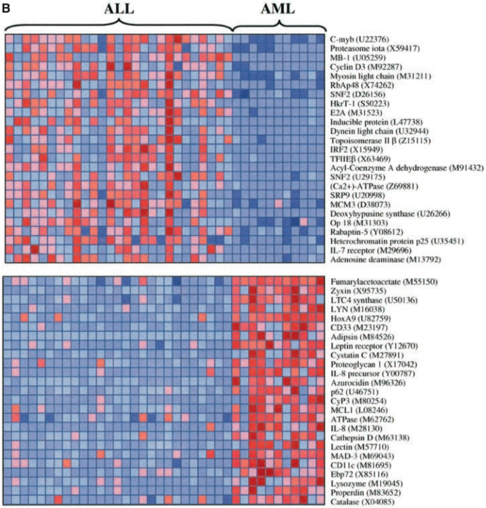




Data

- Gene expression measures for more than 10*k* genes
- Measured typically on less than 100 samples of two (or more) different classes (e.g., different tumors)

Tissue classification from microarray data



Goal

- Design a classifier to automatically assign a class to future samples from their expression profile
- Interpret biologically the differences between the classes

Linear classifiers

The approach

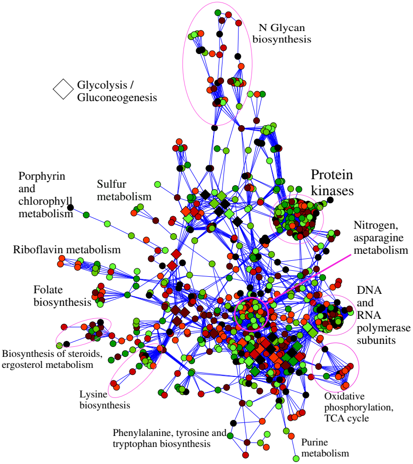
- Each sample is represented by a vector $x = (x_1, \ldots, x_p)$ where $p > 10^5$ is the number of probes
- Classification: given the set of labeled sample, learn a linear decision function:
  $$f_\beta(x) = \sum_{i=1}^{p} \beta_i x_i + \beta_0 \ ,$$
  that is positive for one class, negative for the other
- Interpretation: the weight β_i quantifies the influence of gene i for the classification
- We must use prior knowledge for this small n large p problem.

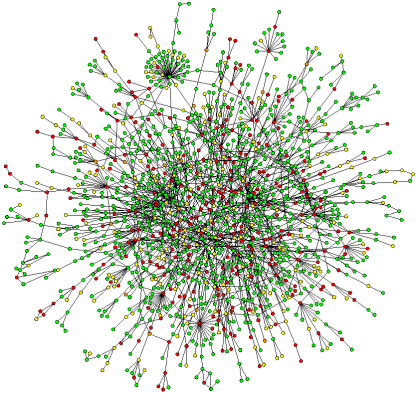
Outline

4 Classification of expression data

- Motivation
- Using gene networks as prior knowledge
- Application

Gene networks


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

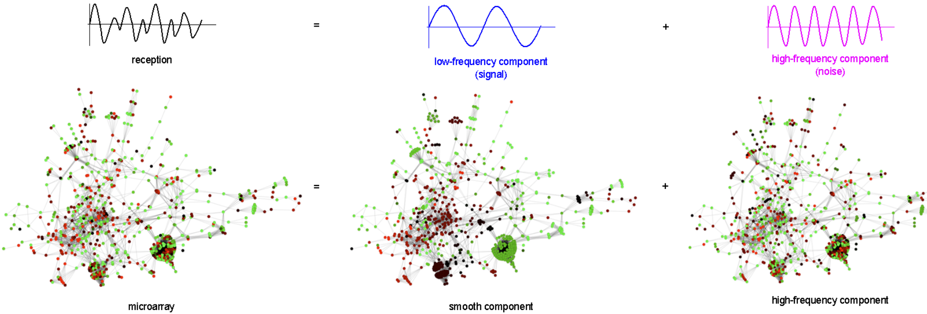


Gene network interpretation

Motivation

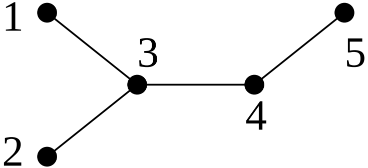
- Basic biological functions usually involve the coordinated action of several proteins:
  - Formation of protein complexes
  - Activation of metabolic, signalling or regulatory pathways
- Many pathways and protein-protein interactions are already known
- Hypothesis: the weights of the classifier should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

The idea



1. Use the gene network to extract the "important information" in gene expression profiles by Fourier analysis on the graph
2. Learn a linear classifier on the smooth components

Notations

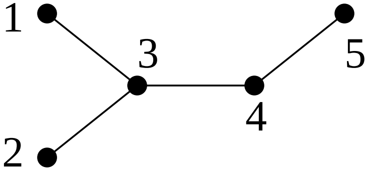


$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

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 the graph:

- *For any $f : \mathcal{X} \to \mathbb{R}$,*

$$f^\top L f = \sum_{i \sim j} \left(f\left(\mathbf{x}_i\right) - f\left(\mathbf{x}_j\right) \right)^2$$

- *L is a symmetric positive semi-definite matrix*
- *0 is an eigenvalue with multiplicity equal to the number of connected components.*

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

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

Proof: eigenstructure of L

- L is symmetric because A and D are symmetric.
- For any $f \in \mathbb{R}^m$, $f^\top L f \geq 0$, 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 $\sum_{i \sim j} \left(f(\mathbf{x}_i) - f\left(\mathbf{x}_j\right) \right)^2 = 0$,
  iff $f(\mathbf{x}_i) = f\left(\mathbf{x}_j\right)$ when $i \sim j$,
  iff f is constant (because the graph is connected).

Fourier basis

Definition

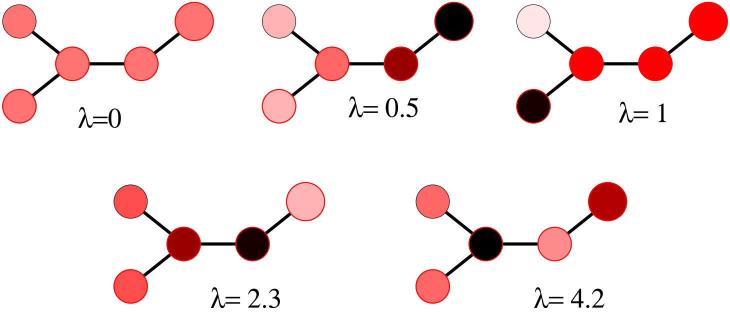
- The eigenvectors $e_1, \ldots, e_n$ of L with eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_n$ form a basis called Fourier basis
- For any $f : V \to \mathbb{R}$, the Fourier transform of f is the vector $\hat{f} \in \mathbb{R}^n$ defined by:

