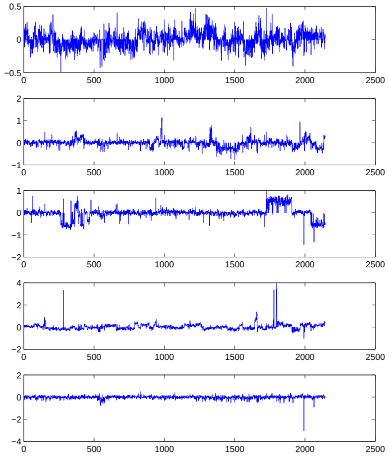
Shrinkage classifiers for genomic and chemical data

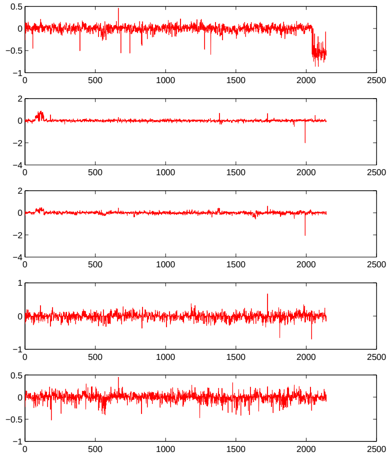
Jean-Philippe Vert
Jean-Philippe.Vert@mines-paristech.fr

Mines ParisTech / Curie Institute / Inserm

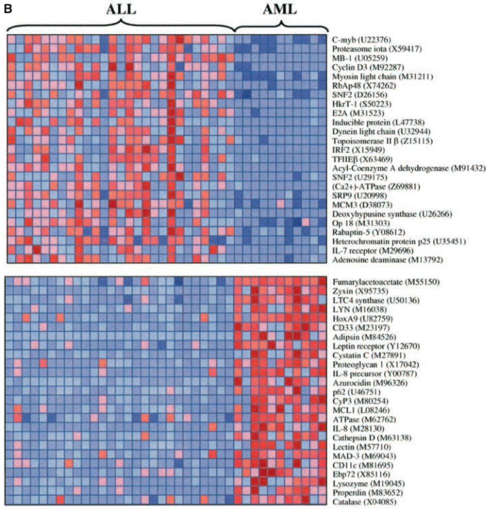
1ère Ecole de Printemps en Apprentissage auTomatique (EPAT 2010), Cap Hornu, France, May 6, 2010

Cancer prognosis

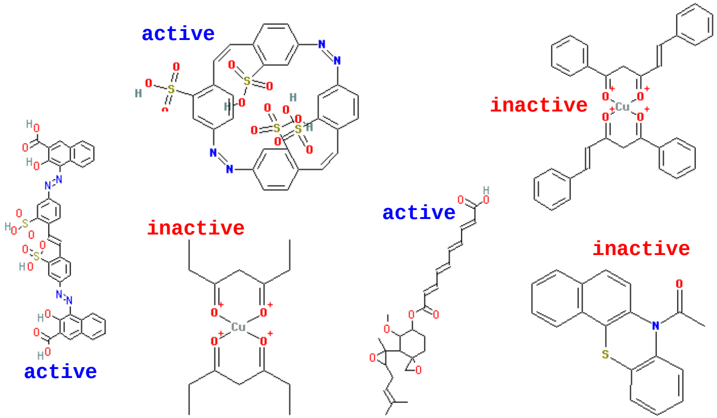




Cancer diagnosis

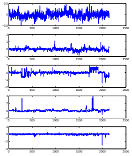


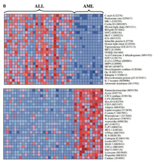
Virtual screening for drug discovery

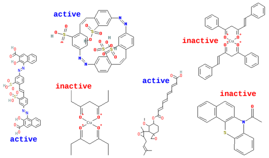


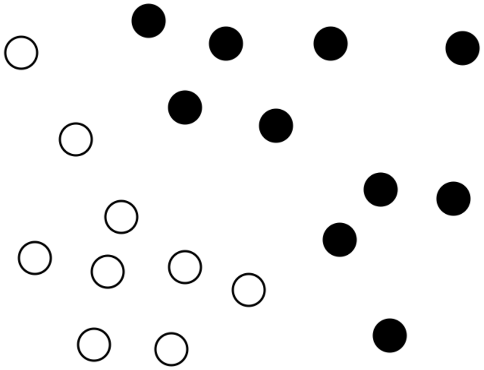
NCI AIDS screen results (from http://cactus.nci.nih.gov).

Pattern recognition, *aka* supervised classification

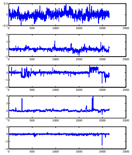


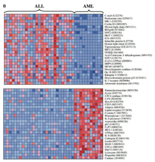


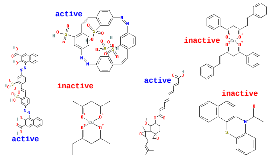


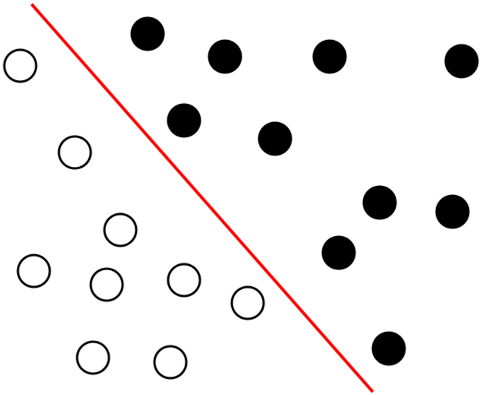


Pattern recognition, *aka* supervised classification

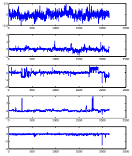


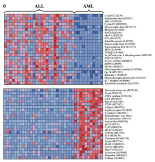


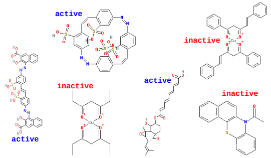


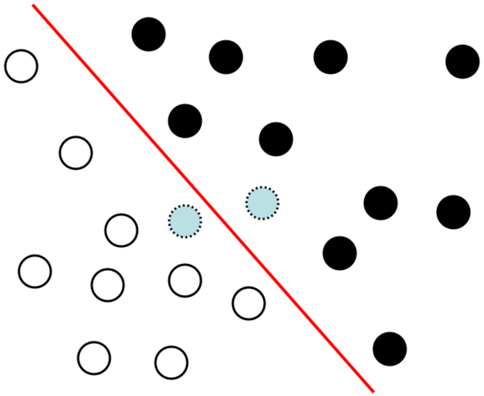


Pattern recognition, *aka* supervised classification

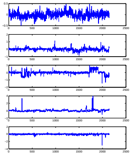


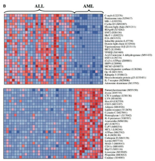


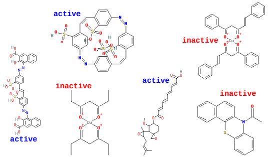


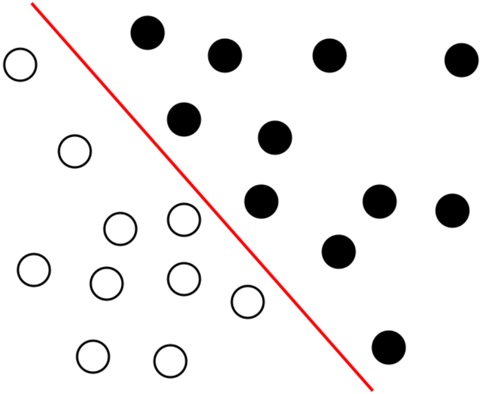


Pattern recognition, *aka* supervised classification

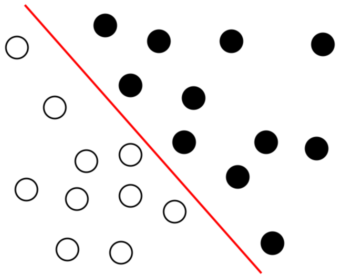








Pattern recognition, *aka* supervised classification



Challenges

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

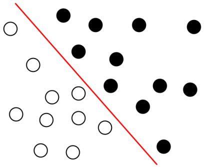
Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

Formalization



The problem

- Given a set of training instances $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in \mathcal{X}$ are data and $y_i \in \mathcal{Y}$ are continuous or discrete variables of interest,
- Estimate a function
$$y = f(x)$$
where x is any new data to be labeled.
- f should be accurate and intepretable.

