Machine learning for ligand-based virtual screening and chemogenomics

Jean-Philippe Vert

Institut Curie - INSERM U900 - Mines ParisTech

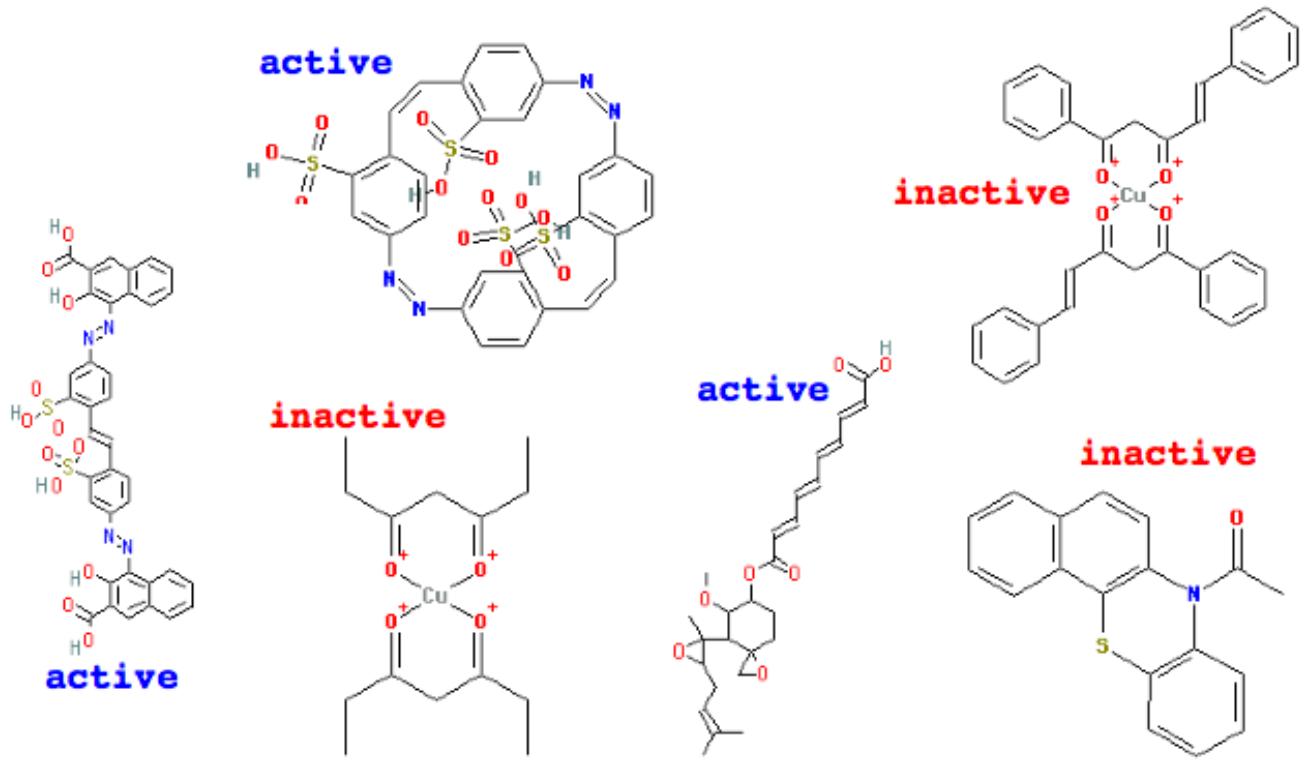
2nd International Summer School on Chemoinformatics,
June 20-24, 2010, Obernai, France