$$\hat{f}_i = f^\top e_i \,, \quad i = 1, \ldots, n.$$

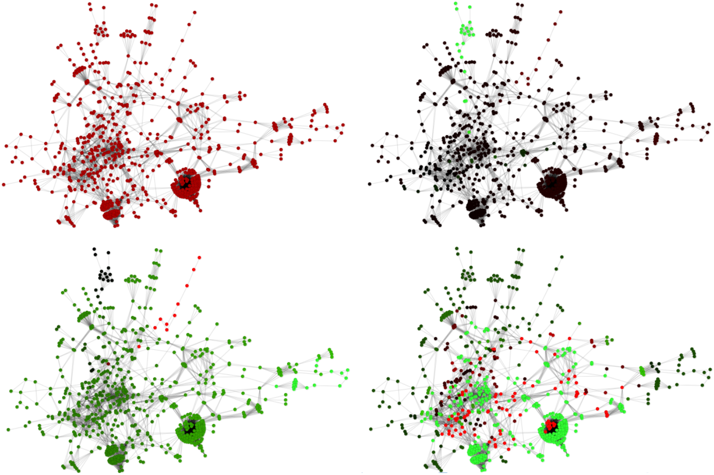
- Obviously the inverse Fourier formula holds:

$$f = \sum_{i=1}^{n} \hat{f}_i e_i \,.$$

Fourier basis



Fourier basis



Smoothing operator

Definition

- Let $\phi : \mathbb{R}^+ \to \mathbb{R}^+$ be non-increasing.
- A smoothing operator S_ϕ transform a function $f : V \to \mathbb{R}$ into a smoothed version:

$$S_\phi(f) = \sum_{i=1}^{n} \hat{f}_i \phi(\lambda_i) e_i \, .$$

Smoothing operators

Examples

- Identity operator ($S_\phi(f) = f$):

$$\phi(\lambda) = 1\,, \quad \forall \lambda$$

- Low-pass filter:

$$\phi(\lambda) = \begin{cases} 1 & \text{if } \lambda \leq \lambda^*\,, \\ 0 & \text{otherwise.} \end{cases}$$

- Attenuation of high frequencies:

$$\phi(\lambda) = \exp(-\beta\lambda)\,.$$

Smoothing operators

Examples

- Identity operator ($S_\phi(f) = f$):

$$\phi(\lambda) = 1\,, \quad \forall\lambda$$

- Low-pass filter:

$$\phi(\lambda) = \begin{cases} 1 & \text{if } \lambda \leq \lambda^*\,, \\ 0 & \text{otherwise.} \end{cases}$$

- Attenuation of high frequencies:

$$\phi(\lambda) = \exp(-\beta\lambda)\,.$$

Smoothing operators

Examples

- Identity operator ($S_\phi(f) = f$):

$$\phi(\lambda) = 1\,, \quad \forall \lambda$$

- Low-pass filter:

$$\phi(\lambda) = \begin{cases} 1 & \text{if } \lambda \leq \lambda^*\,, \\ 0 & \text{otherwise.} \end{cases}$$

- Attenuation of high frequencies:

$$\phi(\lambda) = \exp(-\beta\lambda)\,.$$

Supervised classification and regression

Working with smoothed profiles

- Classical methods for linear classification and regression with a ridge penalty solve:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} l\left(\beta^\top f_i, y_i\right) + \lambda \beta^\top \beta \,.$$

- Applying these algorithms on the smooth profiles means solving:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} l\left(\beta^\top S_\phi(f_i), y_i\right) + \lambda \beta^\top \beta \,.$$

Smooth solution

Lemma

This is equivalent to:

$$\min_{v \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} l\left(v^\top f_i, y_i\right) + \lambda \sum_{i=1}^{p} \frac{\hat{v}_i^2}{\phi(\lambda_i)},$$

hence the linear classifier v is smooth.

Proof

- Let $v = \sum_{i=1}^{n} \phi(\lambda_i) e_i e_i^\top \beta$, then

$$\beta^\top S_\phi(f_i) = \beta^\top \sum_{i=1}^{n} \hat{f}_i \phi(\lambda_i) e_i = f^\top v.$$

- Then $\hat{v}_i = \phi(\lambda_i) \hat{\beta}_i$ and $\beta^\top \beta = \sum_{i=1}^{n} \frac{\hat{v}_i^2}{\phi(\lambda_i)^2}$.

Smooth solution

Lemma

This is equivalent to:

$$\min_{v \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} l\left(v^\top f_i, y_i\right) + \lambda \sum_{i=1}^{p} \frac{\hat{v}_i^2}{\phi(\lambda_i)} \,,$$

hence the linear classifier v is smooth.

Proof

- Let $v = \sum_{i=1}^{n} \phi(\lambda_i) e_i e_i^\top \beta$, then

$$\beta^\top S_\phi(f_i) = \beta^\top \sum_{i=1}^{n} \hat{f}_i \phi(\lambda_i) e_i = f^\top v \,.$$

- Then $\hat{v}_i = \phi(\lambda_i) \hat{\beta}_i$ and $\beta^\top \beta = \sum_{i=1}^{n} \frac{\hat{v}_i^2}{\phi(\lambda_i)^2}$.

Kernel methods

Smoothing kernel

Kernel methods (SVM, kernel ridge regression..) only need the inner product between smooth profiles:

$$
\begin{aligned}
K(f,g) &= S_\phi(f)^\top S_\phi(g) \\
&= \sum_{i=1}^{n} \hat{f}_i \hat{g}_i \phi(\lambda_i)^2 \\
&= f^\top \left(\sum_{i=1}^{n} \phi(\lambda_i)^2 e_i e_i^\top \right) g \\
&= f^\top K_\phi g \,,
\end{aligned}
\tag{5}
$$

with

$$K_\phi = \sum_{i=1}^{n} \phi(\lambda_i)^2 e_i e_i^\top \,.$$

Examples

- For $\phi(\lambda) = \exp(-t\lambda)$, we recover the diffusion kernel:

$$K_\phi = \exp_M(-2tL)\,.$$

- For $\phi(\lambda) = 1/\sqrt{1+\lambda}$, we obtain

$$K_\phi = (L+I)^{-1}\,,$$

and the penalization is:

$$\sum_{i=1}^{n} \frac{\hat{v}_i^2}{\phi(\lambda_i)} = v^\top (L+I)\, v = \|\, v \,\|_2^2 + \sum_{i\sim j}(v_i - v_j)^2\,.$$

Examples

- For $\phi(\lambda) = \exp(-t\lambda)$, we recover the diffusion kernel:

$$K_\phi = \exp_M(-2tL)\,.$$

- For $\phi(\lambda) = 1/\sqrt{1+\lambda}$, we obtain

$$K_\phi = (L+I)^{-1}\ ,$$

and the penalization is:

$$\sum_{i=1}^{n} \frac{\hat{v}_i^2}{\phi(\lambda_i)} = v^\top (L+I)\, v = \|\, v\,\|_2^2 + \sum_{i\sim j} (v_i - v_j)^2\,.$$

Outline

4 Classification of expression data

- Motivation
- Using gene networks as prior knowledge
- Application

Data

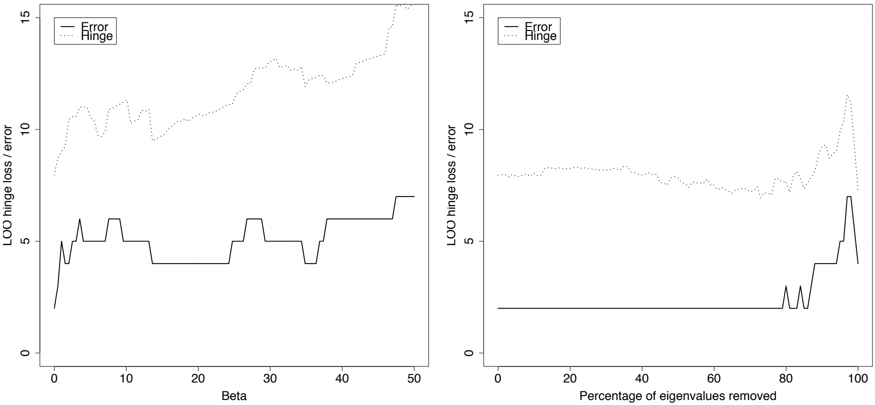
Expression

- Study the effect of low irradiation doses on the yeast
- 12 non irradiated vs 6 irradiated
- Which pathways are involved in the response at the transcriptomic level?