Linear classifiers

The model

- Each sample $x \in \mathcal{X}$ is represented by a vector of features (or descriptors, or patterns):

$$\Phi(x) = (\Phi_1(x), \ldots, \Phi_p(x)) \in \mathbb{R}^p .$$

- Based on the training set we estimate a linear function:

$$f_\beta(x) = \sum_{i=1}^{p} \beta_i \Phi_i(x) = \beta^\top \Phi(x) .$$

Shrinkage classifiers

- For any candidate set of weights $\beta = (\beta_1, \ldots, \beta_p)$ we quantify how "good" the linear function f_β is on the training set with some empirical risk, typicalle:

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

- We choose the β that achieves the minimium empirical risk, subject to some constraint:

$$\Omega(\beta) \leq C .$$

- Equivalently we solve

$$\min_{\beta \in \mathbb{R}^p} R(\beta) + \lambda \Omega(\beta) .$$

Example 1: kernel methods, SVM

- Penalty:

$$\Omega_{\mathrm{SVM}}(\beta) = \| \beta \|_2^2 = \sum_{i=1}^{p} \beta_i^2 \,.$$

- Kernel trick: we can efficiently solve

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} l(\beta^\top \Phi(x_i), y_i) + \lambda \| \beta \|^2 \,,$$

even for large of infinite p, if we can compute efficiently the kernel:

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

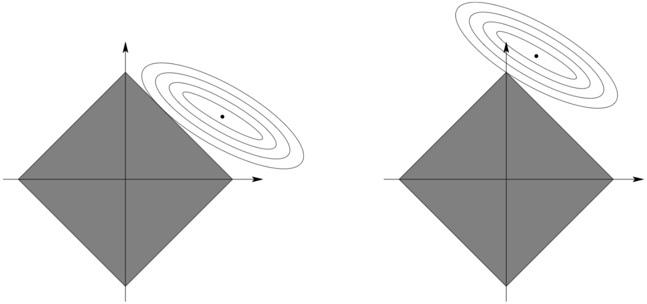
Example 2: feature selection with LASSO

- Penalty:

$$\Omega_{\text{LASSO}}(\beta) = \parallel \beta \parallel_1 = \sum_{i=1}^{p} | \beta_i | \; .$$

- The solution is usually sparse.

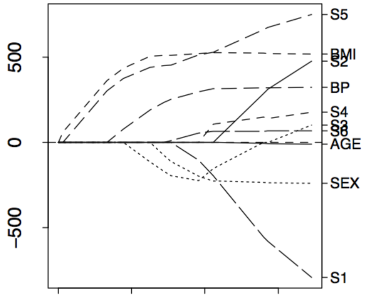
Geometric interpretation with $p = 2$



Efficienty computation of the regularization path

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

- 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)



Shrinkage classifiers - Summary

- We focus on linear classifiers

$$f_\beta(x) = \beta^\top \Phi(x)$$

- We estimate β by solving an optimization problem:

$$\min_{\beta \in \mathbb{R}^p} R(\beta) + \lambda \Omega(\beta_i)$$

Two (related) questions

- How to design the features $\Phi(x)$?
- How to design the penalty $\Omega(\beta)$?
- We will now see some specific answers to these questions for specific problems.

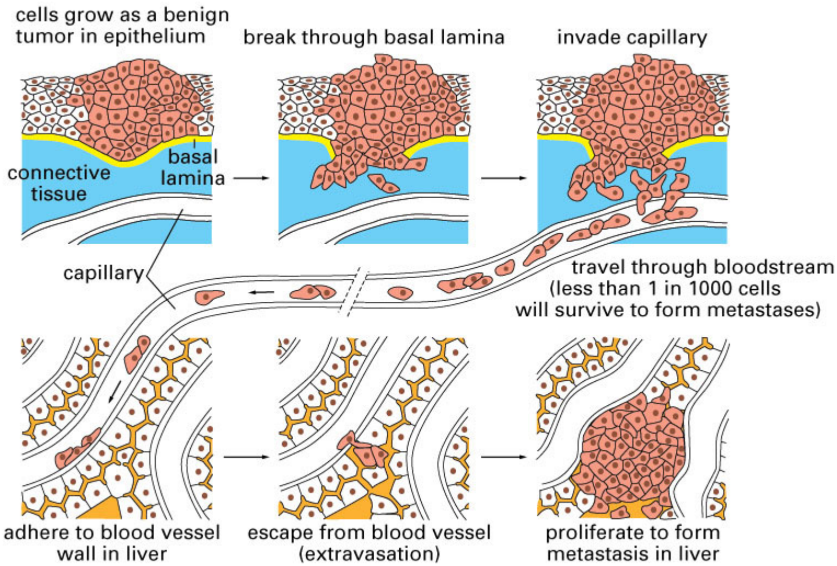
Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

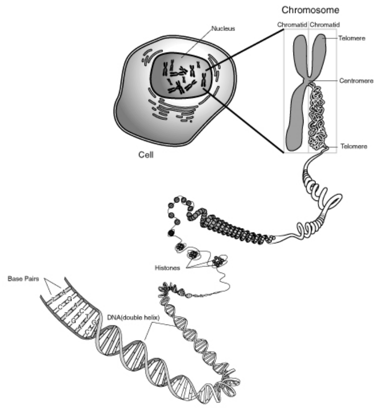
Outline

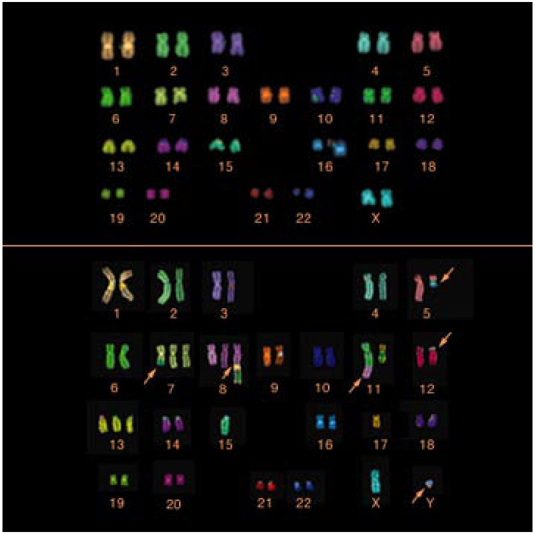
1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

A simple view of cancer progression


cells grow as a benign tumor in epithelium
break through basal lamina
invade capillary
connective tissue
basal lamina
capillary
travel through bloodstream (less than 1 in 1000 cells will survive to form metastases)
adhere to blood vessel wall in liver
escape from blood vessel (extravasation)
proliferate to form metastasis in liver

Chromosomic aberrations in cancer

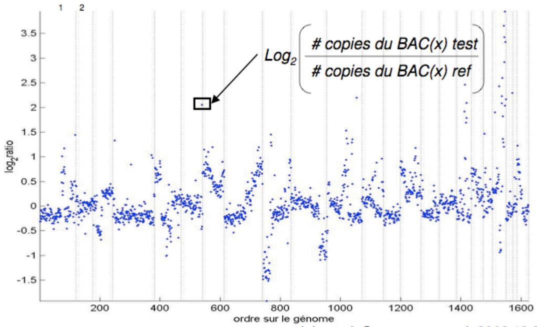




Comparative Genomic Hybridization (CGH)

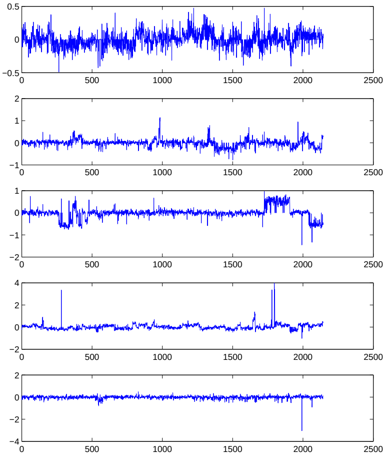
Motivation

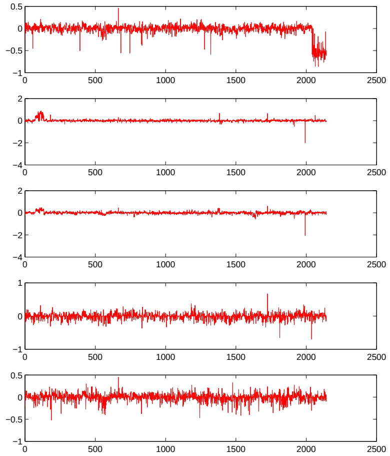
- Comparative genomic hybridization (CGH) data measure the DNA copy number along the genome
- Very useful, in particular in cancer research
- Can we classify CGH arrays for diagnosis or prognosis purpose?