Outline

1. SVM for ligand-based virtual screening
2. Kernels for molecules
3. Towards *in silico* chemogenomics

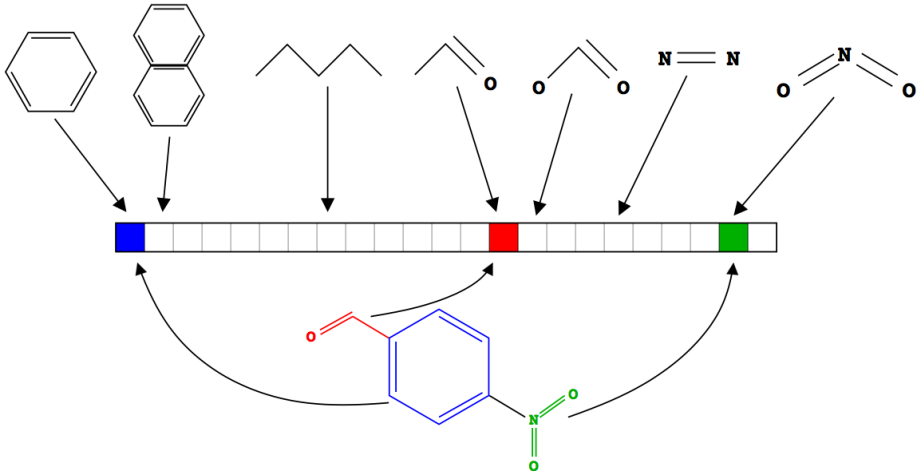
SVM for ligand-based virtual screening

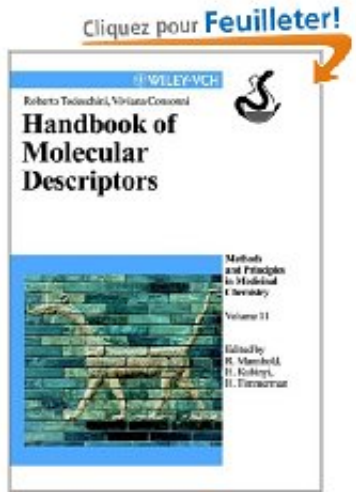
Ligand-based virtual screening / QSAR

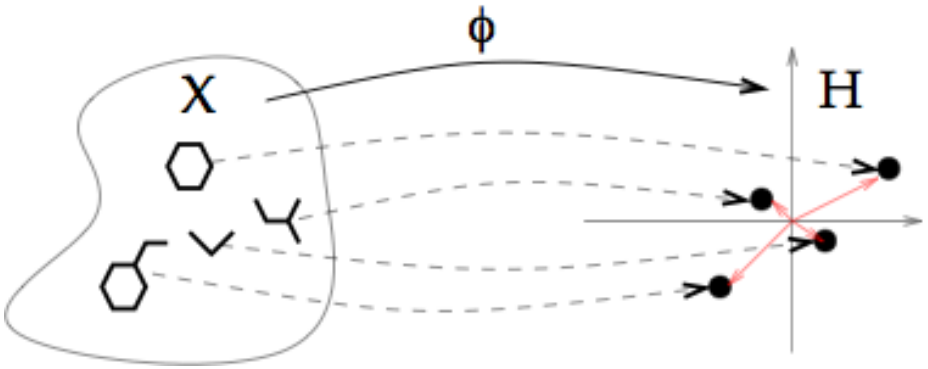


From http://cactus.nci.nih.gov

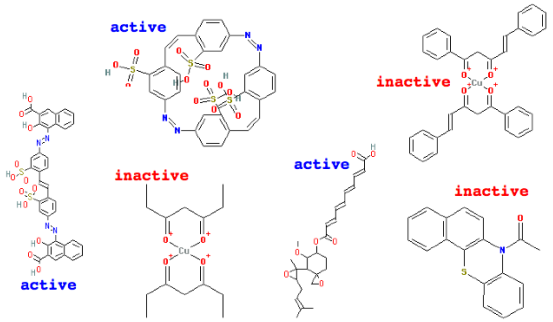
Represent each molecule as a vector…

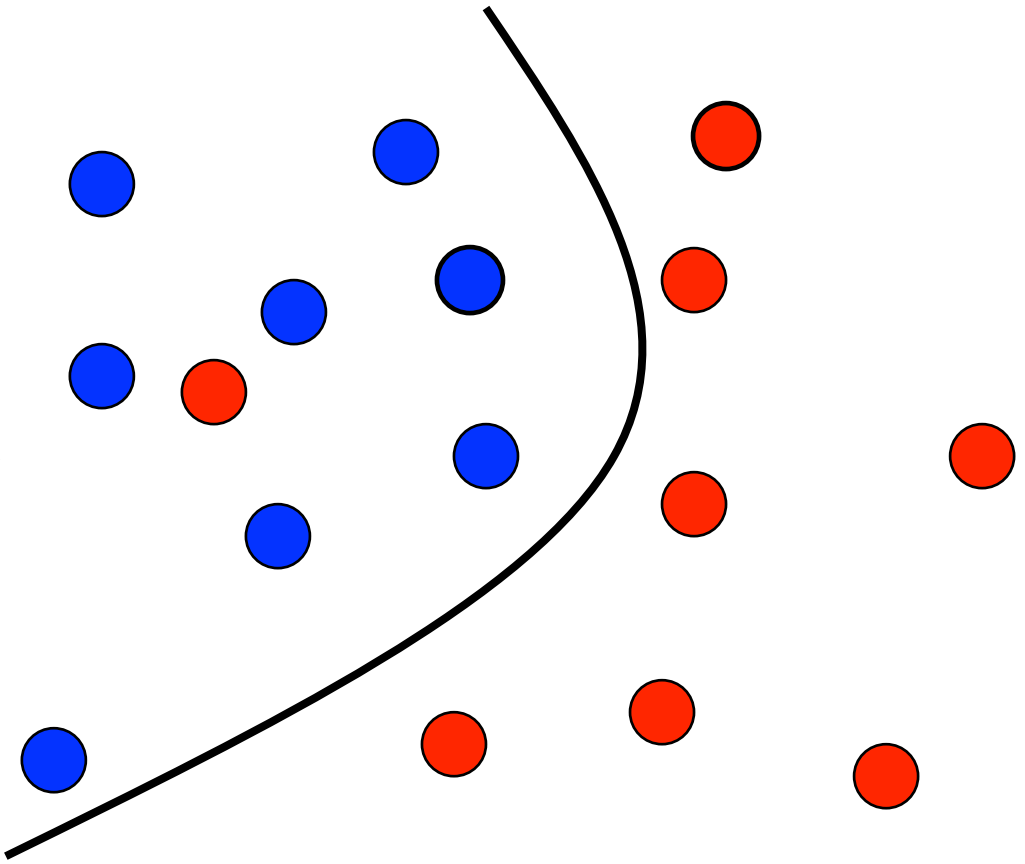






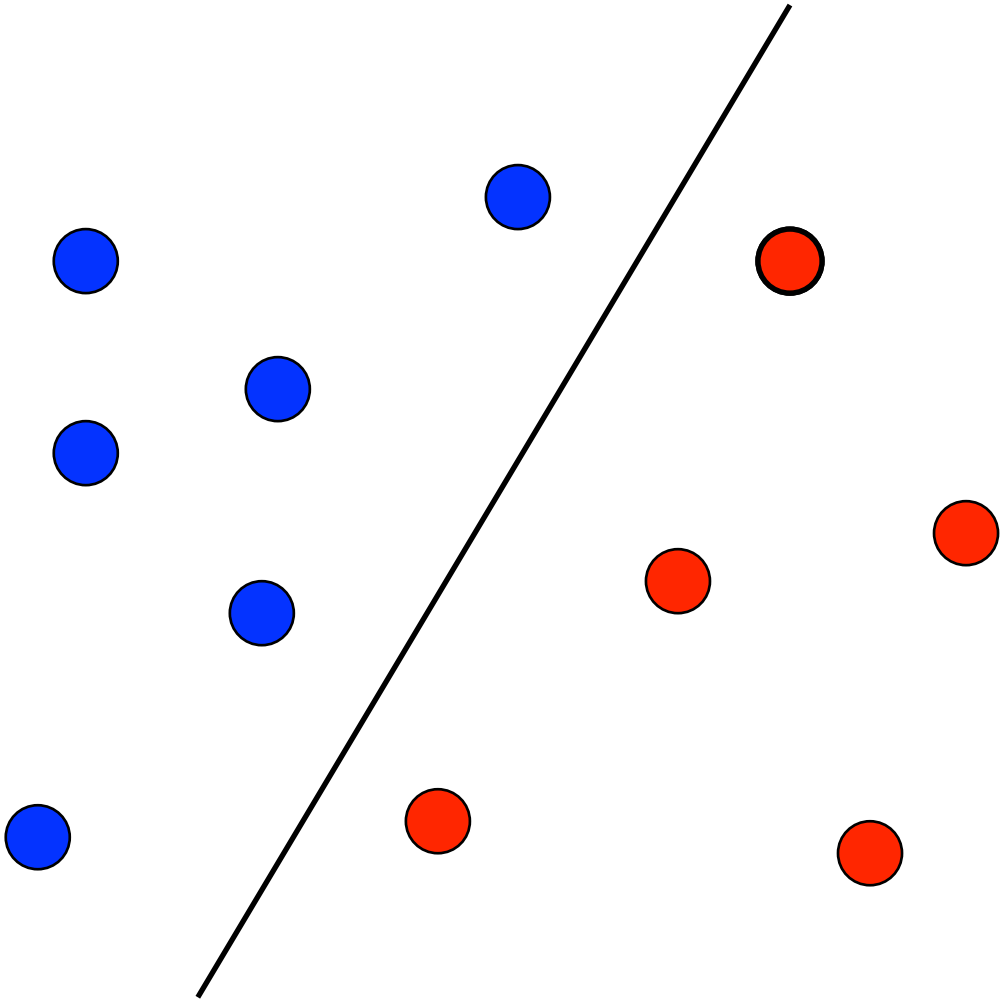
…and discriminate with machine learning



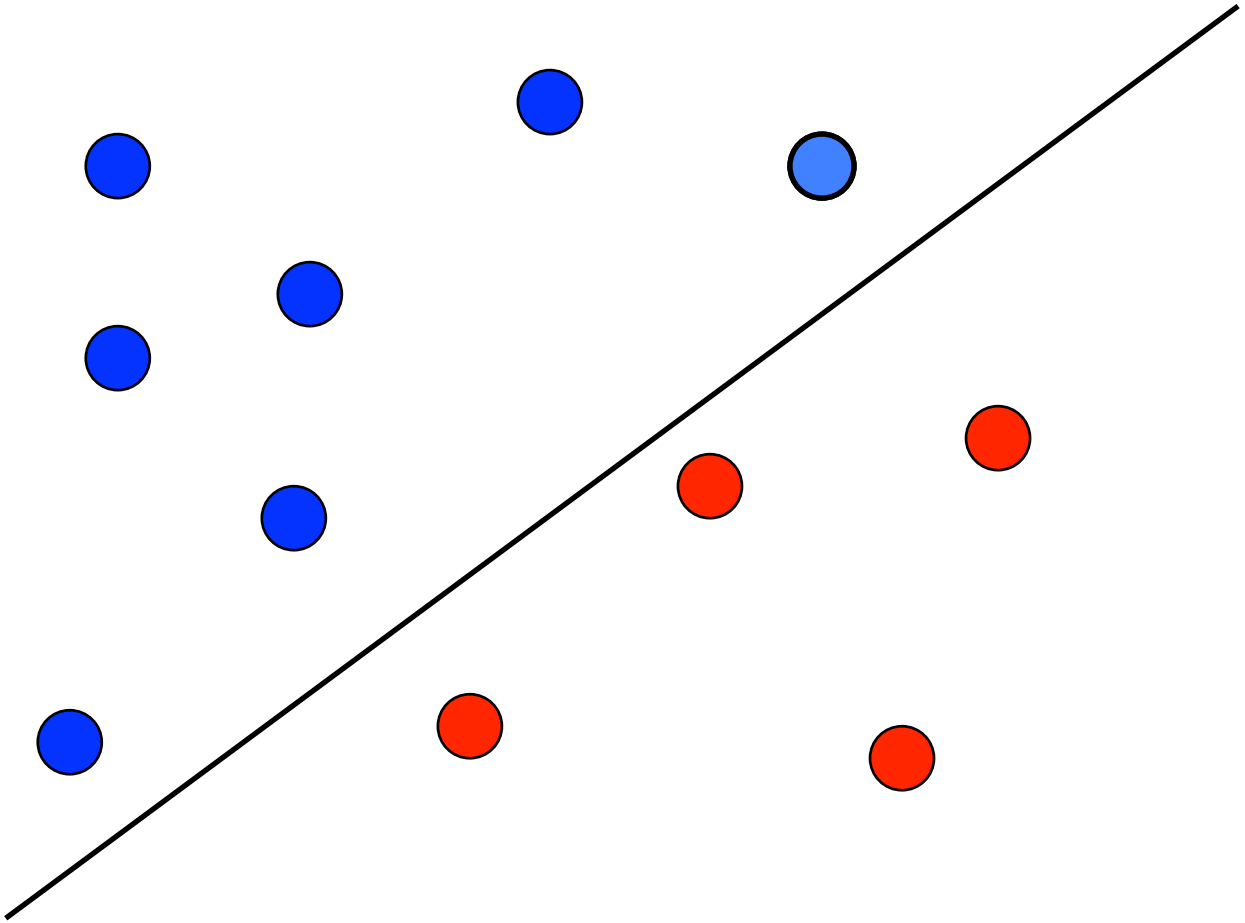


-LDA
-PLS
-Neural network
-Decision trees
-Nearest neighbour
-SVM, …

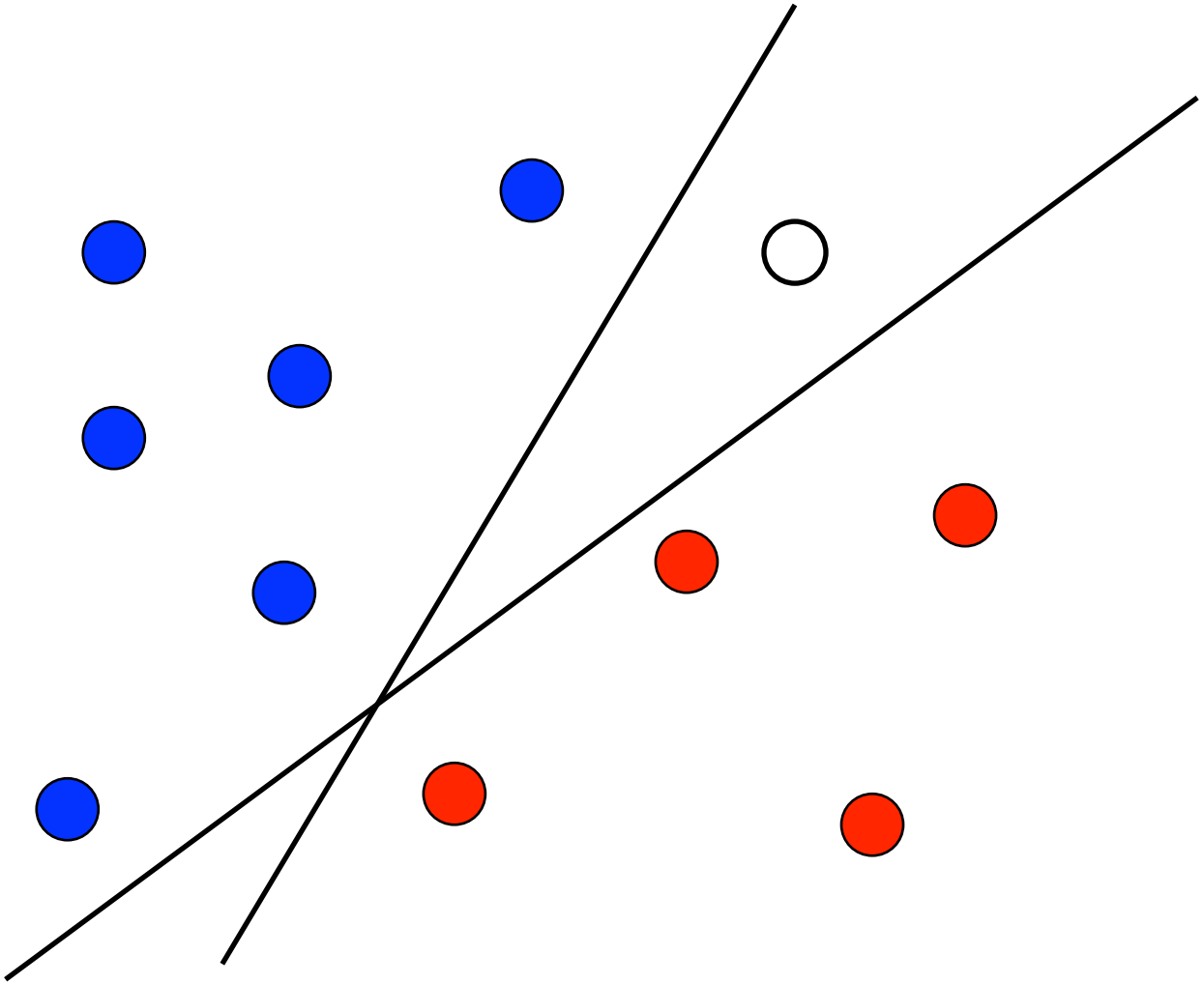
Linear classifier (simple case)