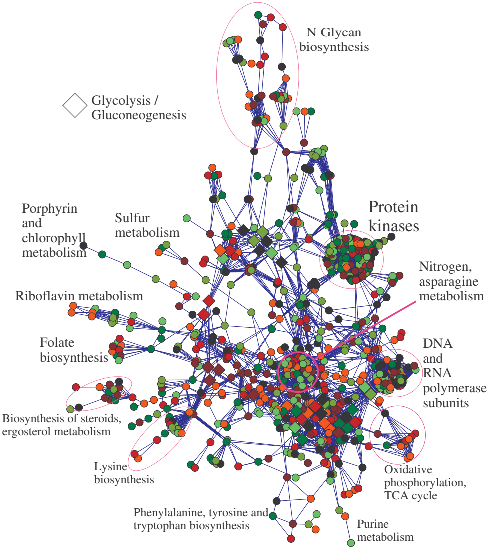
Graph

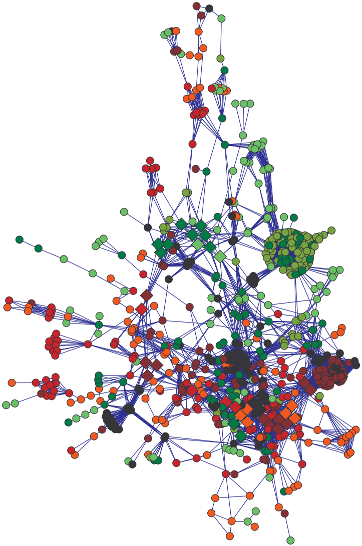
- KEGG database of metabolic pathways
- Two genes are connected is they code for enzymes that catalyze successive reactions in a pathway (metabolic gene network).
- 737 genes, 4694 vertices.

Classification performance

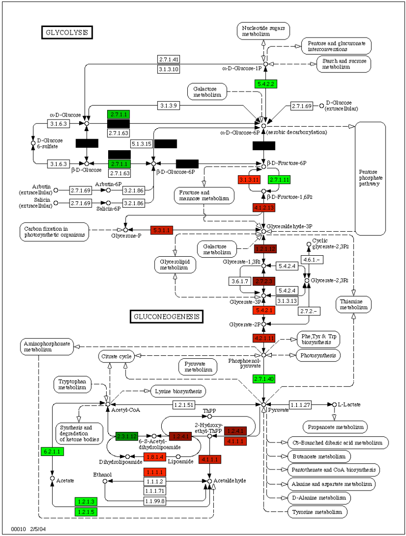


Classifier

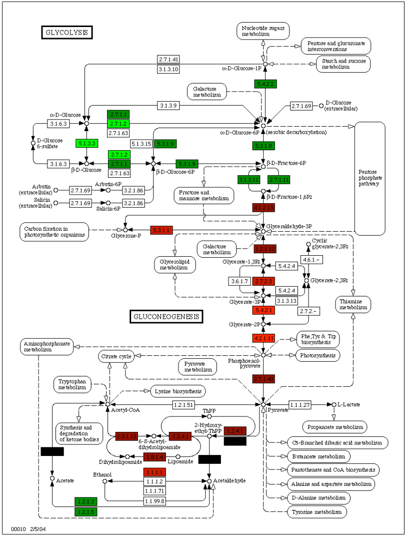

N Glycan biosynthesis
◇ Glycolysis / Gluconeogenesis
Porphyrin and chlorophyll metabolism
Sulfur metabolism
Protein kinases
Nitrogen, asparagine metabolism
Riboflavin metabolism
Folate biosynthesis
DNA and RNA polymerase subunits
Biosynthesis of steroids, ergosterol metabolism
Lysine biosynthesis
Oxidative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



Classifier



a)



b)

Outline

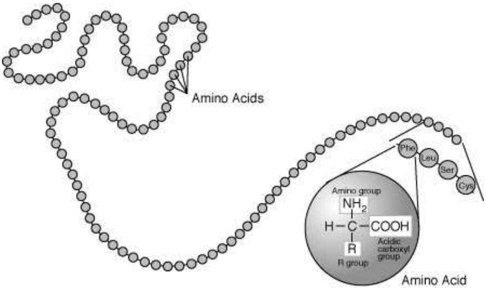
1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Outline

5 Classification of biological sequences

- Motivation
- Feature space approach
- Using generative models
- Derive from a similarity measure
- Application: remote homology detection

Proteins



A : Alanine	V : Valine	L : Leucine
F : Phenylalanine	P : Proline	M : Méthionine
E : Acide glutamique	K : Lysine	R : Arginine
T : Threonine	C : Cysteine	N : Asparagine
H : Histidine	V : Thyrosine	W : Tryptophane
I : Isoleucine	S : Sérine	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..
  ...

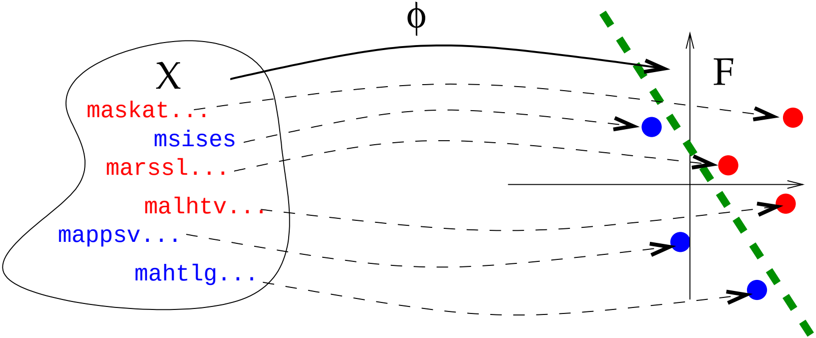
Goal

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

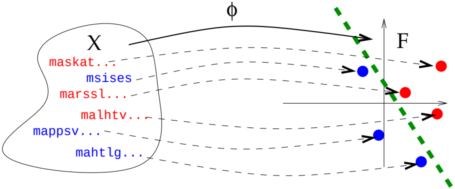
Supervised classification with vector embedding

The idea

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



Example: support vector machine



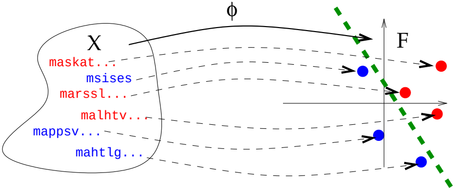
SVM algorithm

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \Phi(x_i)^\top \Phi(x)\right) ,$$

where $\alpha_1, \ldots, \alpha_n$ solve, under the constraints $0 \leq \alpha_i \leq C$:

$$\min_{\alpha}\left(\frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n}\alpha_i\alpha_j y_i y_j \Phi(x_i)^\top \Phi(x_j) - \sum_{i=1}^{n}\alpha_i\right) .$$

Explicit vector embedding



Difficulties

- How to define the mapping $\Phi : \mathcal{X} \rightarrow \mathbb{R}^p$?
- No obvious vector embedding for strings in general.
- How to include prior knowledge about the strings (grammar, probabilistic model...)?