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

Aggressive vs non-aggressive melanoma

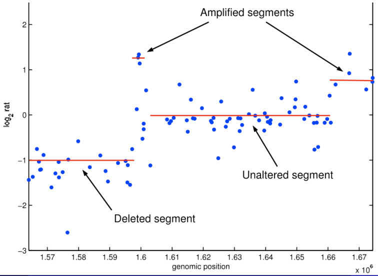




CGH array classification

Prior knowledge

- For a CGH profile $x \in \mathbb{R}^p$, we focus on linear classifiers, i.e., the sign of :
$$f_\beta(x) = \beta^\top x \,.$$
- We expect β to be
  - sparse : not all positions should be discriminative
  - piecewise constant : within a selected region, all probes should contribute equally



Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

Promoting sparsity with the ℓ_1 penalty

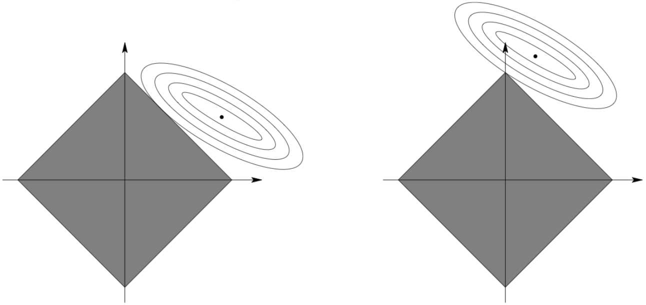
The ℓ_1 penalty (Tibshirani, 1996; Chen et al., 1998)

The solution of

$$\min_{\beta \in \mathbb{R}^p} R(\beta) + \lambda \sum_{i=1}^{p} |\beta_i|$$

is usually sparse.

Geometric interpretation with $p = 2$



Promoting piecewise constant profiles penalty

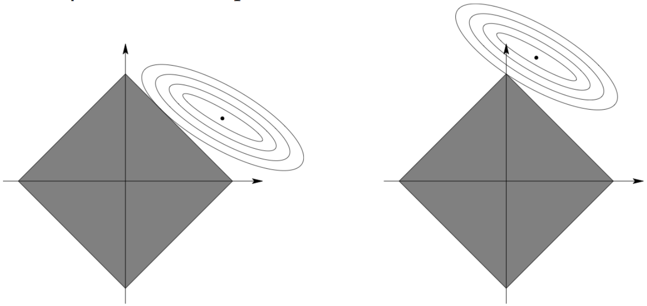
The variable fusion penalty (Land and Friedman, 1996)

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.

Geometric interpretation with $p = 2$



A penalty for CGH array classification

The fused LASSO penalty (Tibshirani et al., 2005)

$$\Omega_{fusedlasso}(\beta) = \sum_{i} |\beta_i| + \sum_{i=1}^{p-1} |\beta_{i+1} - \beta_i| \, .$$

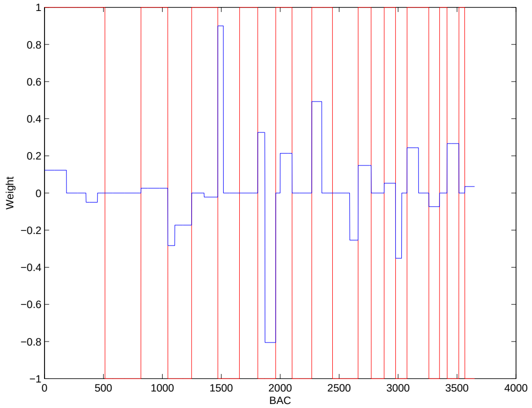
- First term leads to sparse solutions
- Second term leads to piecewise constant solutions

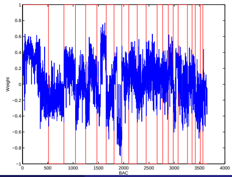
The fused SVM (Rapaport et al., 2008)

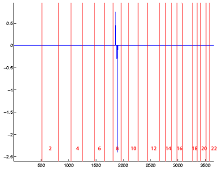
$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \ell\left(y_i, \beta^\top x_i\right) + \lambda \sum_{i=1}^{p} |\beta_i| + \mu \sum_{i=1}^{p-1} |\beta_{i+1} - \beta_i| \, .$$

where ℓ is, e.g., the hinge loss $\ell(y,t) = max(1 - yt, 0)$. It is then a LP.

Application: predicting metastasis in melanoma







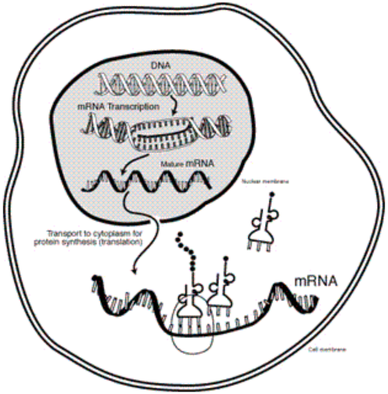
Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

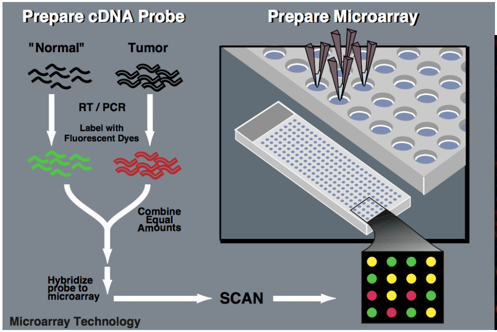
DNA → RNA → protein

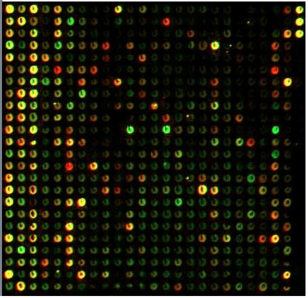


$$\begin{tikzcd}
\boxed{\text{DNA}} \arrow[d, "\text{transcription}"] \\
\boxed{\text{RNA}} \arrow[d, "\text{translation}"] \\
\boxed{\text{Protein}}
\end{tikzcd}$$

- CGH shows the (static) DNA
- Cancer cells have also abnormal (dynamic) gene expression (= transcription)

Tissue profiling with DNA chips

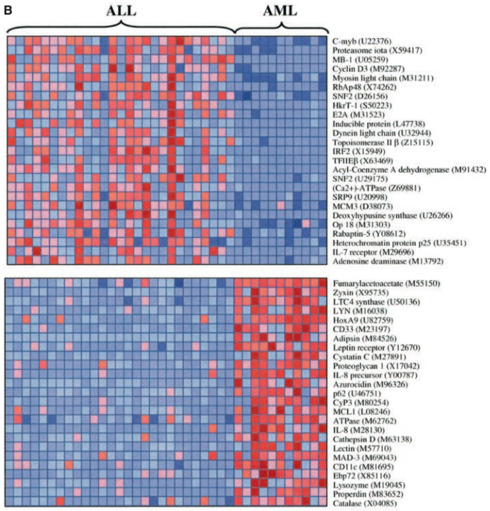




Data

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

Tissue classification from microarray data



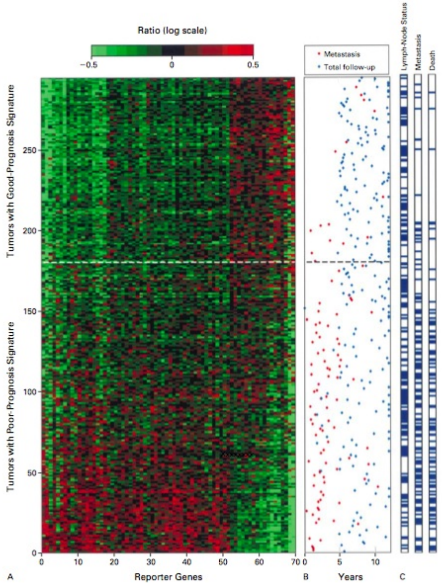
Goal

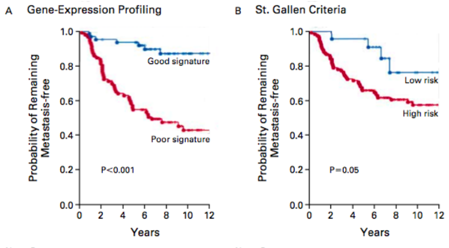
- Design a classifier to automatically assign a class to future samples from their expression profile
- Interpret biologically the differences between the classes

Difficulty

- Large dimension
- Few samples

Prognosis from microarray data (MAMMAPRINT)





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

Gene signature

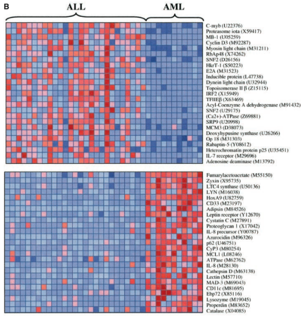
The idea

- We look for a limited set of genes that are sufficient for prediction.
- Equivalently, the linear classifier will be sparse

Motivations

- Bet on sparsity: we believe the "true" model is sparse.
- Interpretation: we will get a biological interpretation more easily by looking at the selected genes.
- Accuracy: by restricting the class of classifiers, we "increase the bias" but "decrease the variance". This should be helpful in large dimensions (it is better to estimate well a wrong model than estimate badly a good model).