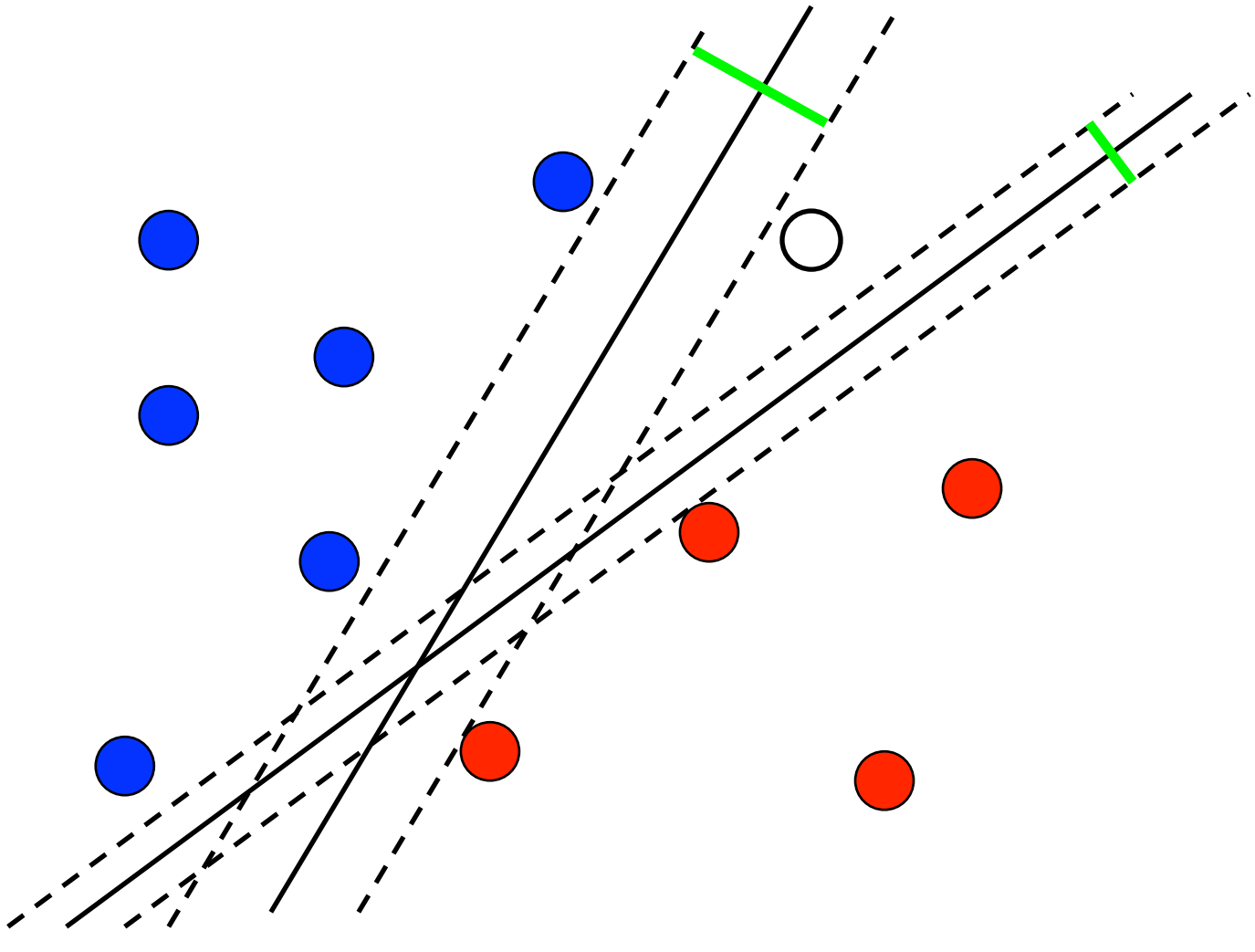
Another possibility…



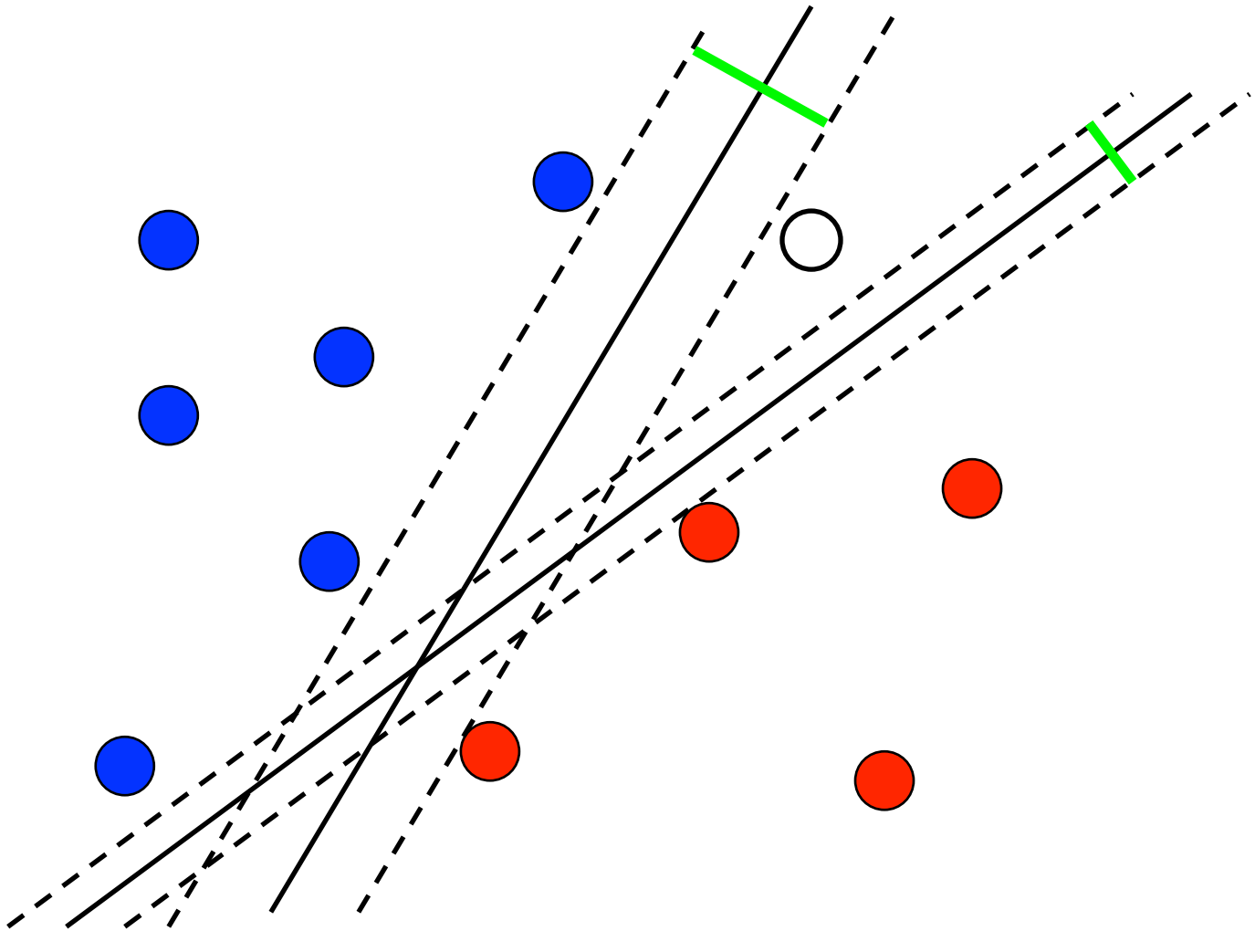
Which one is better?