Implicit vector embedding with kernels

The kernel trick

- Many algorithms just require inner products of the embeddings
- We call it a kernel between strings:

$$K(x, x') \triangleq \Phi(x)^\top \Phi(x')$$

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)

Implicit vector embedding with kernels

The kernel trick

- Many algorithms just require inner products of the embeddings
- We call it a kernel between strings:

$$K(x, x') \triangleq \Phi(x)^\top \Phi(x')$$

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
- Derive a kernel from a generative model
  - Fisher kernel
  - Mutual information kernel
  - Marginalized kernel
- Derive a kernel from a similarity measure
  - Local alignment 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

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

5 Classification of biological sequences

- Motivation
- Feature space approach
- Using generative models
- Derive from a similarity measure
- Application: remote homology detection

Vector embedding for strings

The idea

Represent each sequence **x** by a fixed-length numerical vector $\Phi(\mathbf{x}) \in \mathbb{R}^p$. 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)

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

Vector embedding for strings

The idea

Represent each sequence $\mathbf{x}$ by a fixed-length numerical vector $\Phi(\mathbf{x}) \in \mathbb{R}^p$. 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.,

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

where $\Phi_u\left(\mathbf{x}\right)$ 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

- The 3-spectrum of

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

  is:

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

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

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

- This is formally a sum over $|\mathcal{A}|^k$ terms, but at most $|\mathbf{x}| - k + 1$ terms are non-zero in $\Phi(\mathbf{x})$

Substring indexation in practice

- 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 $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$
- 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.

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:

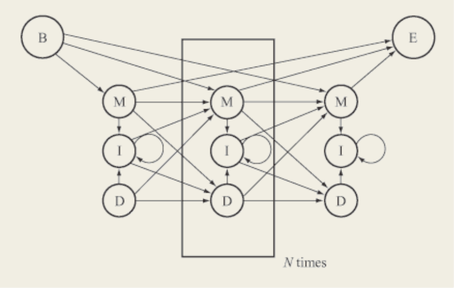
- Motif kernels (Logan et al., 2001): the dictionary is a library of motifs, the similarity function is a matching function
- Pairwise kernel (Liao & Noble, 2003): the dictionary is the training set, the similarity is a classical measure of similarity between sequences.

Outline

5. Classification of biological sequences
   - Motivation
   - Feature space approach
   - Using generative models
   - Derive from a similarity measure
   - Application: remote homology detection

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 (under several assumptions).

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
- $K\left(\mathbf{x},\mathbf{x}'\right) =< \phi\left(\mathbf{x}\right), \phi\left(\mathbf{x}'\right) >_{L_2(w)}$ with

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

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:

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

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

Context-tree model

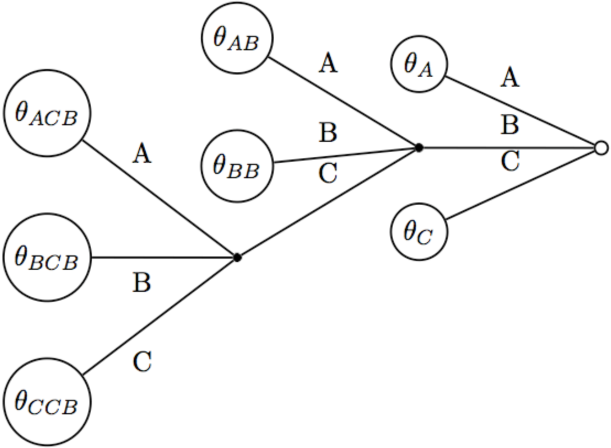
Definition

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

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

- $\mathcal{D}$ 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 $\Phi_{\mathcal{Z}} : \mathcal{Z} \rightarrow \mathcal{H}$ such that:

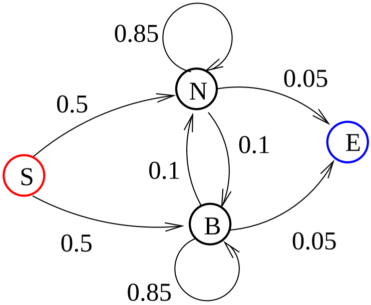
$$K_{\mathcal{Z}}\left(\mathbf{z}, \mathbf{z}'\right) = \left\langle \Phi_{\mathcal{Z}}\left(\mathbf{z}\right), \Phi_{\mathcal{Z}}\left(\mathbf{z}'\right) \right\rangle_{\mathcal{H}} \ .$$

- 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 $\mathcal{X}$. □

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):

NNNNNBBBBBBBBBNNNNNNNNNNNBBBBBB
1001011101111010010111001111011

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:

$$\mathbf{z} = 1001011101111010010111001111011,$$
$$\mathbf{z}' = 0011010110011111011010111101100101,$$

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

1-spectrum marginalized kernel on observed data

- The marginalized kernel for observed data is:

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

with

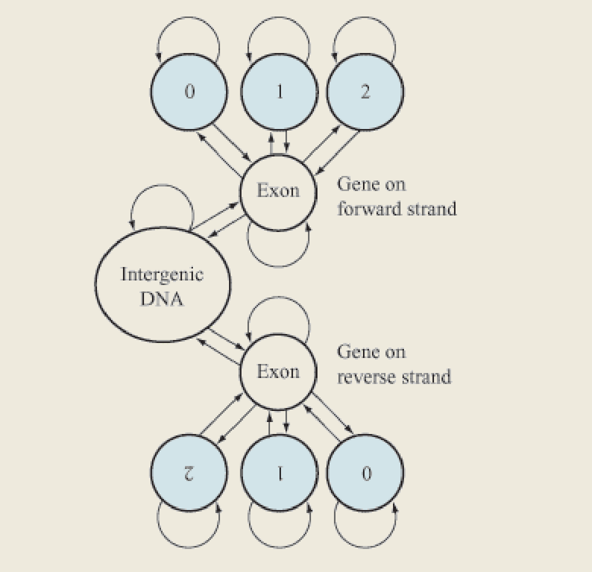
$$
\Phi_{a,s}\left(\mathbf{x}\right) = \sum_{\mathbf{y}\in\mathcal{S}^*} P\left(\mathbf{y}|\mathbf{x}\right) n_{a,s}\left(\mathbf{x},\mathbf{y}\right)
$$