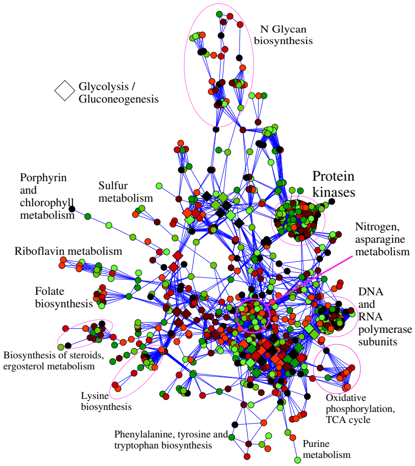
But...

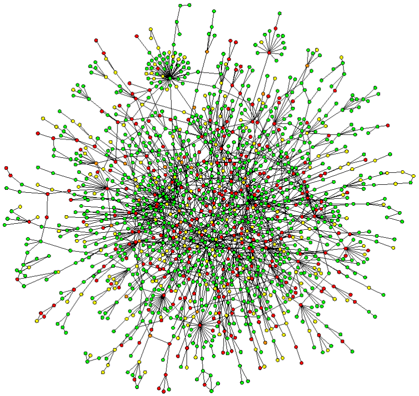


Challenging the idea of gene signature

- We often observe little stability in the genes selected...
- Is gene selection the most biologically relevant hypothesis?
- What about thinking instead of "pathways" or "modules" signatures?

Gene networks

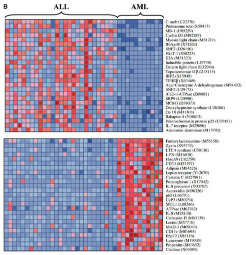

N Glycan biosynthesis
◇ Glycolysis / Gluconeogenesis
Porphyrin and chlorophyll metabolism
Sulfur metabolism
Protein kinases
Nitrogen, asparagine metabolism
Riboflavin metabolism
Folate biosynthesis
DNA and RNA polymerase subunits
Biosynthesis of steroids, ergosterol metabolism
Lysine biosynthesis
Oxidative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

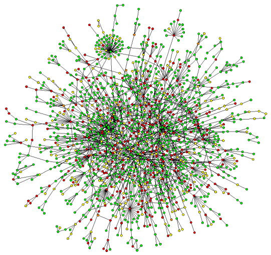


Gene networks and expression data

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

$$\min_{\beta} R(\beta) + \lambda \Omega_G(\beta)$$

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.

Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

Graph based penalty

Prior hypothesis

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

An idea (Rapaport et al., 2007)

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

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

Graph based penalty

Prior hypothesis

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

An idea (Rapaport et al., 2007)

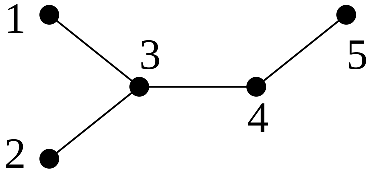
$$\Omega_{spectral}(\beta) = \sum_{i \sim j} (\beta_i - \beta_j)^2 \,,$$

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

Graph Laplacian

Definition

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



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

Spectral penalty as a kernel

Theorem

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

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

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

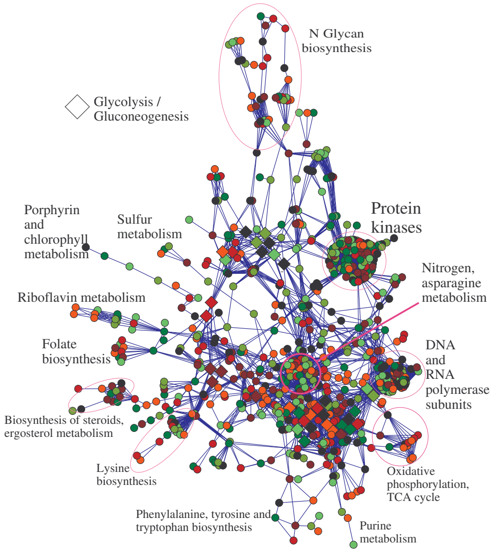
$$\min_{\gamma \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} l\left(\gamma^\top \Phi(x_i), y_i\right) + \lambda \gamma^\top \gamma \,,$$

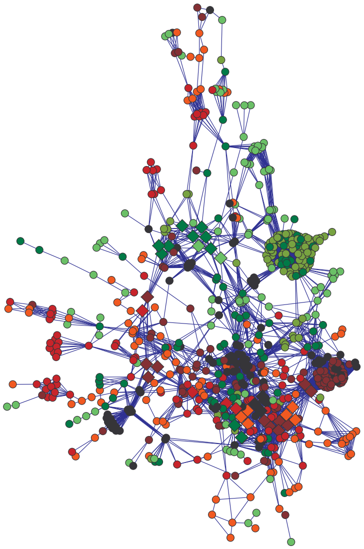
and where

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

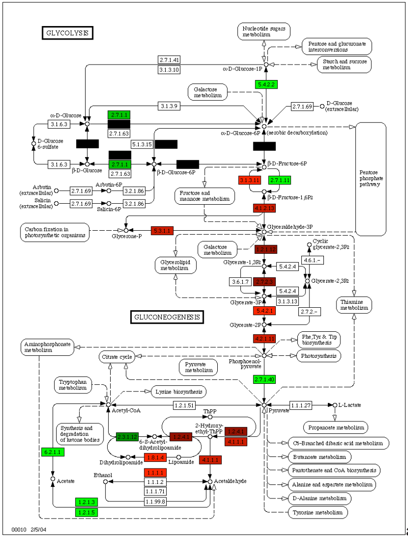
for $K_G = L^*$, the pseudo-inverse of the graph Laplacian.

Classifiers

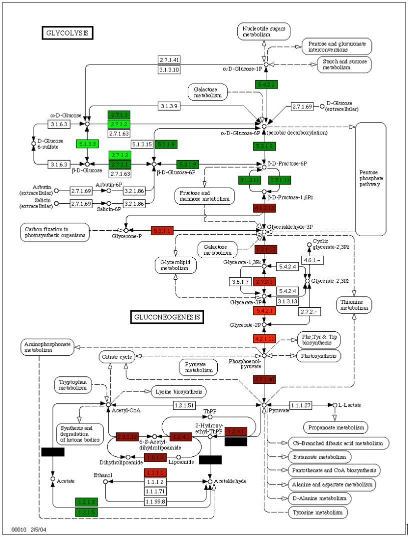

N Glycan biosynthesis
Glycolysis / Gluconeogenesis
Porphyrin and chlorophyll metabolism
Sulfur metabolism
Protein kinases
Nitrogen, asparagine metabolism
Riboflavin metabolism
Folate biosynthesis
DNA and RNA polymerase subunits
Biosynthesis of steroids, ergosterol metabolism
Lysine biosynthesis
Oxidative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

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

- The diffusion kernel:

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

penalizes high frequencies of β in the Fourier domain.