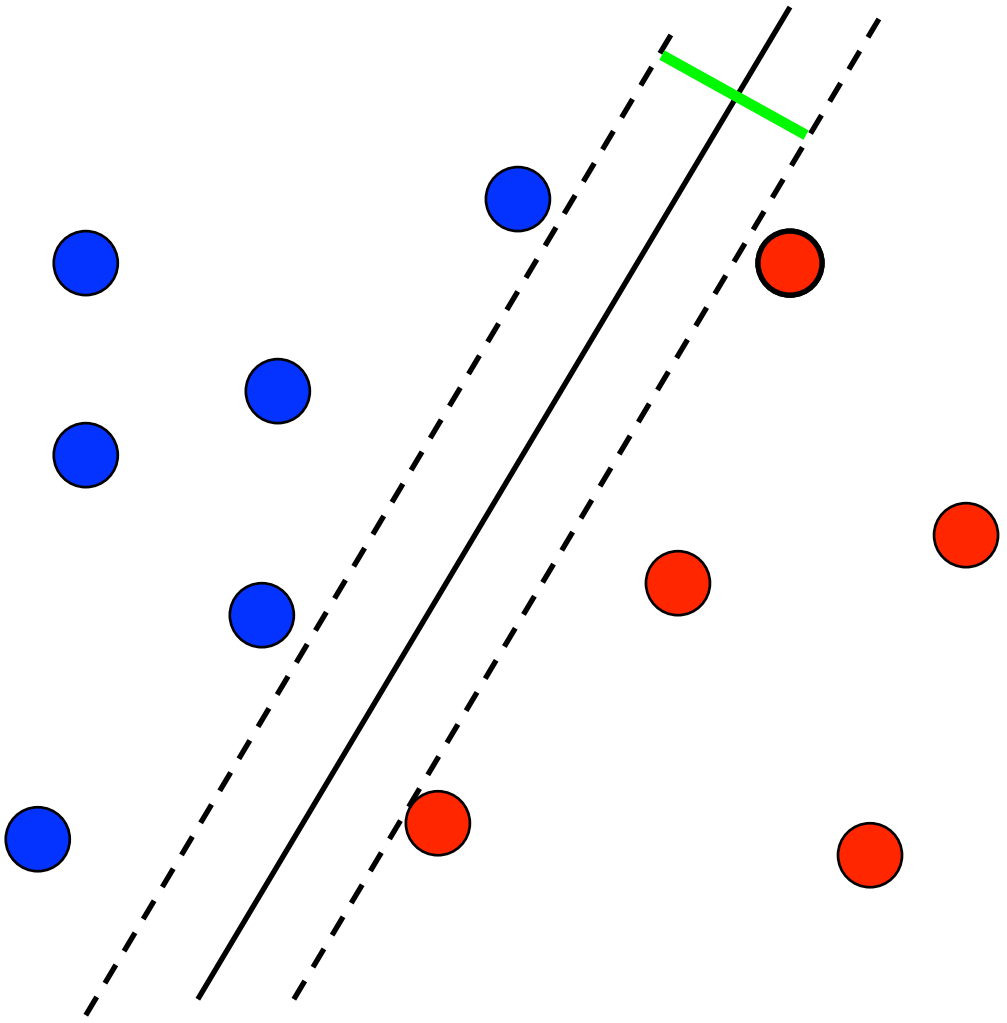
Vapnik's answer: margin



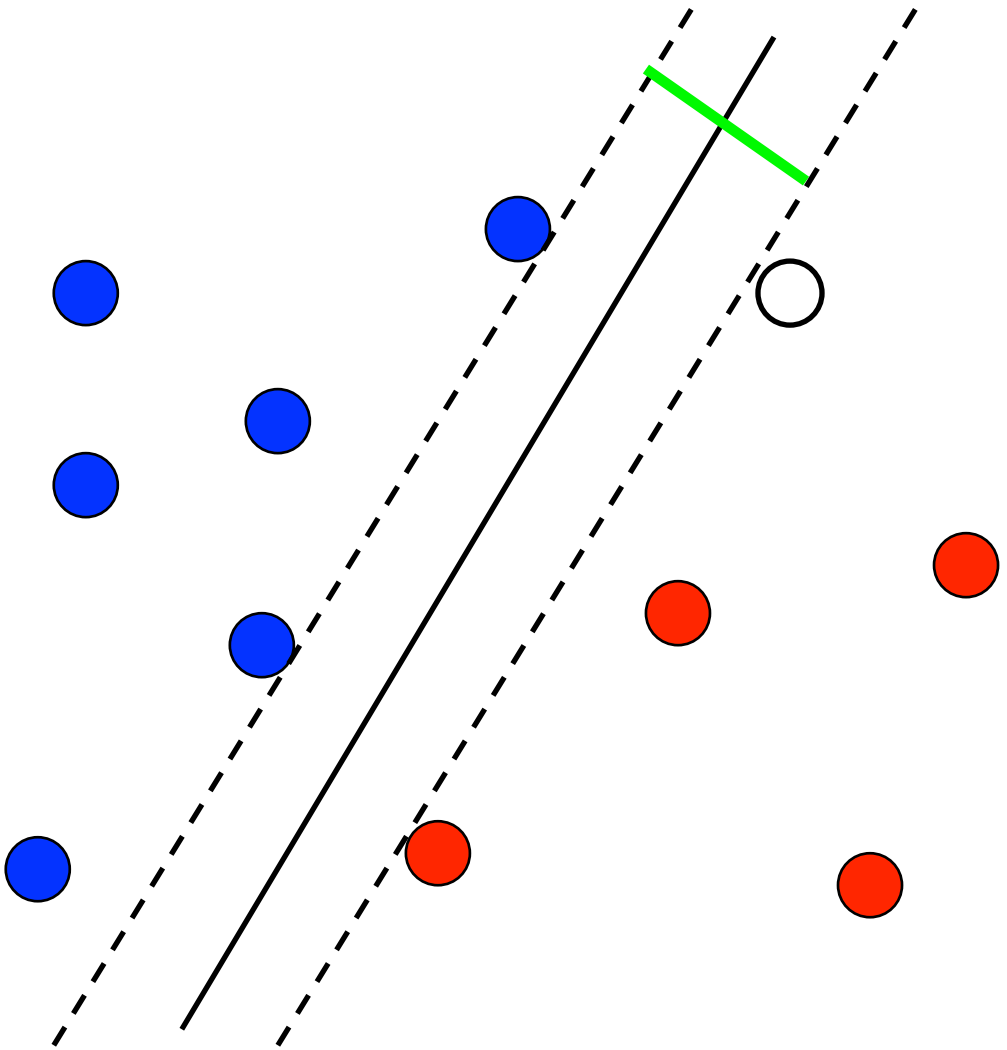
Vapnik's answer: margin



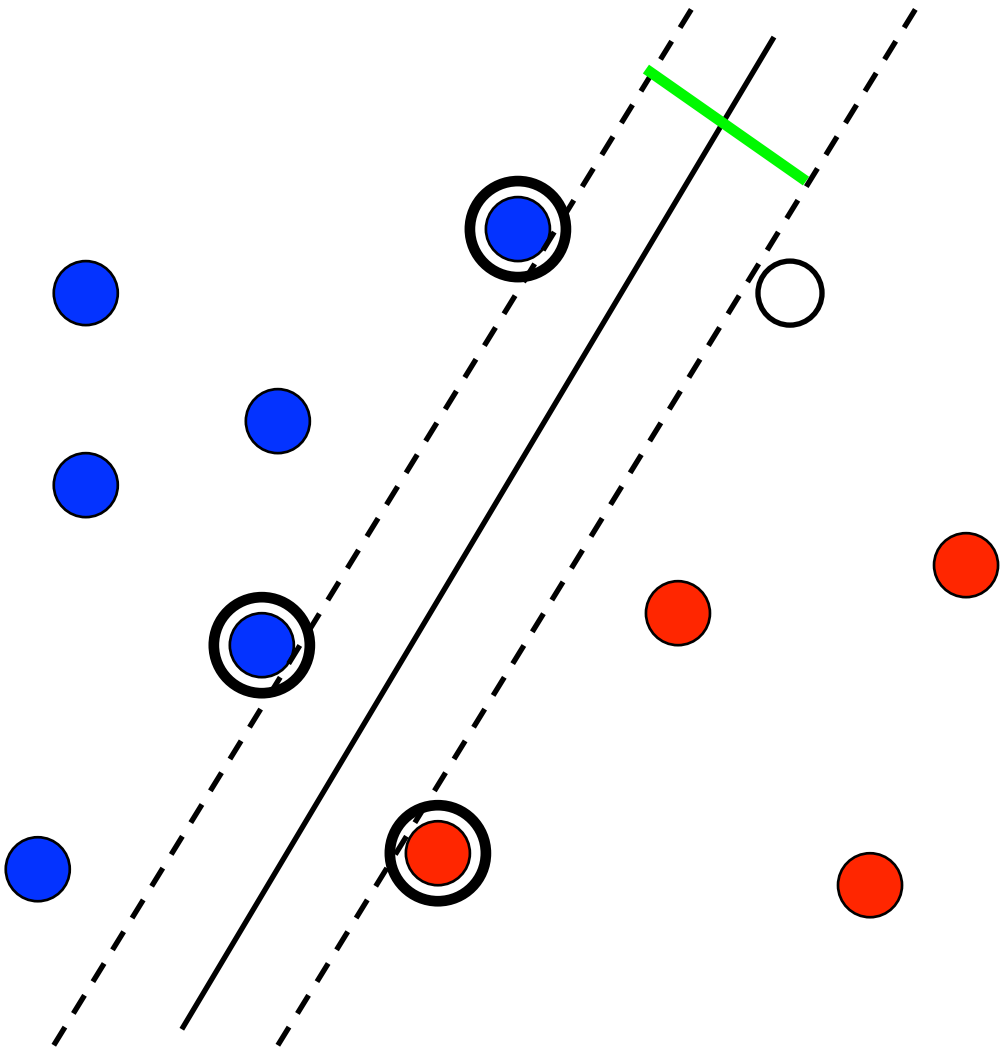
Vapnik's answer: margin



The best: largest margin



Support vectors

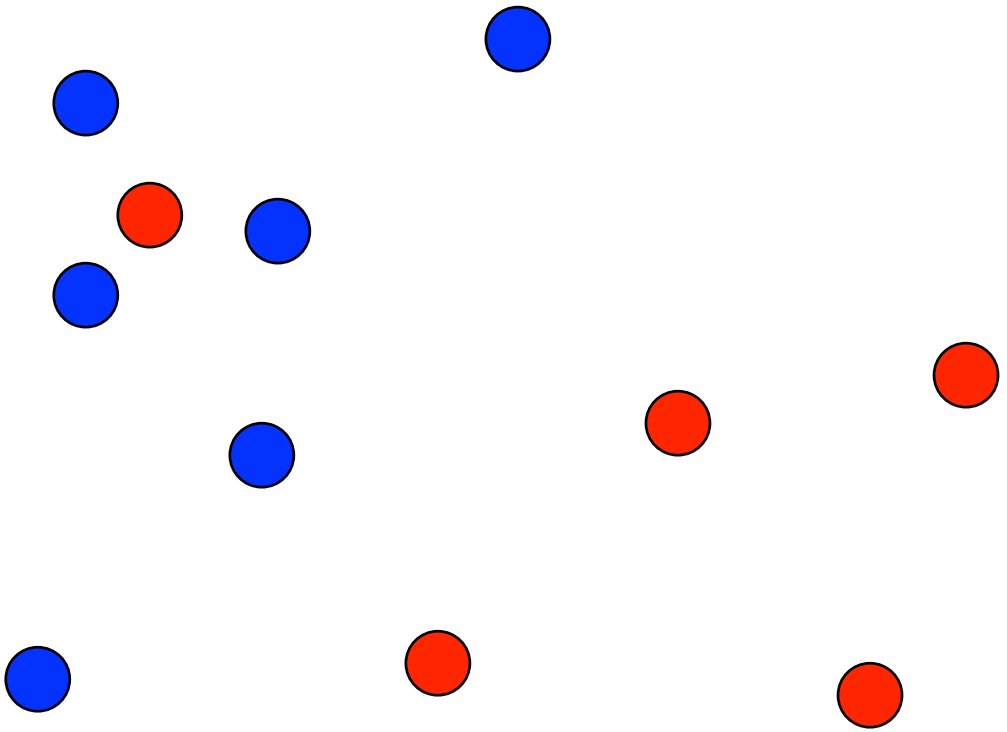


Implementation

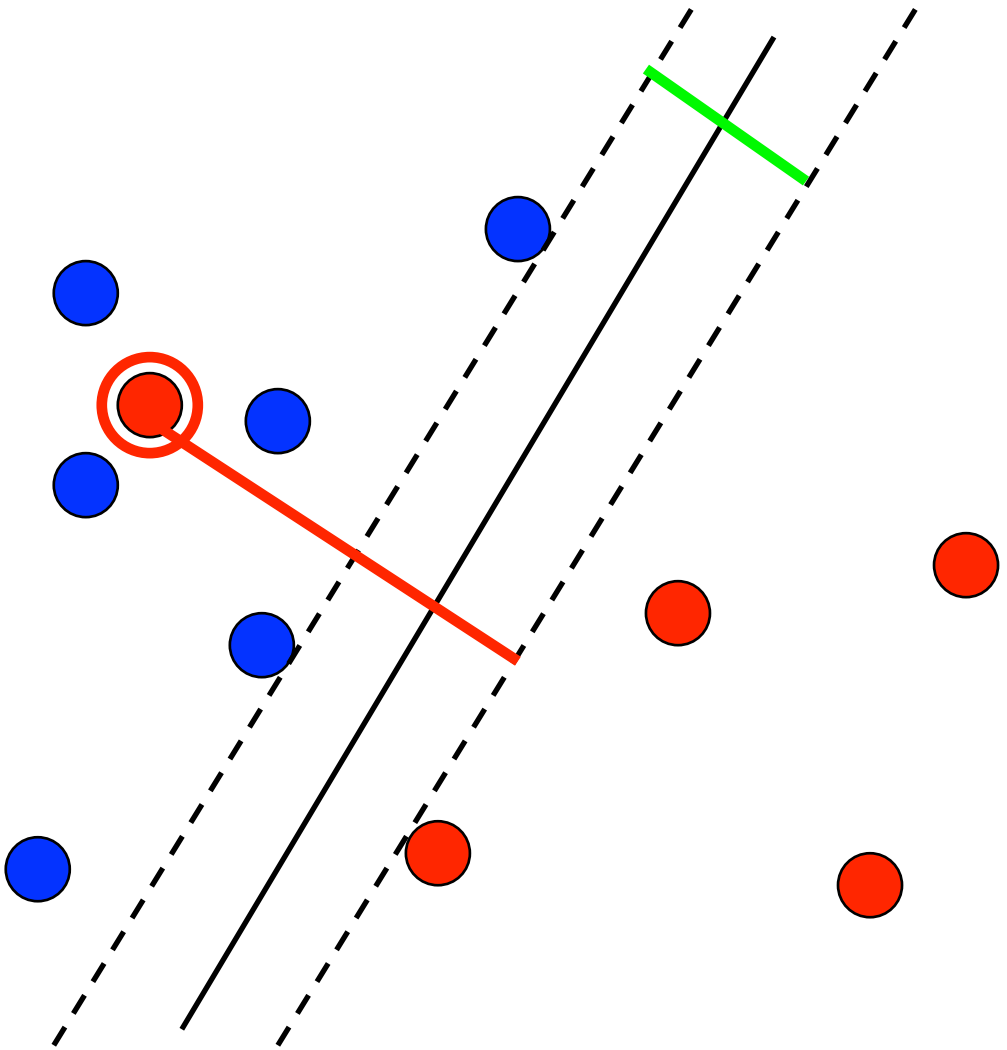
$$\max_f \{margin(f)\} \Longleftrightarrow \min_f \left\{ \frac{1}{margin(f)} \right\}$$

- The problem of finding the largest margin hyperplane is easy to solve (but not by yourself!)
- Unique solution, no local optimum (convex optimization problem)
- Only depends on the support vectors

New problem



New problem



Soft-margin SVM

- Find a trade-off between:
  - Large margin
  - Few misclassification
- Mathematically:

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

- Still easy to solve (for a good choice of « error »). C is a parameter.

An interesting property

- To train a SVM we just need the matrix of pairwise distances:

$$D_{i,j} = ||X_i - X_j||^2$$

- The predictor has the form:

$$f(X) = \sum_{i \in SV} w_i ||X_i - X||^2$$

Generalization (*Kernel trick)*

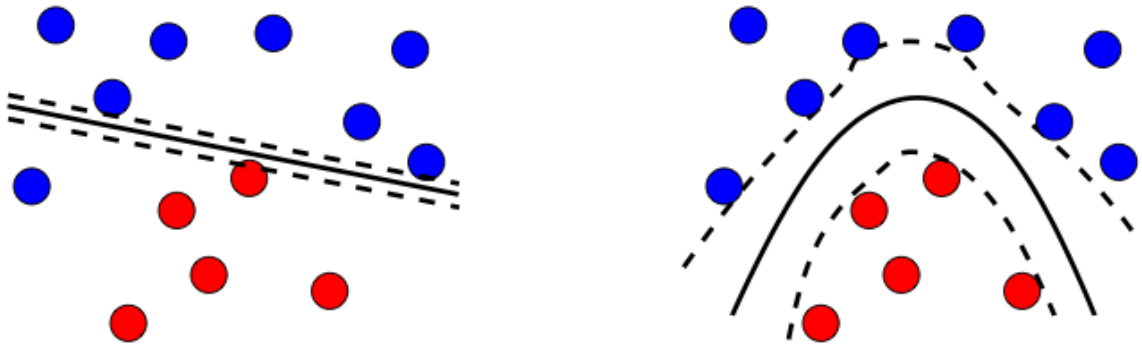
- Take a distance d(X,X')
- Train a SVM from the matrix of pairwise distances:

$$D_{i,j} = d(X_i, X_j)^2$$

- The predictor now is:

$$f(X) = \sum_{i \in SV} w_i d(X_i, X)^2$$

Example: nonlinear SVM



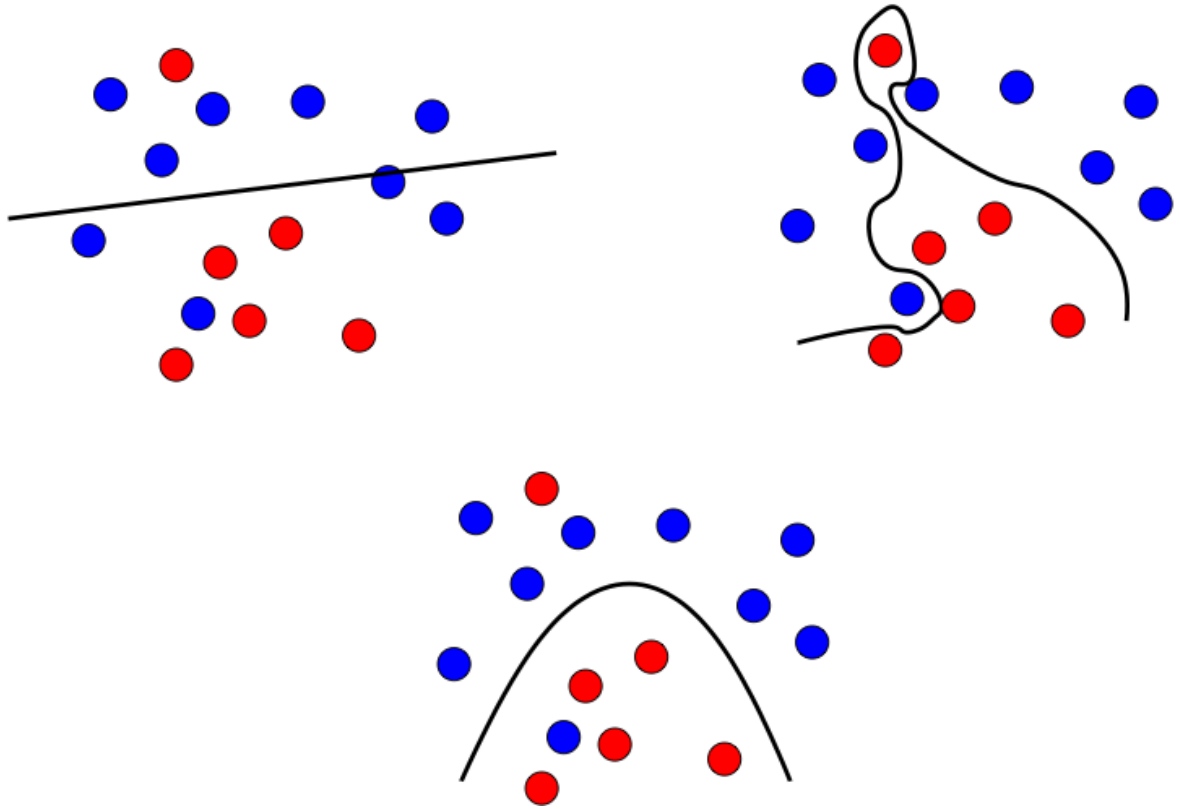
- Take a Gaussian distance:

$$d(X, X')^2 = 1 - \exp\left(-\frac{||X - X'||^2}{2\sigma^2}\right)$$

- We can then learn nonlinear predictors:

$$f(X) = \sum_{i \in SV} w_i \exp\left(-\frac{||X - X_i||^2}{2\sigma^2}\right) + cte$$