Computation of the 1-spectrum marginalized kernel

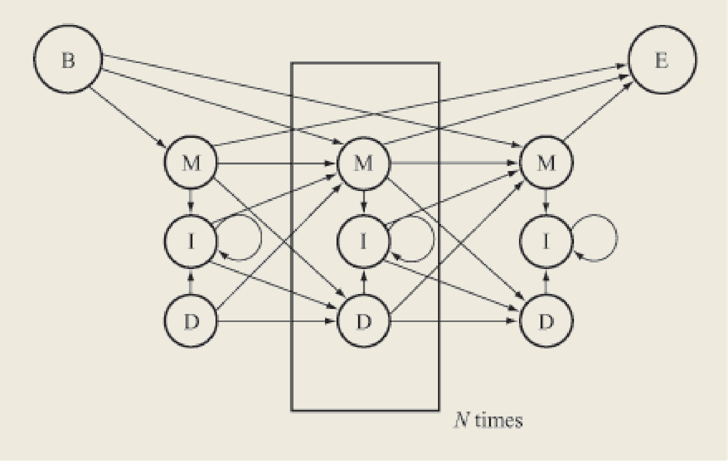
$$
\begin{aligned}
\Phi_{a,s}(\mathbf{x}) &= \sum_{\mathbf{y}\in\mathcal{S}*} P(\mathbf{y}|\mathbf{x})\, n_{a,s}(\mathbf{x},\mathbf{y}) \\
&= \sum_{\mathbf{y}\in\mathcal{S}*} P(\mathbf{y}|\mathbf{x}) \left\{ \sum_{i=1}^{n} \delta(x_i,a)\,\delta(y_i,s) \right\} \\
&= \sum_{i=1}^{n} \delta(x_i,a) \left\{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

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

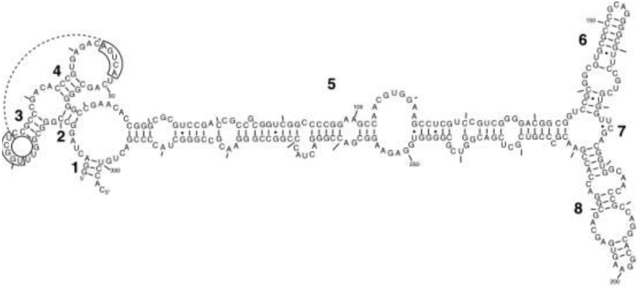
HMM example (DNA)



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)

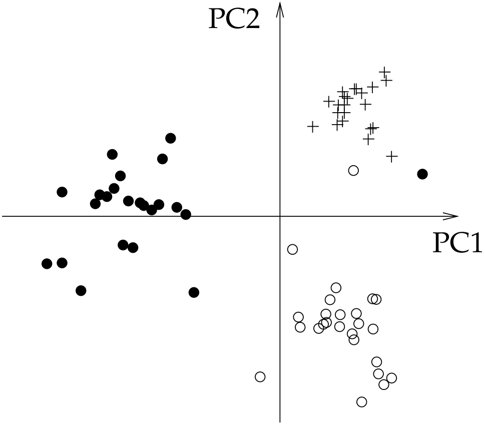
- Feature: number of occurrences of each (base,state) combination
- Marginalization using classical inside/outside algorithm

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

5 Classification of biological sequences

- Motivation
- Feature space approach
- Using generative models
- Derive from a similarity measure
- Application: remote homology detection

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 $g : \mathbb{N} \rightarrow \mathbb{R}$

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...

LA kernel

The local alignment kernel:

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

is symmetric positive definite (Vert et al., 2004).

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 (Vert et al., 2004).

LA kernel is p.d.: proof

- If K_1 and K_2 are p.d. kernels for strings, then their convolution defined by:

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

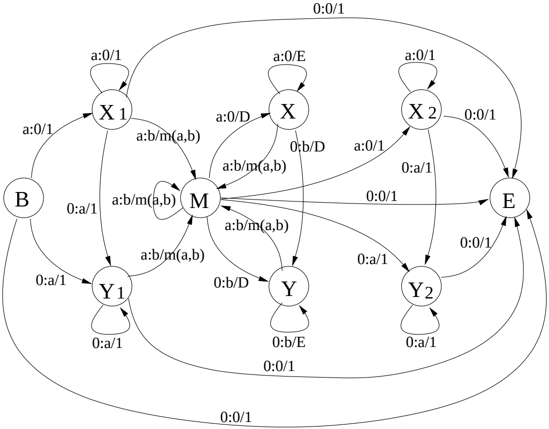
  is also p.d. (Haussler, 1999).
- LA kernel is p.d. because it is a convolution kernel (Haussler, 1999):

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

  where K_0, K_a and K_g are three basic p.d. kernels (Vert et al., 2004).

LA kernel in practice

- Implementation by dynamic programming in $O(|\mathbf{x}| \times |\mathbf{x}'|)$



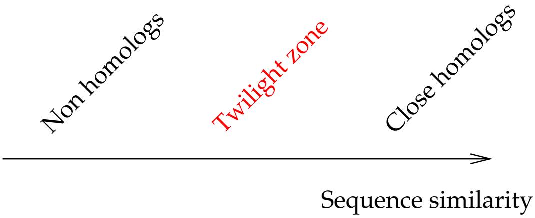
- In practice, values are too large (exponential scale) so taking its logarithm is a safer choice (but not p.d. anymore!)

Outline

5 Classification of biological sequences

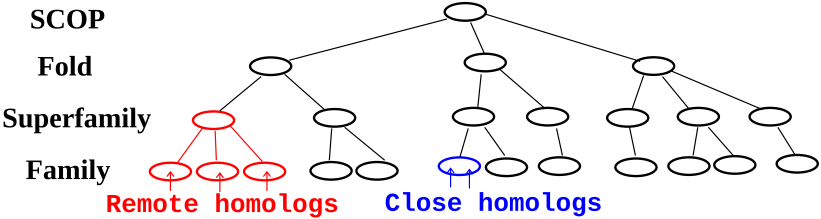
- Motivation
- Feature space approach
- Using generative models
- Derive from a similarity measure
- Application: remote homology detection

Remote homology



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

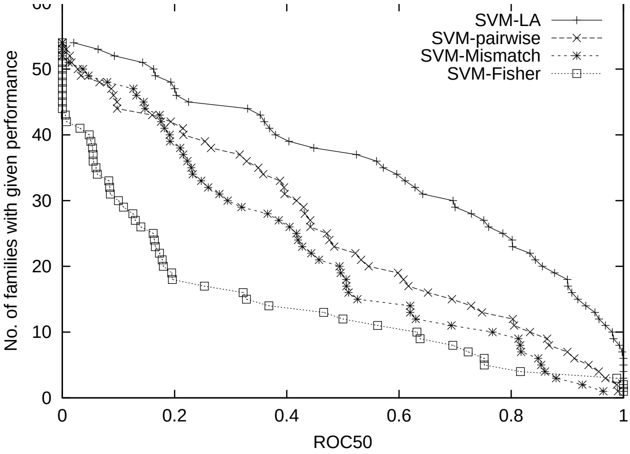
SCOP database



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).