Other penalties without kernels

- Gene selection + Piecewise constant on the graph

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

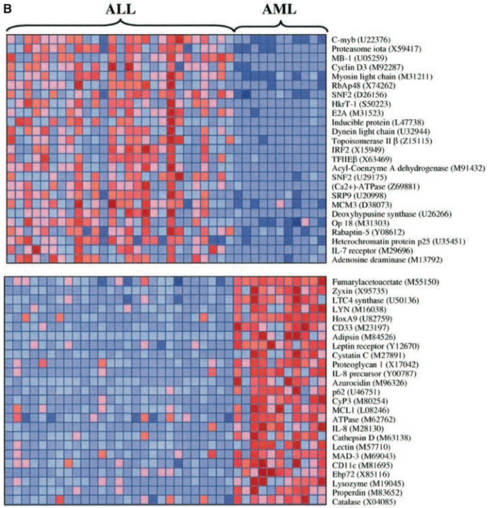
- Gene selection + smooth on the graph

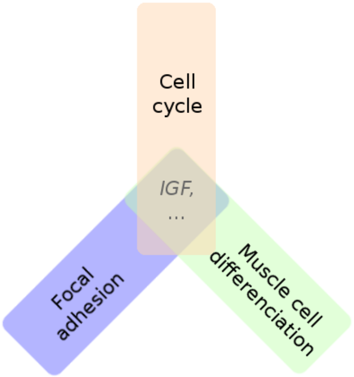
$$\Omega(\beta) = \sum_{i \sim j} (\beta_i - \beta_j)^2 + \sum_{i=1}^{p} | \beta_i |$$

Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

How to select jointly genes belonging to predefined pathways?



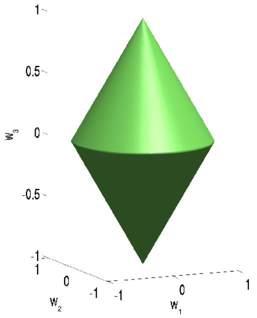


Selecting pre-defined groups of variables

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*:

$$\Omega_{group}(w) = \sum_g \|w_g\|_2$$

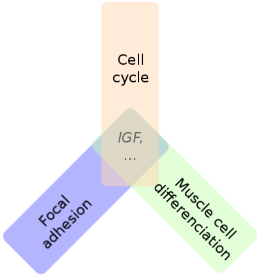


$$\Omega(w_1, w_2, w_3) = \|(w_1, w_2)\|_2 + \|w_3\|_2$$

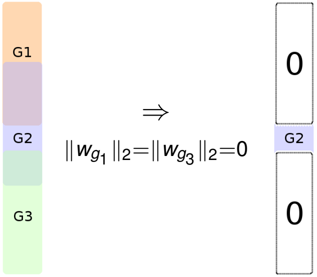
What if a gene belongs to several groups?

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



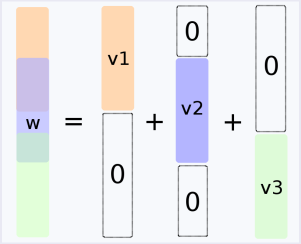
Removal of *any* group containing a gene $\Rightarrow$ the weight of the gene is 0.

Overlap norm (Jacob et al., 2009)

An idea

Introduce latent variables v_g:

$$\begin{cases} \min_{w,v} L(w) + \lambda \sum_{g \in \mathcal{G}} \| v_g \|_2 \\ w = \sum_{g \in \mathcal{G}} v_g \\ \operatorname{supp}\left(v_g\right) \subseteq g. \end{cases}$$



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

$$\begin{cases} \min\limits_{w,v} L(w) + \lambda \sum\limits_{g \in \mathcal{G}} \|v_g\|_2 \\ w = \sum_{g \in \mathcal{G}} v_g \\ \text{supp}\left(v_g\right) \subseteq g. \end{cases} = \min_{w} L(w) + \lambda \Omega_{overlap}(w)$$

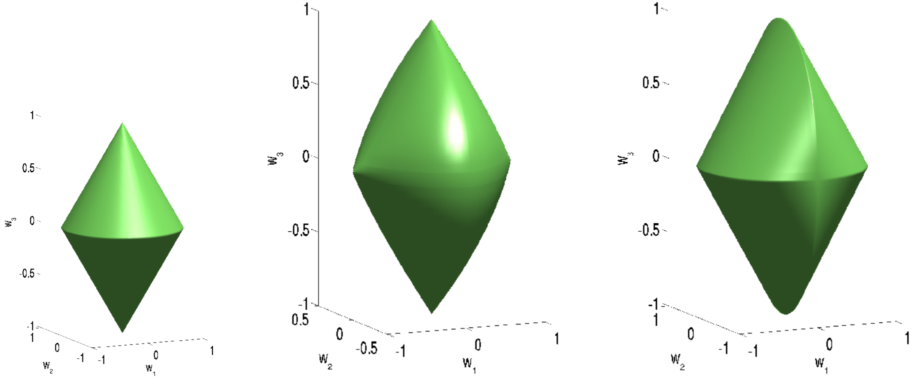
with

$$\Omega_{overlap}(w) \triangleq \begin{cases} \min\limits_{v} \sum\limits_{g \in \mathcal{G}} \|v_g\|_2 \\ w = \sum_{g \in \mathcal{G}} v_g \\ \text{supp}\left(v_g\right) \subseteq g. \end{cases} \qquad (*)$$

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 $L(w) + \lambda \Omega^{\mathcal{G}}_{\text{overlap}}(w)$.

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

$$\left\{ g \in \mathcal{G} | \hat{v}_g \neq 0 \right\} = \left\{ g \in \mathcal{G} | \bar{v}_g \neq 0 \right\}.$$

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 $n \rightarrow \infty$,
- with very high probability,

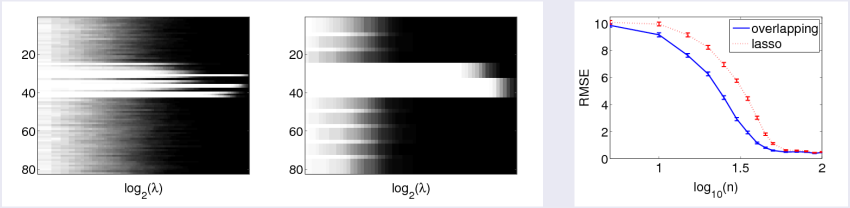
the optimal solution $\hat{w}$ admits a unique decomposition $(\hat{v}_g)_{g \in \mathcal{G}}$ such that

$$\left\{ g \in \mathcal{G} | \hat{v}_g \neq 0 \right\} = \left\{ g \in \mathcal{G} | \bar{v}_g \neq 0 \right\} .$$

Experiments

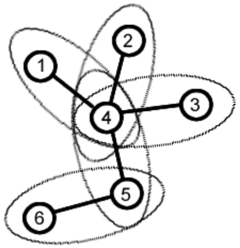
Synthetic data: overlapping groups

- 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

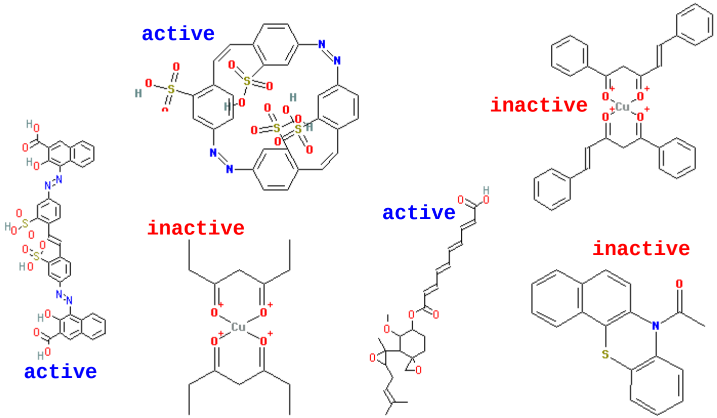
- Graph on the genes.