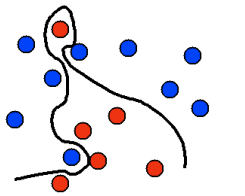
The fundamental trade-off: regularity (margin) vs error

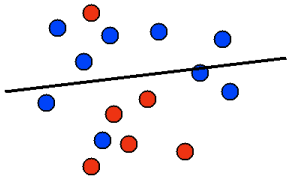


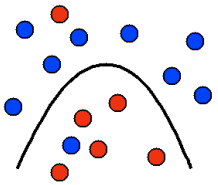
C controls the trade-off

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

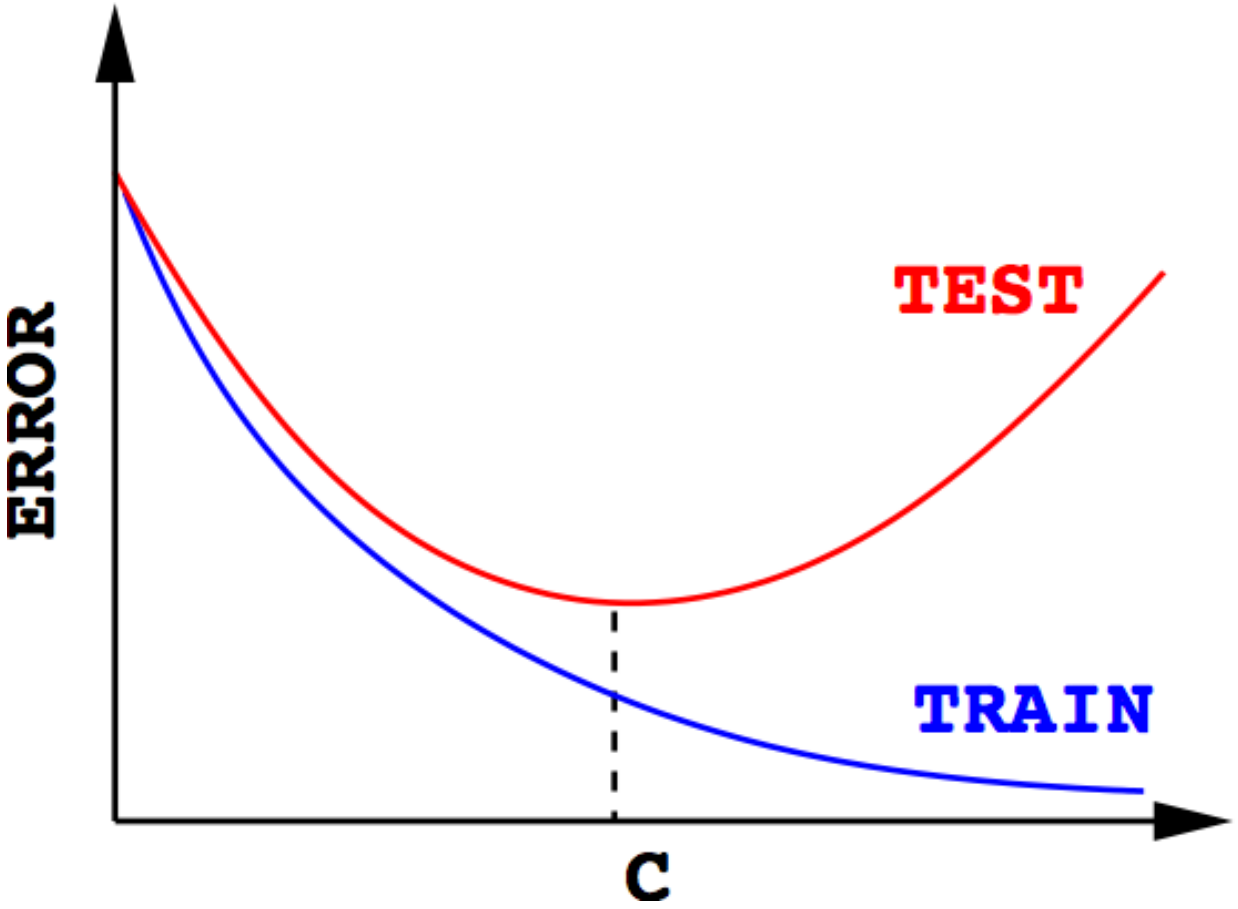
- Large C :
  - makes few errors
- Small C :
  - ensure a large margin
- Intermediate C:
  - finds a trade-off







Why it is important to care about the trade-off



Don’t trust the default « C=1 » !

Choosing C

- Split the annotated data in 2: training / validation
- Train a predictor on the training set
- Evaluate the performance on the validation set
- Choose C to minimize the validation error
- (you may repeat all this several times -> cross-validation)

SVM in practice
(eg: libsvm with Python)

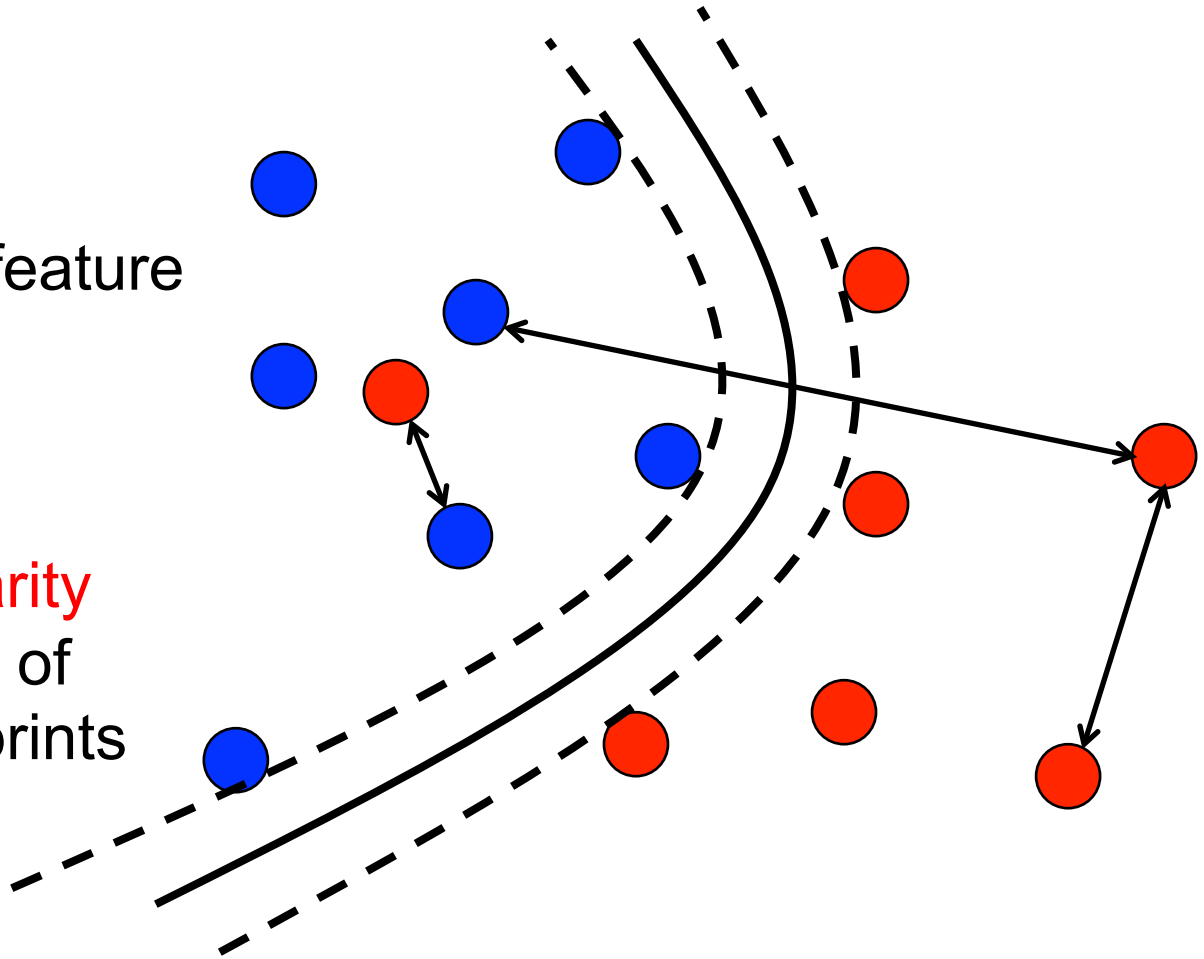
```
1> from svm import *
2> param = svm_parameter(kernel_type=LINEAR,C=10)
3> prob = svm_problem([1,-1],[[1,0,1],[-1,0,-1]])
4> m = svm_model(prob, param)
5> r = m.predict([1, 1, 1])
```

SVM summary

- Large margin

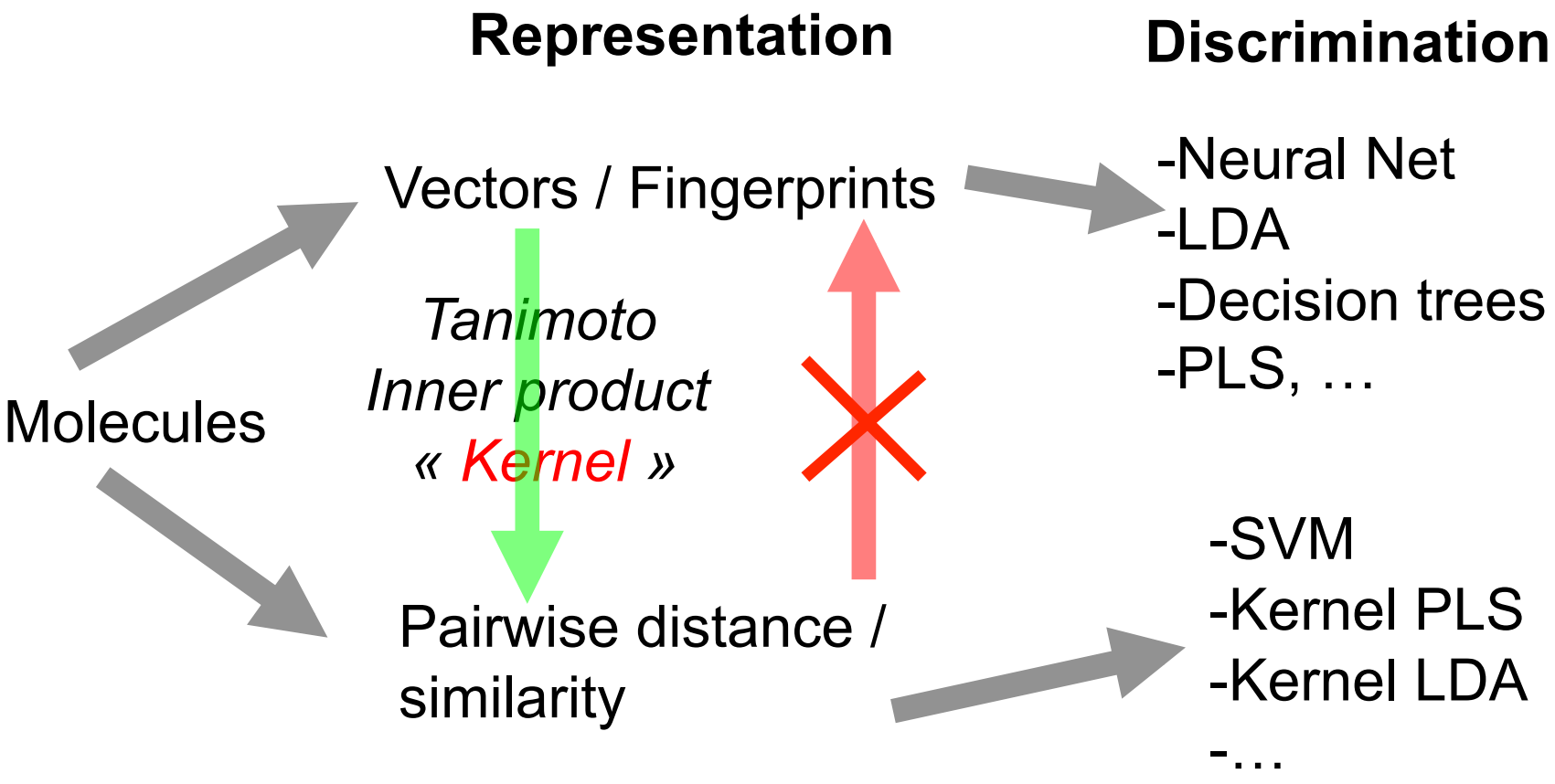
- Nonlinear, no feature selection

- Need pairwise distance / similarity as input instead of vectors / fingerprints

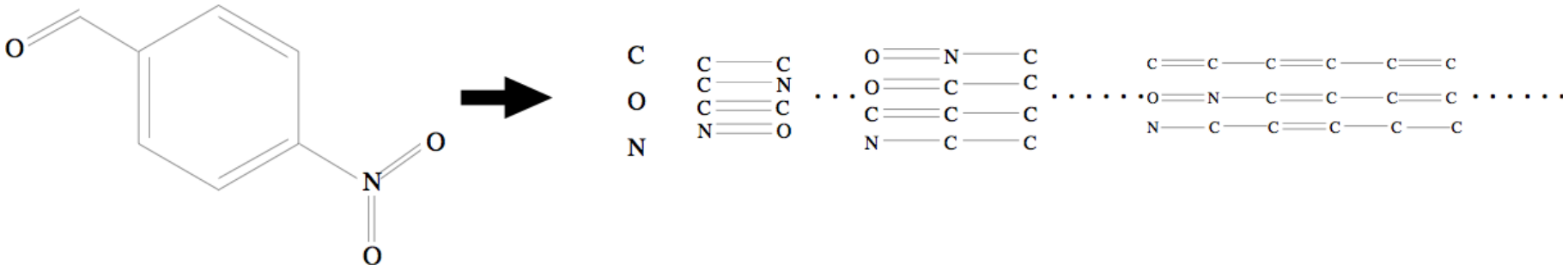


Kernels for small molecules

From descriptors to similarities



2D fragment kernels (walks)