Outline

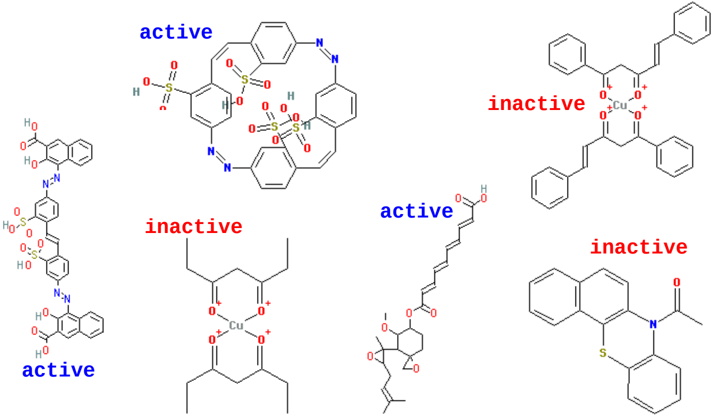
1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Outline

6 Virtual screening and QSAR

- Motivation
- 2D Kernel
- 3D Pharmacophore Kernel

Ligand-Based Virtual Screening and QSAR



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

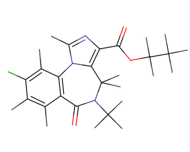
More formally...

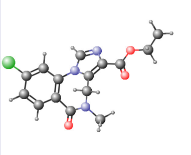
Objective

Build models to predict biochemical properties Y of small molecules from their structures X, using a training set of (X, Y) pairs.

Structures X

$C_{15}H_{14}ClN_3O_3$





Properties Y

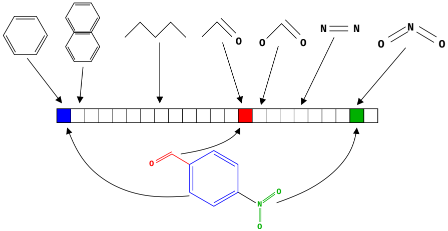
- binding to a therapeutic target,
- pharmacokinetics (ADME),
- toxicity...

Classical approaches

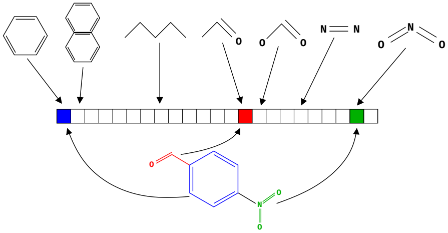
Two steps

1. Map each molecule to a vector of fixed dimension using molecular descriptors
   - Global properties of the molecules (mass, logP...)
   - 2D and 3D descriptors (substructures, fragments, ...)
2. Apply an algorithm for regression or pattern recognition.
   - PLS, ANN, ...

Example: 2D structural keys



Which descriptors?



Difficulties

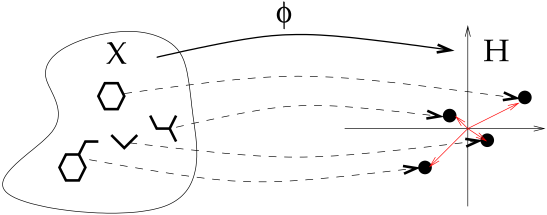
- Many descriptors are needed to characterize various features (in particular for 2D and 3D descriptors)
- But too many descriptors are harmful for memory storage, computation speed, statistical estimation

Kernels

Definition

- Let $\Phi(x) = (\Phi_1(x), \ldots, \Phi_p(x))$ be a vector representation of the molecule x
- The kernel between two molecules is defined by:

$$K(x, x') = \Phi(x)^\top \Phi(x') = \sum_{i=1}^{p} \Phi_i(x)\Phi_i(x') \,.$$

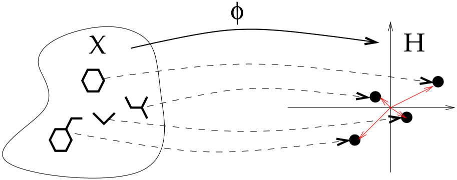


Making kernels for molecules

- Strategy 1: use well-known molecular descriptors to represent molecules m as vectors $\Phi(m)$, and then use kernels for vectors, e.g.:

$$K(m_1, m_2) = \Phi(m_1)^\top \Phi(m_2).$$

- Strategy 2: invent new kernels to do things you can not do with strategy 1, such as using an infinite number of descriptors. We will now see two examples of this strategy, extending 2D and 3D molecular descriptors.



Summary

The problem

- Regression and pattern recognition over molecules
- Classical vector representation is both statistically and computationally challenging

The kernel approach

By defining a kernel for molecules we can work implicitly in large (potentially infinite!) dimensions:

- Allows to consider a large number of potentially important features.
- No need to store explicitly the vectors (no problem of memory storage or hash clashes) thanks to the kernel trick
- Use of regularized statistical algorithm (SVM, kernel PLS, kernel perceptron...)to handle the statistical problem of large dimension

Summary

The problem

- Regression and pattern recognition over molecules
- Classical vector representation is both statistically and computationally challenging

The kernel approach

By defining a kernel for molecules we can work implicitly in large (potentially infinite!) dimensions:

- Allows to consider a large number of potentially important features.
- No need to store explicitly the vectors (no problem of memory storage or hash clashes) thanks to the kernel trick
- Use of regularized statistical algorithm (SVM, kernel PLS, kernel perceptron...)to handle the statistical problem of large dimension

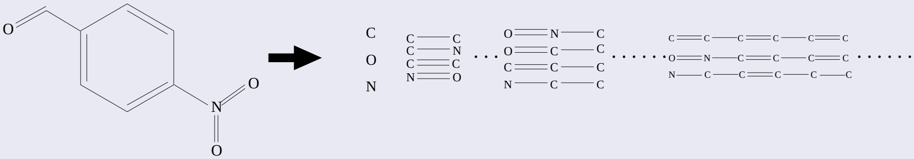
Outline

6 Virtual screening and QSAR
- Motivation
- 2D Kernel
- 3D Pharmacophore Kernel

Motivation: 2D Fingerprints

Features

A vector indexed by a large set of molecular fragments



Pros

- Many features
- Easy to detect

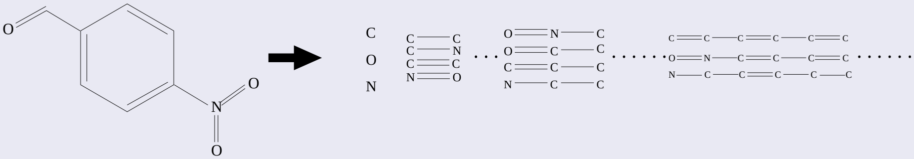
Cons

- Too many features?
- Hashing $\Longrightarrow$ clashes

Motivation: 2D Fingerprints

Features

A vector indexed by a large set of molecular fragments



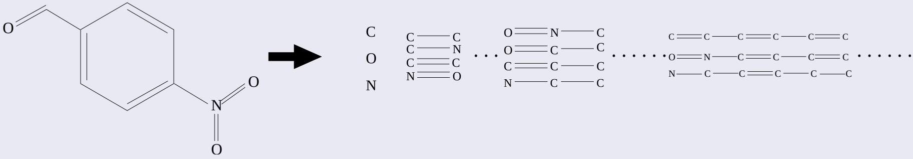
Pros

- Many features
- Easy to detect

Cons

- Too many features?
- Hashing $\Longrightarrow$ clashes

SVM approach



Let $\Phi(x)$ the vector of fragment counts:

- Long fragments lead to large dimensions :
  SVM can learn in high dimension
- $\Phi(x)$ is too long to be stored, and hashes induce clashes:
  SVM do not need $\Phi(x)$, they just need the kernel

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

2D fingerprint kernel

Definition

- For any $d > 0$ let $\phi_d(x)$ be the vector of counts of all fragments of length d:

$$\phi_1(x) = (\quad \texttt{\#(C),\#(O),\#(N),} \quad \ldots)^\top$$
$$\phi_2(x) = (\quad \texttt{\#(C-C),\#(C=O),\#(C-N),} \quad \ldots)^\top \quad \text{etc...}$$

- The 2D fingerprint kernel is defined, for $\lambda < 1$, by

$$K_{2D}(x, x') = \sum_{d=1}^{\infty} \lambda^d \phi_d(x)^\top \phi_d(x') \, .$$

- This is an inner product in the space of 2D fingerprints of infinite length.

2D kernel computation

Theorem

The 2D fingerprint kernel between two molecules x and x' can be computed with a worst-case complexity $O\left((\mid x \mid \times \mid x' \mid)^3\right)$ (much faster in practice).

Remarks

- The complexity is not related to the length of the fragments considered (although faster computations are possible if the length is limited).
- Solves the problem of clashes and memory storage.
- Allows to work with infinite-length fingerprints without computing them!

2D kernel computation

Theorem

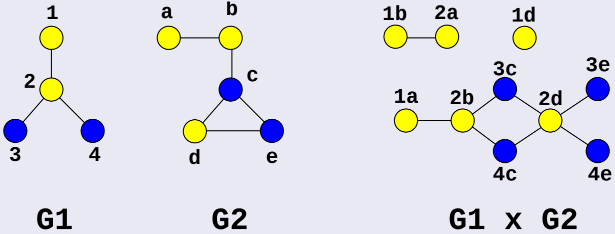
The 2D fingerprint kernel between two molecules x and x' can be computed with a worst-case complexity $O\left((\mid x \mid \times \mid x' \mid)^3\right)$ (much faster in practice).

Remarks

- The complexity is not related to the length of the fragments considered (although faster computations are possible if the length is limited).
- Solves the problem of clashes and memory storage.
- Allows to work with infinite-length fingerprints without computing them!

2D kernel computation trick

- Rephrase the kernel computation as that as counting the number of walks on a graph (the product graph)

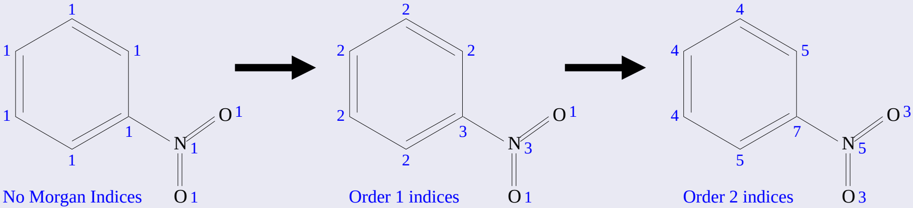


- The infinite counting can be factorized

$$\lambda A + \lambda^2 A^2 + \lambda^3 A^3 + \ldots = (I - \lambda A)^{-1} - I\,.$$

Extensions 1: label enrichment

Atom relabebling with the Morgan index

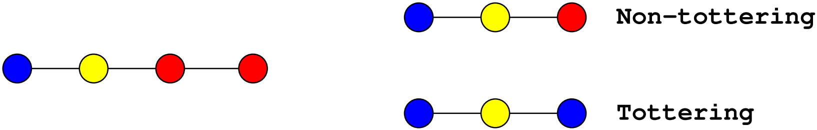


- Compromise between fingerprints and structural keys features.
- Other relabeling schemes are possible.
- Faster computation with more labels (less matches implies a smaller product graph).

Extension 2: Non-tottering walk kernel

Tottering walks

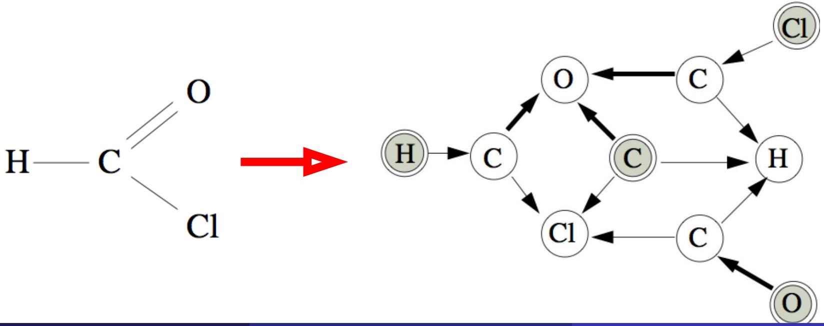
A tottering walk is a walk $w = v_1 \ldots v_n$ with $v_i = v_{i+2}$ for some i.



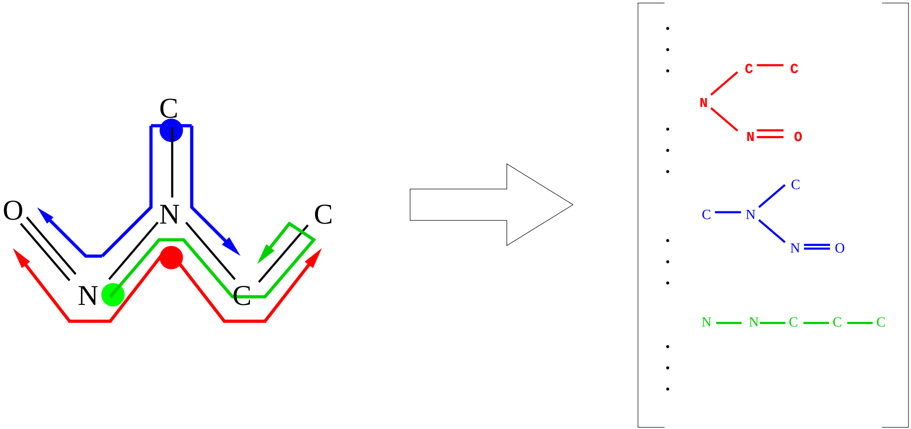
- Tottering walks seem irrelevant for many applications
- Focusing on non-tottering walks is a way to get closer to the path kernel (e.g., equivalent on trees).

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).