METHOD	ℓ_1	$\Omega_{graph}(.)$
ERROR	0.39 ± 0.04	0.36 ± 0.01
AV. SIZE C.C.	1.03	1.30

Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

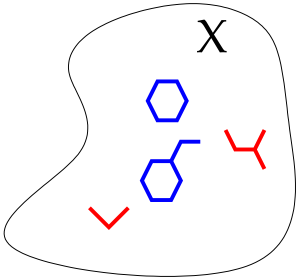
Motivation



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

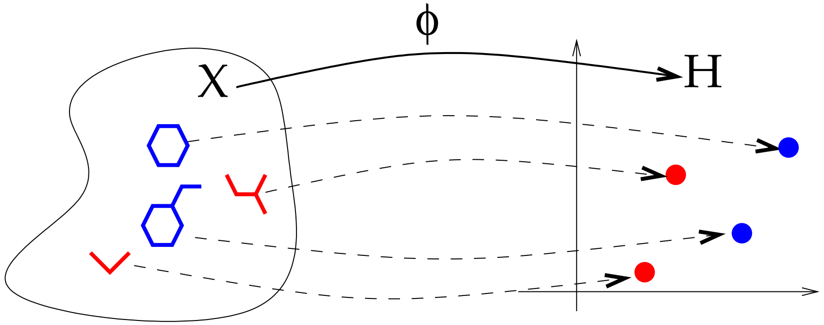
The approach

1. Represent 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 $\mathbb{R}^p$.



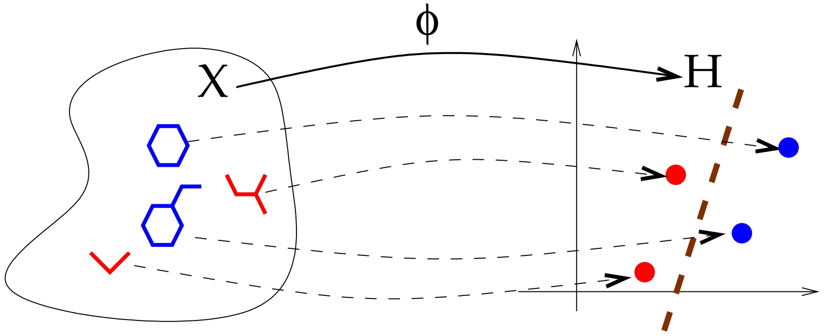
The approach

1. Represent 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 $\mathbb{R}^p$.



The approach

1. Represent 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 $\mathbb{R}^p$.



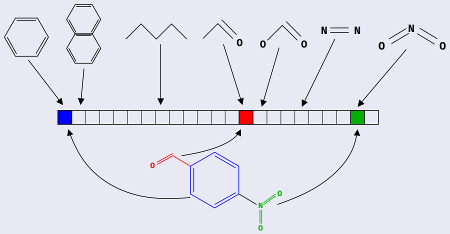
Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

Example

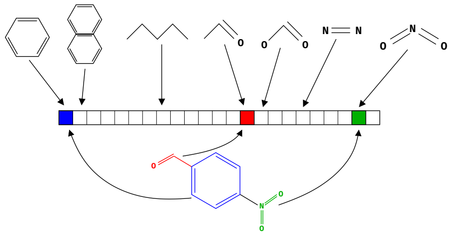
2D structural keys in chemoinformatics

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



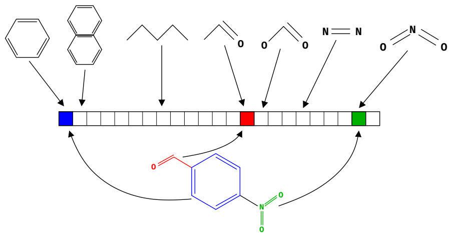
- Use a machine learning algorithms such as SVM, NN, PLS, decision tree, ...

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

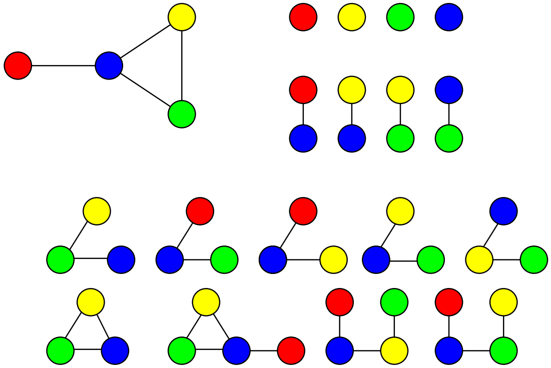


- Often we believe that the presence substructures are important predictive patterns
- Hence it makes sense to represent a graph by features that indicate the presence (or the number of occurrences) of particular substructures
- However, detecting the presence of particular substructures may be computationally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

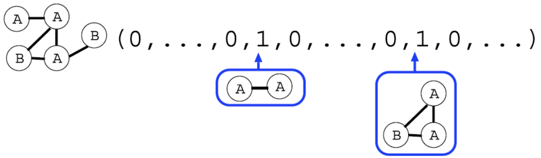
Subgraphs

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?



Theorem

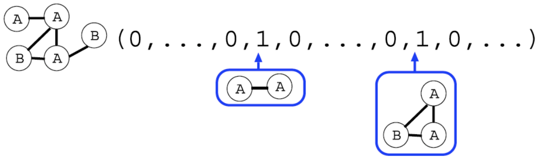
Computing all subgraph occurrences is NP-hard.

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.

□

Indexing by all subgraphs?



Theorem

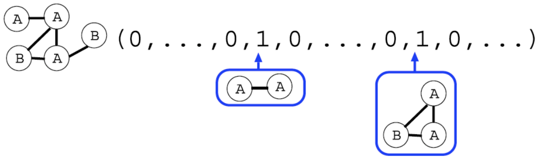
Computing all subgraph occurrences is NP-hard.

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.

$\square$

Indexing by all subgraphs?



Theorem

Computing all subgraph occurrences is NP-hard.

Proof.

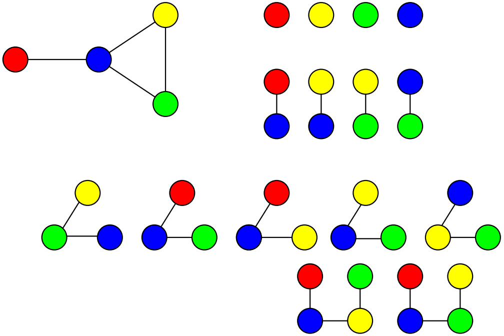
- The linear graph of size n is a subgraph of a graph X with n vertices iff X has an Hamiltonian path
- The decision problem whether a graph has a Hamiltonian path is NP-complete.

$\square$

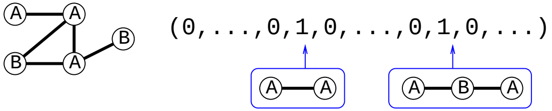
Paths

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?



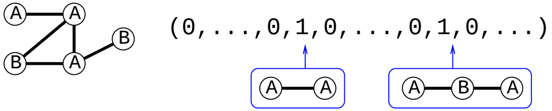
Theorem

Computing all path occurrences is NP-hard.

Proof.

Same as for subgraphs. □

Indexing by all paths?



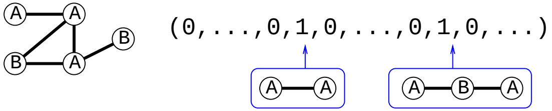
Theorem

Computing all path occurrences is NP-hard.

Proof.

Same as for subgraphs. □

Indexing by all paths?



Theorem

Computing all path occurrences is NP-hard.

Proof.

Same as for subgraphs. □

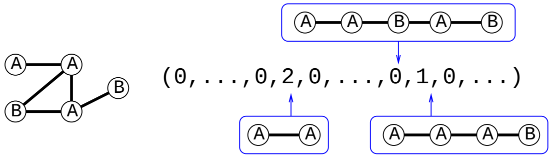
Indexing by what?

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)

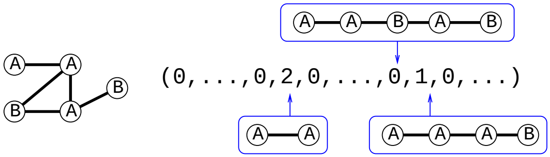
Example : Indexing by all shortest paths



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.

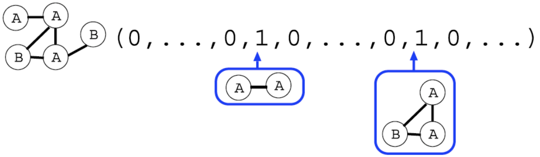
Example : Indexing by all shortest paths



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.

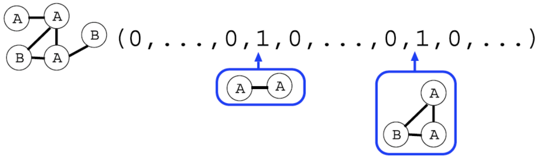
Example : Indexing by all subgraphs up to *k* vertices



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.