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

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

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

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

Kashima et al. (2003), Gärtner et al. (2003)

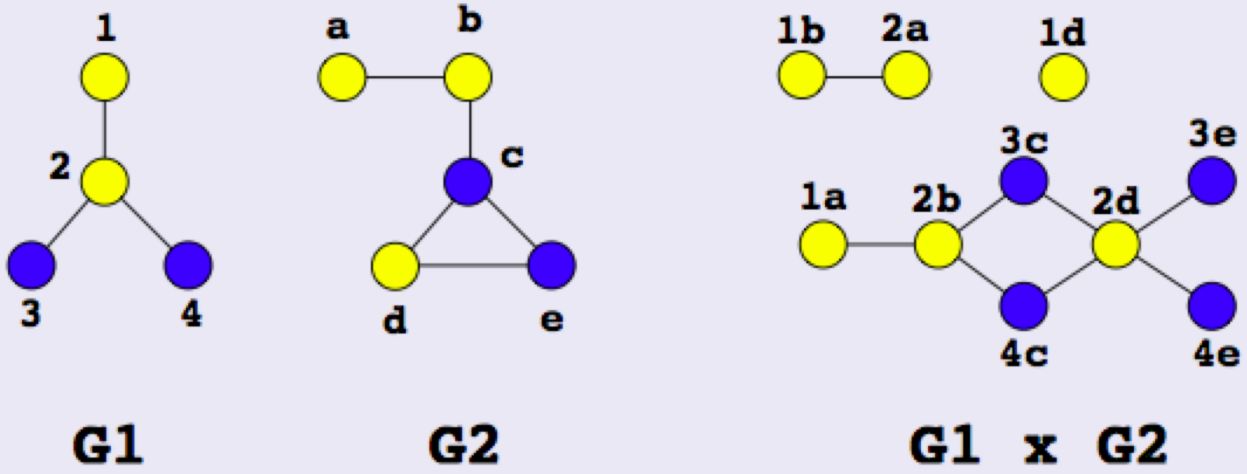
Properties of the 2D fragment kernel

- Corresponds to a fingerprint of infinite size
- Can be computed efficiently in O(|x|^3 |x'|^3) (much faster in practice)
- Solves the problem of clashes and memory storage (fingerprints are not computed explicitly)

Kashima et al. (2003), Gärtner et al. (2003)

2D kernel computational trick

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

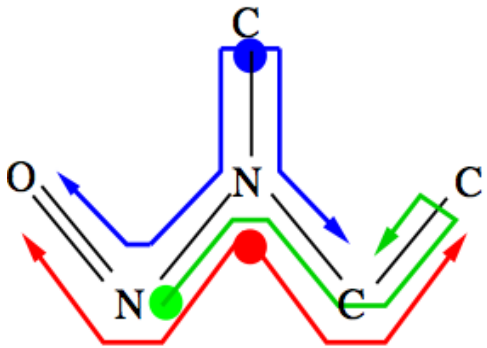


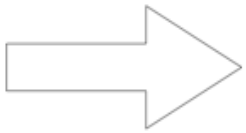
- The infinite counting can be factorized

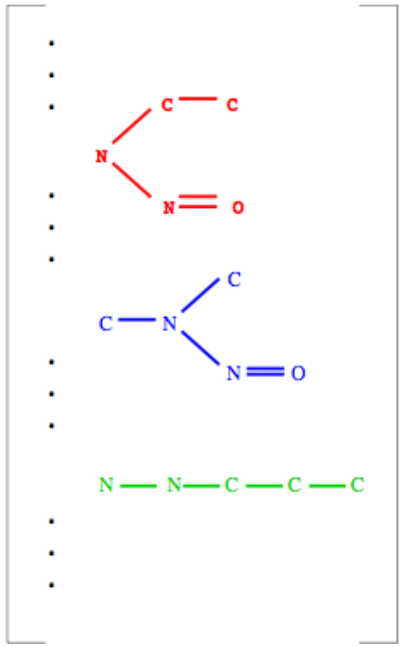
$$\lambda A + \lambda^2 A^2 + \lambda^3 A^3 + \ldots = (I - \lambda A)^{-1} - I.$$

Extension: subtree patterns

« All subtree patterns »





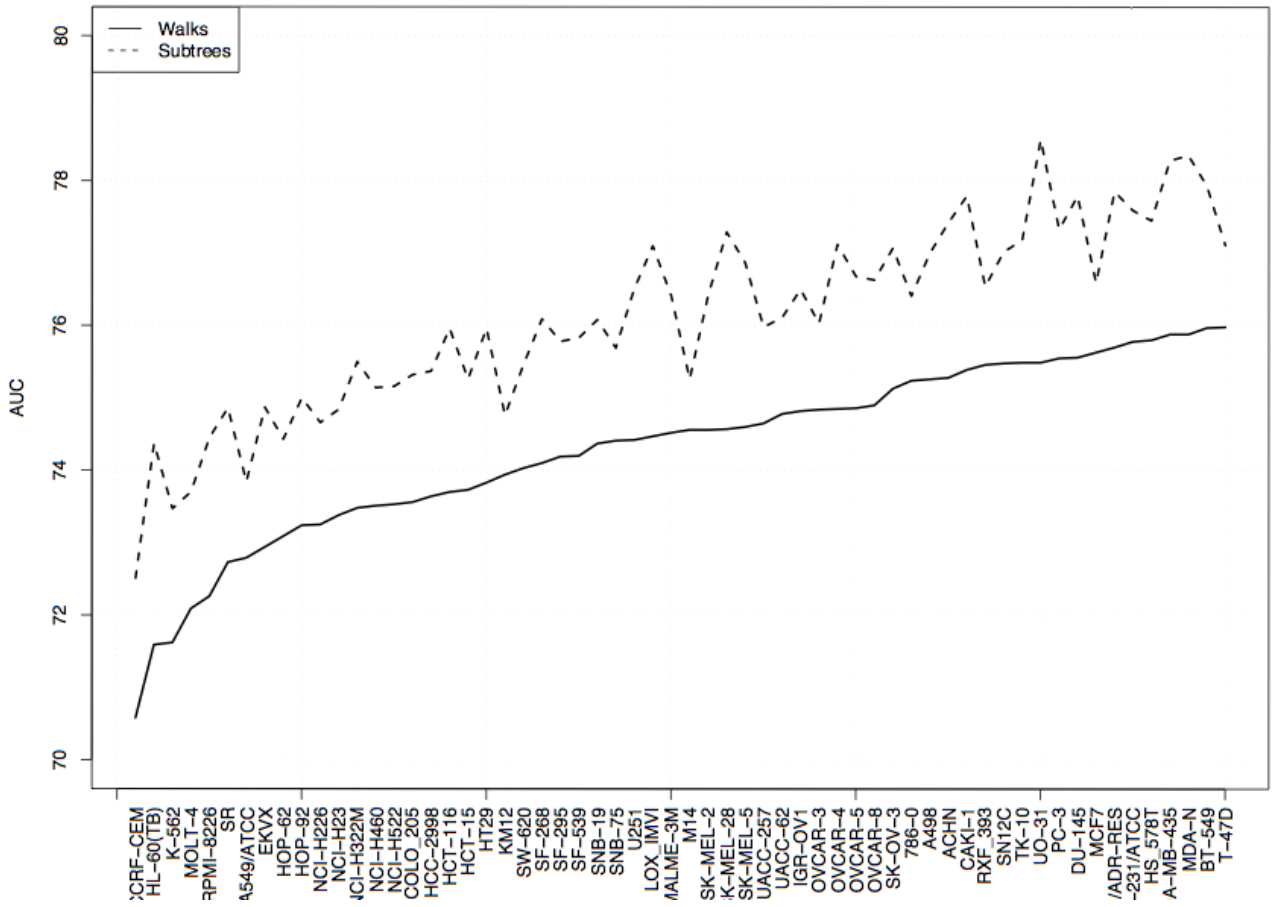


Mahé and V., *Mach. Learn*, 2009.

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

Ramon et al. (2004), Mahé & V. (2009)

2D subtree vs walk kernel



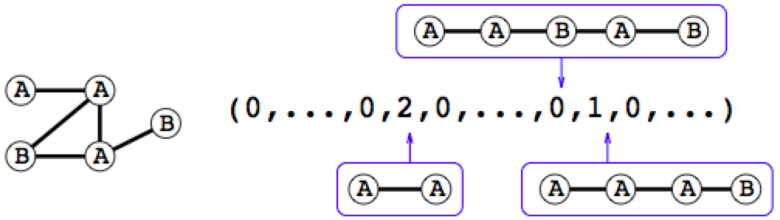
NCI 60 dataset

Mahé & V. (2009)

Other 2D kernels

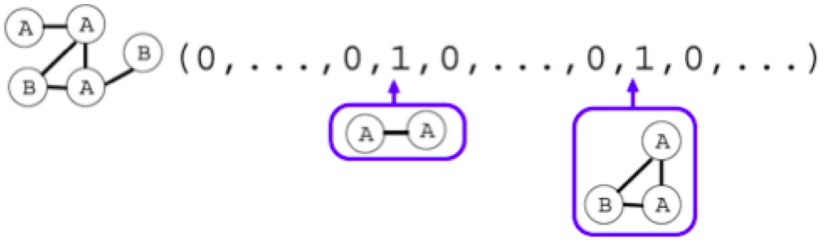
- Indexing by all **shortest paths**

(Borgwardt & Kriegel 2005)



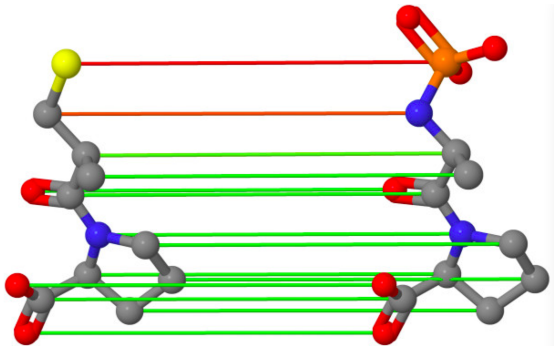
- Indexing by all **small subgraphs**

(Shervashidze et al. 2009)