Extensions 3: tree-like fragments



Experiments

MUTAG dataset

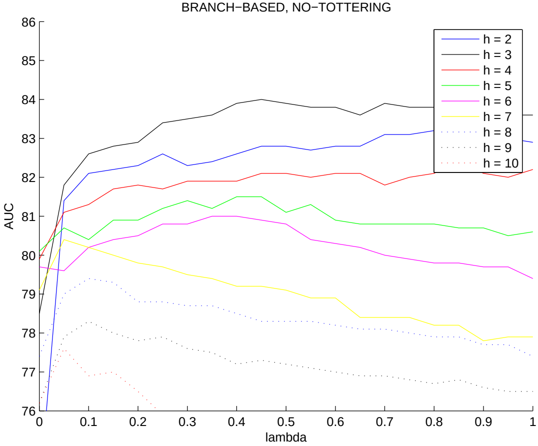
- aromatic/hetero-aromatic compounds
- high mutagenic activity /no mutagenic activity, assayed in *Salmonella typhimurium*.
- 188 compouunds: 125 + / 63 -

Results

10-fold cross-validation accuracy

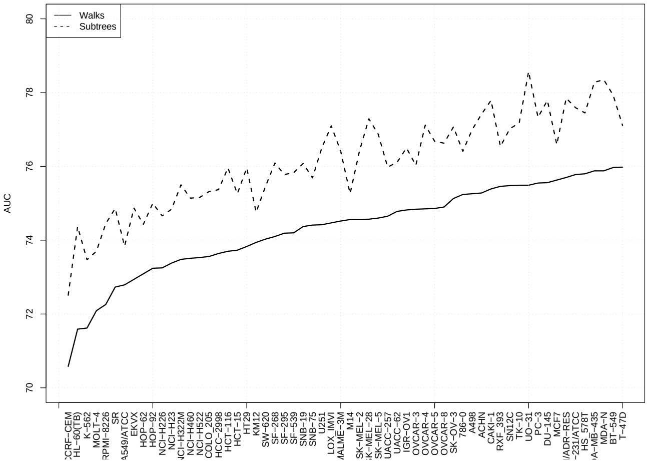
Method	Accuracy
Progol1	81.4%
2D kernel	91.2%

Subtree kernels



AUC as a function of the branching factors for different tree depths (from Mahé et al., 2007).

2D Subtree vs walk kernels



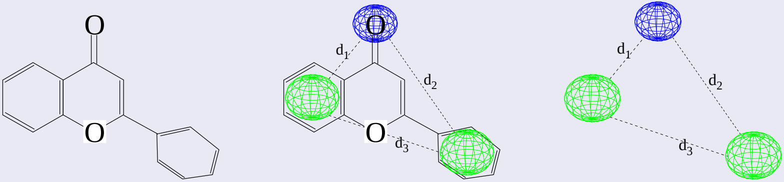
Screening of inhibitors for 60 cancer cell lines.

Outline

6 Virtual screening and QSAR
- Motivation
- 2D Kernel
- 3D Pharmacophore Kernel

Space of pharmacophore

3-points pharmacophores



A set of 3 atoms, and 3 inter-atom distances:

$$\mathcal{T} = \{((x_1, x_2, x_3), (d_1, d_2, d_3)), x_i \in \{\text{atom types}\}; d_i \in \mathbb{R}\}$$

3D fingerprint kernel

Pharmacophore fingerprint

1. Discretize the space of pharmacophores $\mathcal{T}$ (e.g., 6 atoms or groups of atoms, 6-7 distance bins) into a finite set $\mathcal{T}_d$
2. Count the number of occurrences $\phi_t(x)$ of each pharmacophore bin t in a given molecule x, to form a pharmacophore fingerprint.

3D kernel

A simple 3D kernel is the inner product of pharmacophore fingerprints:

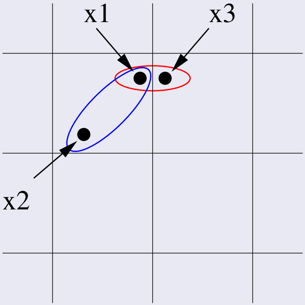
$$K(x, x') = \sum_{t \in \mathcal{T}_d} \phi_t(x)\phi_t(x') \, .$$

Discretization of the pharmacophore space

Common issues

1. If the bins are too large, then they are not specific enough
2. If the bins are too large, then they are too specific

In all cases, the arbitrary position of boundaries between bins affects the comparison:



$\rightarrow d(x_1, x_3) < d(x_1, x_2)$

BUT $\text{bin}(x_1) = \text{bin}(x_2) \neq \text{bin}(x_3)$

Kernels between pharmacophores

A small trick

$$
\begin{aligned}
K(x,y) &= \sum_{t \in \mathcal{T}_d} \phi_t(x)\phi_t(y) \\
&= \sum_{t \in \mathcal{T}_d} \Big(\sum_{p_x \in \mathcal{P}(x)} \mathbf{1}(\mathrm{bin}(\mathbf{p_x}) = \mathbf{t}) \Big) \Big(\sum_{p_y \in \mathcal{P}(y)} \mathbf{1}(\mathrm{bin}(\mathbf{p_y}) = \mathbf{t}) \Big) \\
&= \sum_{p_x \in \mathcal{P}(x)} \sum_{p_y \in \mathcal{P}(y)} \mathbf{1}(\mathrm{bin}(\mathbf{p_x}) = \mathrm{bin}(\mathbf{p_y}))
\end{aligned}
$$

General pharmacophore kernel

$$
K(x,y) = \sum_{p_x \in \mathcal{P}(x)} \sum_{p_y \in \mathcal{P}(y)} K_P(p_x, p_y)
$$

New pharmacophore kernels

- Discretizing the pharmacophore space is equivalent to taking the following kernel between individual pharmacophores:

$$K_P(p_1, p_2) = \mathbf{1}\left(\mathrm{bin}(\mathbf{p_x}) = \mathrm{bin}(\mathbf{p_y})\right)$$

- For general kernels, there is no need for discretization!
- For example, is $d(p_1, p_2)$ is a Euclidean distance between pharmacophores, take:

$$K_P\left(p_1, p_2\right) = \exp\left(-\gamma d\left(p_1, p_2\right)\right) \ .$$

Experiments

4 public datasets

- BZR: ligands for the benzodiazepine receptor
- COX: cyclooxygenase-2 inhibitors
- DHFR: dihydrofolate reductase inhibitors
- ER: estrogen receptor ligands

	TRAIN		TEST	
	Pos	Neg	Pos	Neg
BZR	94	87	63	62
COX	87	91	61	64
DHFR	84	149	42	118
ER	110	156	70	110

Experiments

Results (accuracy)

Kernel	BZR	COX	DHFR	ER
2D (Tanimoto)	71.2	63.0	76.9	77.1
3D fingerprint	75.4	67.0	76.9	78.6
3D not discretized	**76.4**	**69.8**	**81.9**	**79.8**

Outline

1. Pattern recognition and regression
2. Support vector machines
3. Classification of CGH data
4. Classification of expression data
5. Classification of biological sequences
6. Virtual screening and QSAR
7. Conclusion

Summary

- Genomic data are often high-dimensional and structured
- Inference is possible with the constrained ERM principle
- Prior knowledge can be included in the contraint, e.g.:
  - Sparsity-inducing priors
  - Euclidean balls with kernels
- The final performance depends a lot on the prior constraint when few examples are availabe ⟹ it is a good place to put some effort