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

1 Shrinkage linear classifiers

2 Cancer prognosis from DNA copy number variations
- Motivation
- Penalty inducing piecewise constant classifier

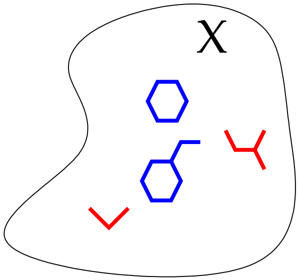
3 Diagnosis and prognosis from gene expression data
- Motivation
- Penalties for smooth classifiers
- Penalties for structured feature selection

4 Graph classification
- Explicit computation of features
- Graph kernels
- Feature selection for all subgraph indexation

5 Conclusion

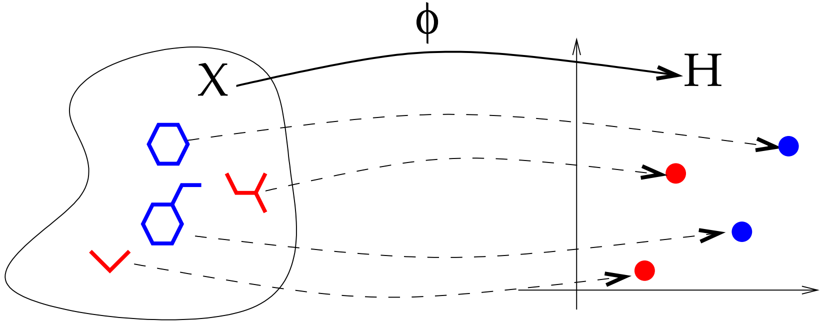
The idea

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 $\mathcal{H}$.



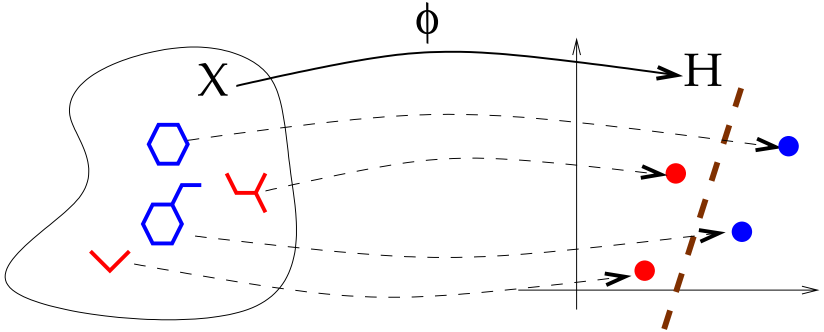
The idea

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 $\mathcal{H}$.



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 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?

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.

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 $G_1 \simeq G_2$). $\square$

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

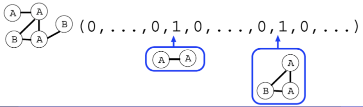
Definition

- 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.

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 (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.

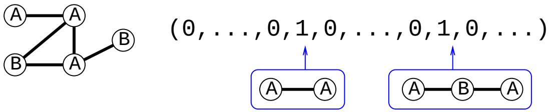
Proof (2/2)

- 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.,

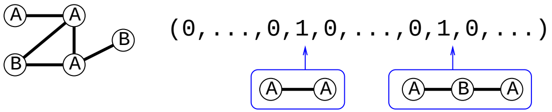
$$K_{path}(G_1, G_2) = \sum_{H \in \mathcal{P}} \lambda_H \Phi_H(G_1) \Phi_H(G_2),$$

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

Proposition (Gärtner et al., 2003)

Computing the path kernel is NP-hard.

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

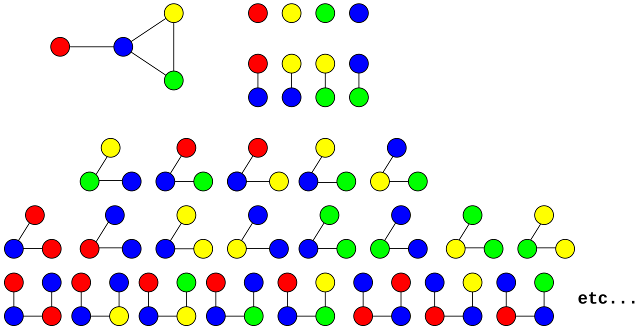
Expressiveness vs Complexity trade-off

- 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.

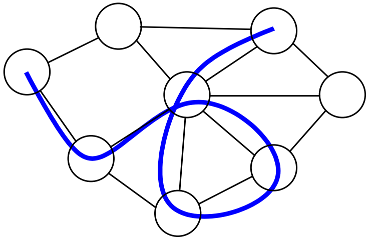
Walks

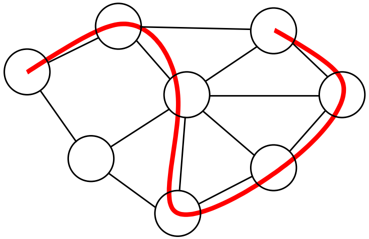
Definition

- 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:

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

- 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

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:

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

- 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 *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).
- 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).

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).

- 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).

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

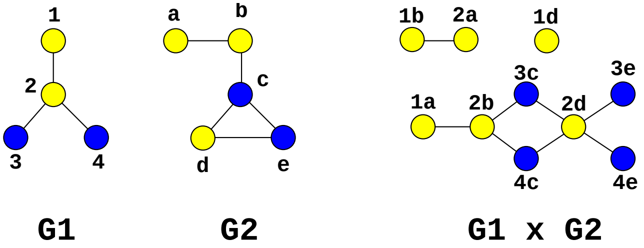
These three kernels (*n*th-order, random and geometric walk kernels) can be computed efficiently in polynomial time.

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. $E = \left\{ ((v_1, v_2), (v_1', v_2')) \in V \times V \, : \, (v_1, v_1') \in E_1 \text{ and } (v_2, v_2') \in E_2 \right\}$.



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

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

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 *n*th-order walk kernel

- For the *n*th-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:

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

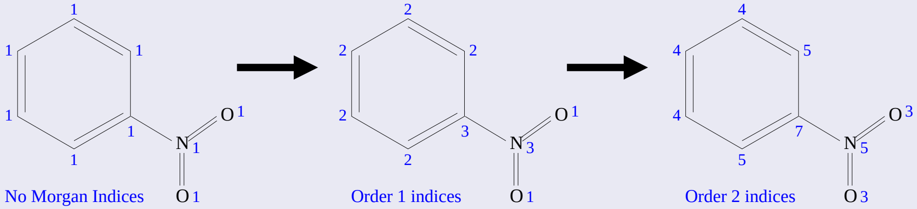
- Let Λ_i be the vector of $\lambda^i(v)$ and Λ_t be the matrix of $\lambda^t(v, v')$:

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

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

Extensions 1: label enrichment

Atom relabebling with the Morgan index

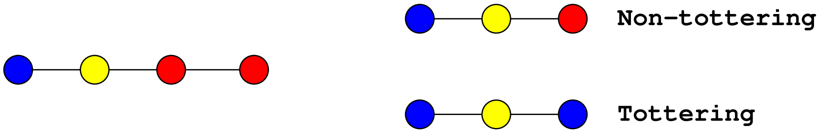


- Compromise between fingerprints and structural keys features.
- Other relabeling schemes are possible (graph coloring).
- Faster computation with more labels (less matches implies a smaller product graph).

Extension 2: Non-tottering walk kernel

Tottering walks

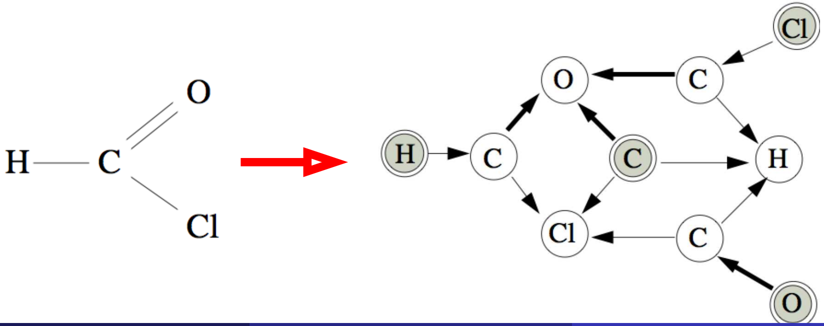
A tottering walk is a walk $w = v_1 \dots v_n$ with $v_i = v_{i+2}$ for some i.