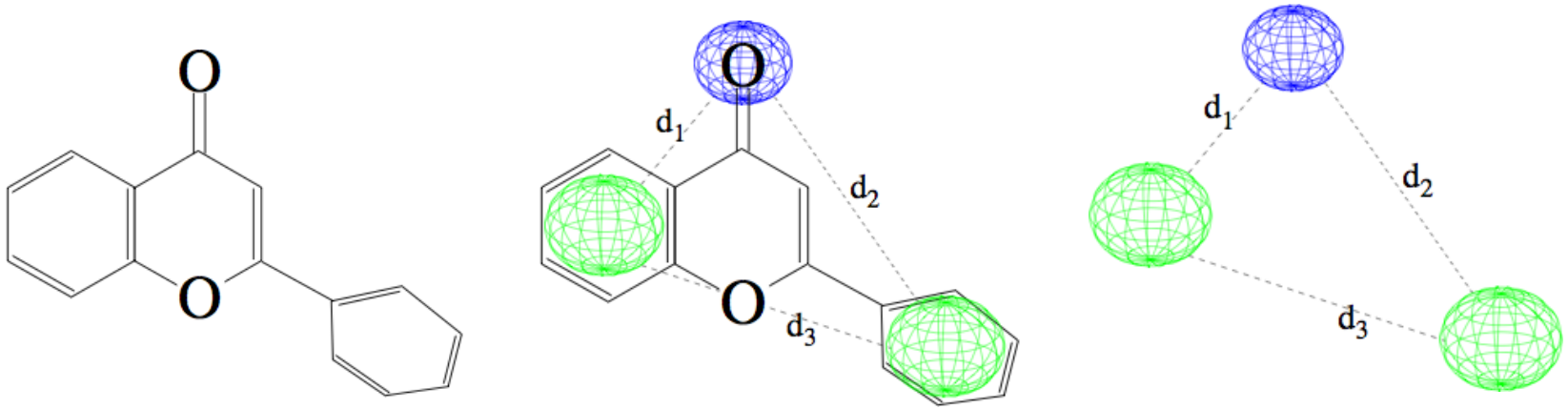


- **Optimal assignment** kernel

(Fröhlich et al. 2005)



3-point pharmacophores



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

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

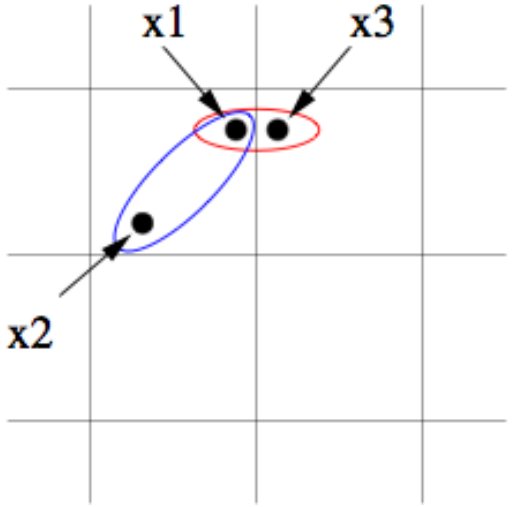
Mahé et al., *J. Chem. Inf. Model.*, 2006.

3D fingerprint kernel

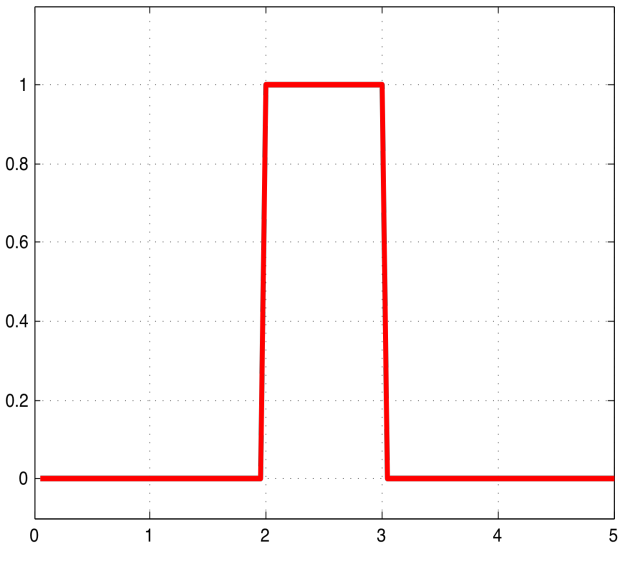
1. Discretize the space of pharmacophores $\mathcal{T}$ (e.g., 6 atoms or groups of atoms, 6-7 distance bins) into a finite set $\mathcal{T}_d$
2. Count the number of occurrences $\phi_t(x)$ of each pharmacophore bin t in a given molecule x, to form a pharmacophore fingerprint.

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

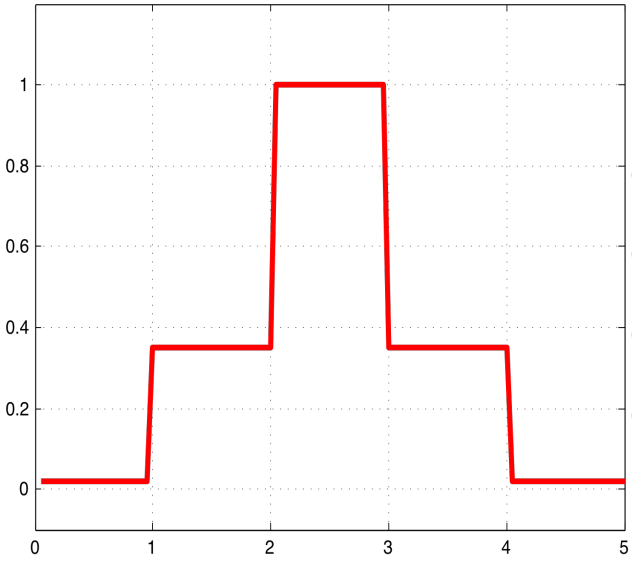
$$K(x, x') = \sum_{t \in \mathcal{T}_d} \phi_t(x) \phi_t(x') \ .$$



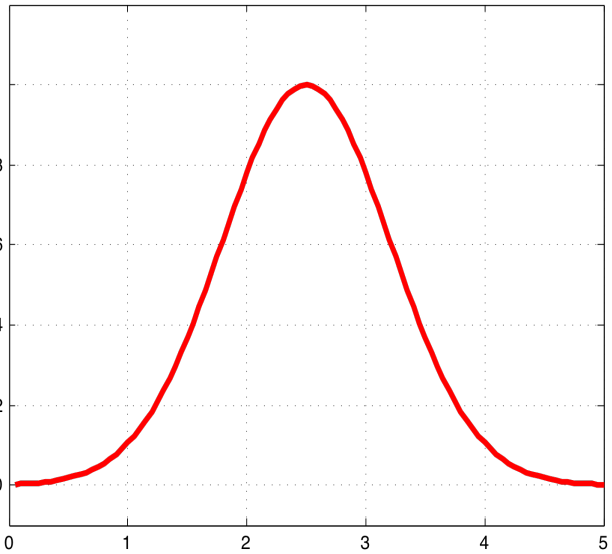
Removing discretization artifacts



3D Fingerprint



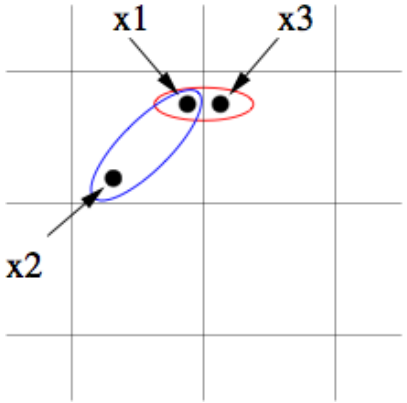
3D Fuzzy Fingerprint



3D Kernel

From the fingerprint kernel to the pharmacophore kernel

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



$$
K(x,y) = \sum_{p_x \in \mathcal{P}(x)} \sum_{p_y \in \mathcal{P}(y)} \exp\left(-\gamma ||p_x - p_y||^2\right)
$$

Experiments

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

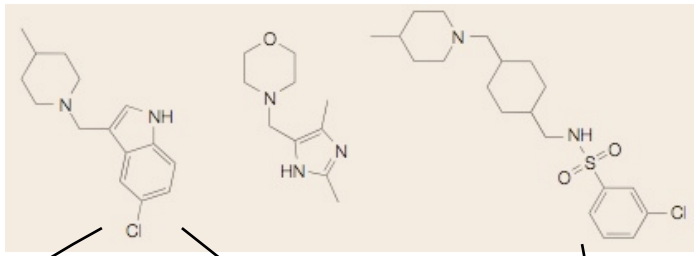
Kernel	BZR	COX	DHFR	ER
2D (Tanimoto)	71.2	63.0	76.9	77.1
3D fingerprint	75.4	67.0	76.9	78.6
3D not discretized	**76.4**	**69.8**	**81.9**	**79.8**

Mahé et al., *J. Chem. Inf. Model.*, 2006.

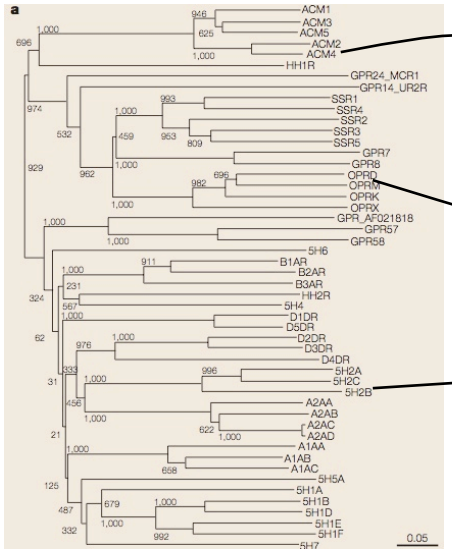
Towards *in silico* chemogenomics

Chemogenomics

Chemical space



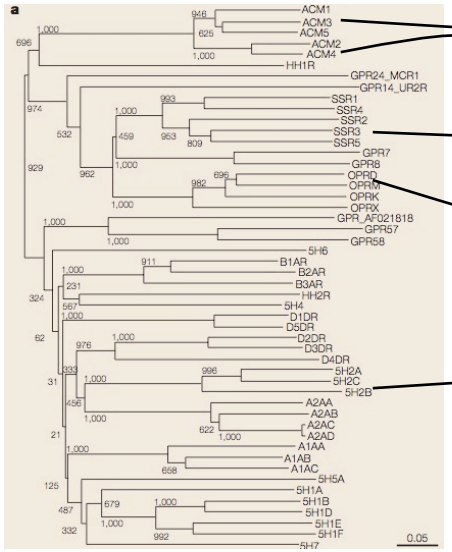
Target family

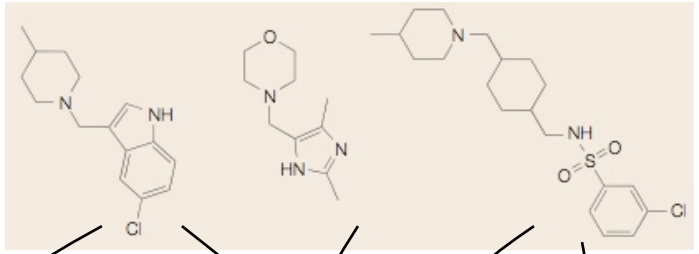


In silico Chemogenomics

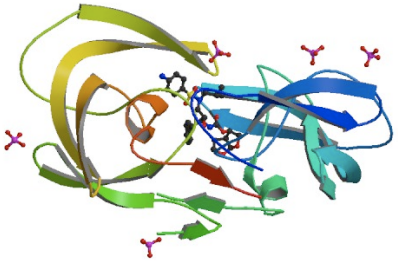
Chemical space

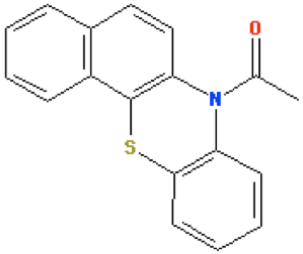
Target family





Fingerprint for a (target,molecule) pair?

t= 

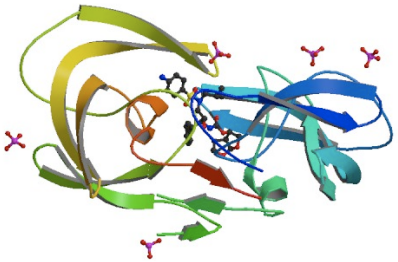
c = 

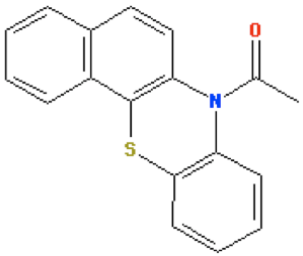
$\Phi_{tar}(t)$ =
- -Sequence
- -Structure
- -Evolution
- -Expression
- -…

$\Phi_{lig}(c)$ =
- -2D
- -3D
- -Pharmacophore
- -MW, logP, …

$$\Phi(c,t) = ???$$

Fingerprint for a (target,molecule) pair?