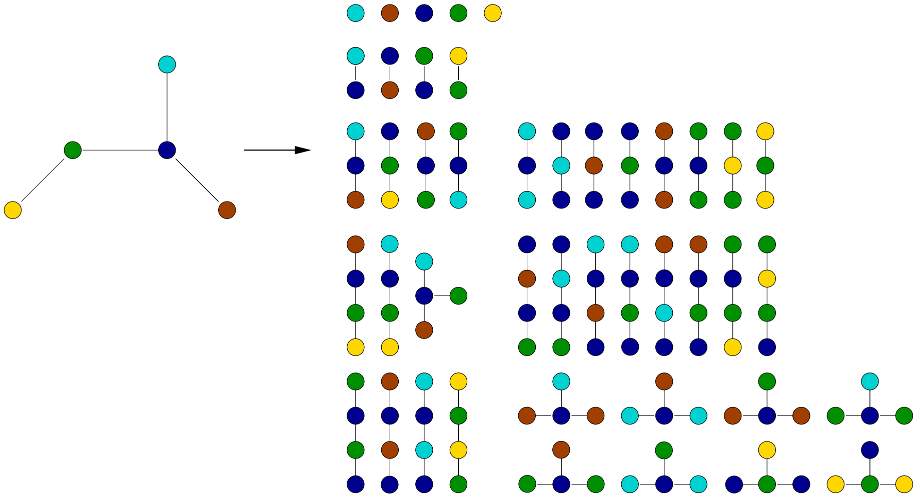
- Tottering walks seem irrelevant for many applications
- Focusing on non-tottering walks is a way to get closer to the path kernel (e.g., equivalent on trees).

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

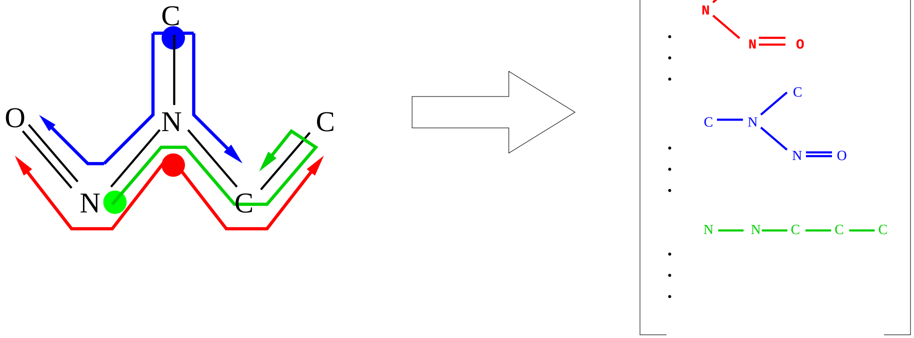
- Second-order Markov random walk to prevent tottering walks
- Written as a first-order Markov random walk on an augmented graph
- Normal walk kernel on the augmented graph (which is always a directed graph).



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.

Application in chemoinformatics (Mahé et al., 2004)

MUTAG dataset

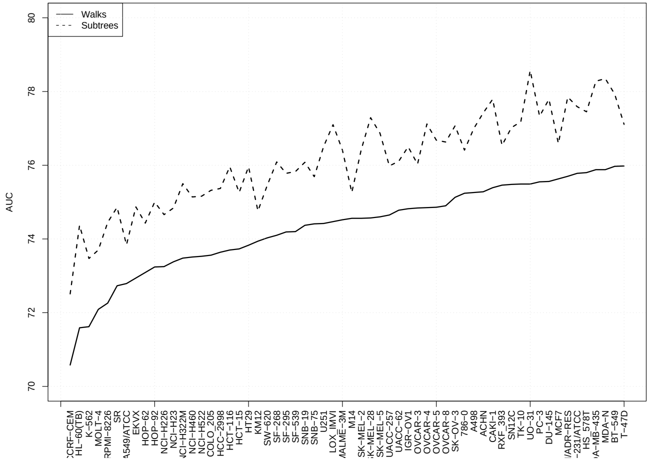
- aromatic/hetero-aromatic compounds
- high mutagenic activity /no mutagenic activity, assayed in *Salmonella typhimurium*.
- 188 compouunds: 125 + / 63 -

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.

Summary: graph kernels

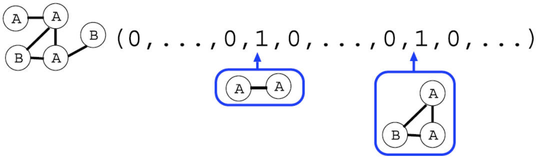
What we saw

- 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

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

Motivation

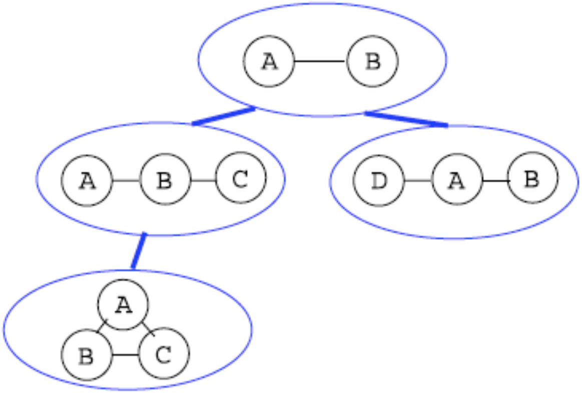


- Indexing by all subgraphs is appealing but intractable in practice (both explicitly and with the kernel trick)
- Can we work implicitly with this representation using sparse learning, e.g., LASSO regression or boosting?
- This may lead to both accurate predictive model and the identification of discriminative patterns.
- The iterations of LARS or boosting amount to an optimization problem over subgraphs, which may be solved efficiently using graph mining technique...

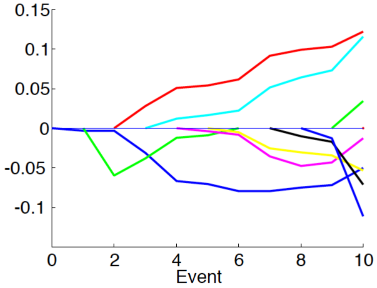
Boosting over subgraph indexation (Kudo et al., 2004)

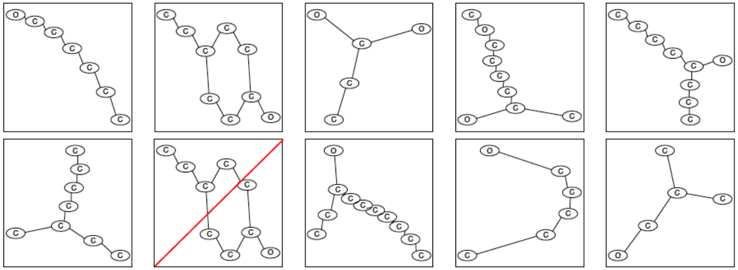
- Weak learner = decision stump indexed by subgraph H and $\alpha = \pm 1$:
$$h_{\alpha,H}(G) = \alpha \Phi_H(G)$$
- Boosting: at each iteration, for a given distribution $d_1 + \ldots + d_n = 1$ over the training points (G_i, y_i), select a weak learner (subgraph $\tilde{H}$) which maximizes the gain
$$gain(H, \alpha) = \sum_{i=1}^{n} y_i h_{\alpha,H}(G_i) \,.$$
- This can be done "efficiently" by branch-and-bound over a DFS code tree (Yan and Han, 2002).

The DFS code tree



Graph LASSO regularization path (Tsuda, 2007)





Summary

- Sparse learning is practically feasible in the space of graphs indexed by all subgraphs
- Leads to subgraph selection
- Several extensions
  - LASSO regularization path (Tsuda, 2007)
  - gboost (Saigo et al., 2009)
- A beautiful and promising marriage between machine learning and data mining

Outline

1. Shrinkage linear classifiers
2. Cancer prognosis from DNA copy number variations
   - Motivation
   - Penalty inducing piecewise constant classifier
3. Diagnosis and prognosis from gene expression data
   - Motivation
   - Penalties for smooth classifiers
   - Penalties for structured feature selection
4. Graph classification
   - Explicit computation of features
   - Graph kernels
   - Feature selection for all subgraph indexation
5. Conclusion

Conclusion

- Machine learning with complex and structured data becomes the rule
- Shrinkage methods (SVM, LASSO, ...) are widely used with default penalty function, and offer nice possibilities to include prior knowledge in the penalty while remaining a convex optimization problem.
- We surveyed several ideas
  - Learning with kernels
  - Learning with sparsity
  - Feature construction
- Performance and interpretability are both important