T= 

c = 

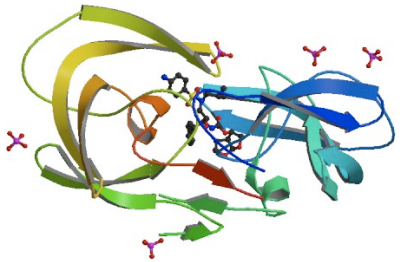
$\Phi_{tar}(t)$ = -Sequence -Structure -Evolution -Expression -…

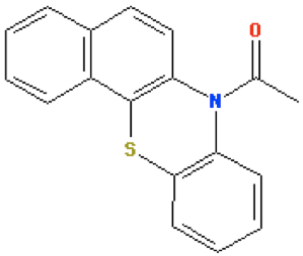
$\Phi_{lig}(c)$ = -2D -3D -Pharmacophore -logP, …

$$\Phi(c,t) = \Phi_{lig}(c) \otimes \Phi_{tar}(t)$$

10^6 10^3 10^3

Similarity for (target,molecule) pairs

t= 

c = 

$K_{target}(t,t')$ = -Sequence -Structure -Evolution -Expression -…

$K_{ligand}(c,c')$ = -2D -3D -Pharmacophore -logP, …

$$K\left((c,t),(c',t')\right) = K_{target}(t,t') \times K_{ligand}(c,c')$$

Summary: SVM for chemogenomics

1. Choose a kernel (similarity) for targets
2. Choose a kernel (similarity) for ligands
3. Train a SVM model with the product kernel for (target/ligand) pairs

Application: virtual screening of GPCR

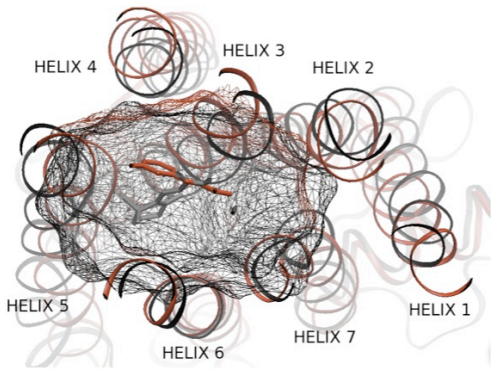
Data: GLIDA database filtered for drug-like compounds

- 2446 ligands
- 80 GPCR
- 4051 interactions
- *4051 negative interactions generated randomly*

Ligand similarity

-2D Tanimoto

-3D pharmacophore



Target similarities

-0/1 Dirac (no similarity)

-Multitask (uniform similarity)

-GLIDA's hierarchy similarity

-Binding pocket similarity (31 AA)

(Jacob et al., *BMC Bioinformatics*, 2008)

Results (mean accuracy over GPCRs)

5-fold cross-validation

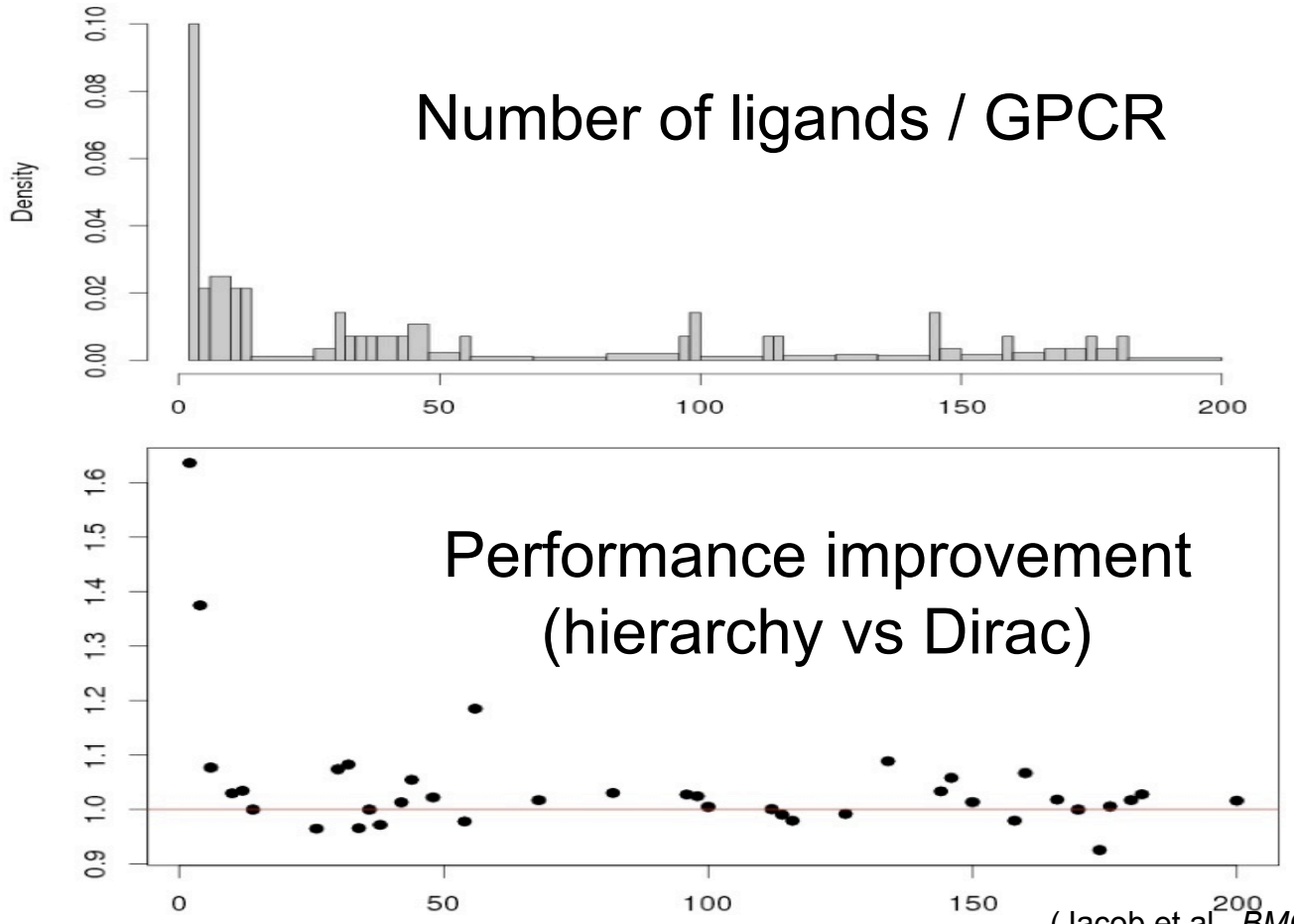
$K_{tar} \backslash K_{lig}$	2D Tanimoto	3D pharmacophore
Dirac	86.2 ± 1.9	84.4 ± 2.0
multitask	88.8 ± 1.9	85.0 ± 2.3
hierarchy	93.1 ± 1.3	88.5 ± 2.0
binding pocket	90.3 ± 1.9	87.1 ± 2.3

Orphan GPCRs setup

$K_{tar} \backslash K_{lig}$	2D Tanimoto	3D pharmacophore
Dirac	50.0 ± 0.0	50.0 ± 0.0
multitask	56.8 ± 2.5	58.2 ± 2.2
hierarchy	77.4 ± 2.4	76.2 ± 2.2
binding pocket	78.1 ± 2.3	76.6 ± 2.2

(Jacob et al., *BMC Bioinformatics*, 2008)

Influence of the number of known ligands

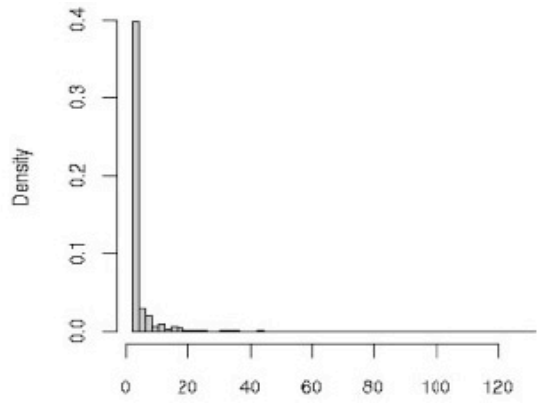


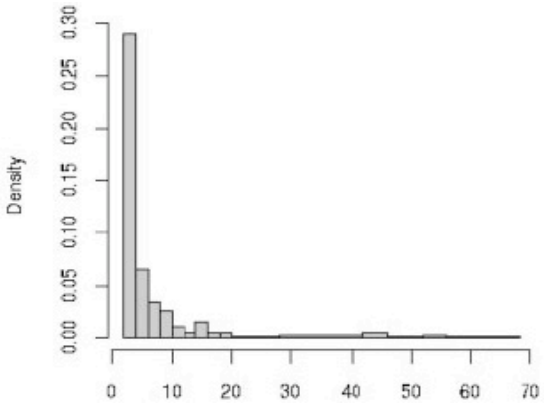
(Jacob et al., *BMC Bioinformatics*, 2008)

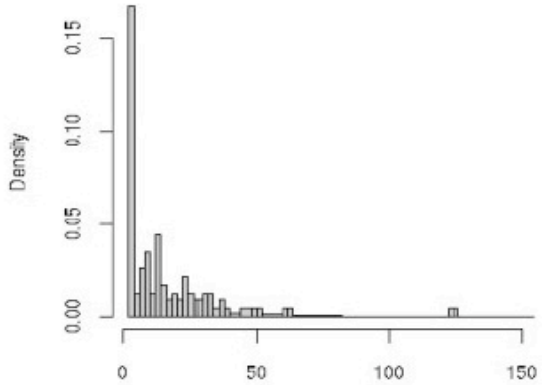
Screening of enzymes, GPCRs, ion channels

Data: KEGG BRITE database, redundancy removed

Enzymes	**GPCRs**	**Ion channels**
-675 targets	-100 targets	-114 targets
-524 molecules	-219 molecules	-462 molecules
-1218 interactions	-399 interactions	-1165 interactions
-1218 negatives	-399 negatives	-1165 negatives







(Jacob and V., *Bioinformatics*, 2008)

Results (mean AUC)

10-fold CV

K_{tar} \ Target	Enzymes	GPCR	Channels
Dirac	0.646±0.009	0.750±0.023	0.770±0.020
Multitask	0.931±0.006	0.749±0.022	0.873±0.015
Hierarchy	0.955±0.005	0.926±0.015	0.925±0.012
Mismatch	0.725±0.009	0.805±0.023	0.875±0.015
Local alignment	0.676±0.009	0.824±0.021	0.901±0.013

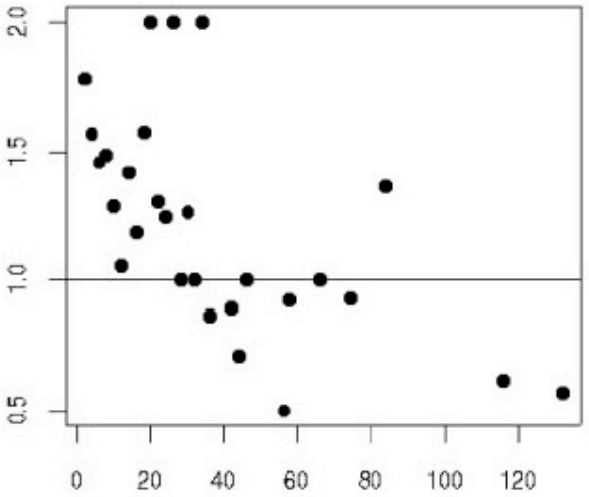
Orphan setting

K_{tar} \ Target	Enzymes	GPCR	Channels
Dirac	0.500±0.000	0.500±0.000	0.500±0.000
Multitask	0.902±0.008	0.576±0.026	0.704±0.026
Hierarchy	0.938±0.006	0.875±0.020	0.853±0.019
Mismatch	0.602±0.008	0.703±0.027	0.729±0.024
Local alignment	0.535±0.005	0.751±0.025	0.772±0.023

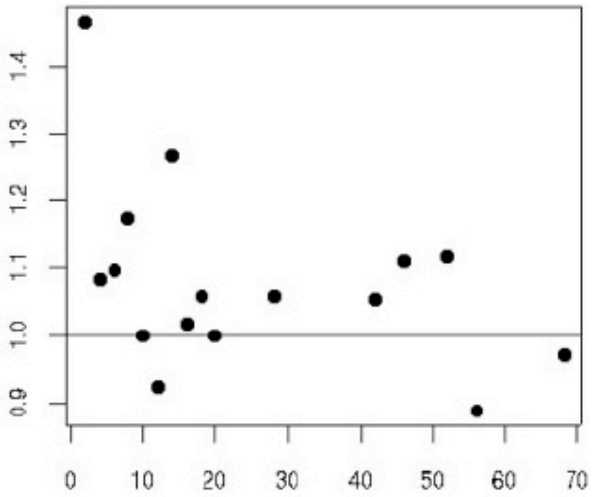
(Jacob and V., *Bioinformatics*, 2008)

Influence of the number of known ligands

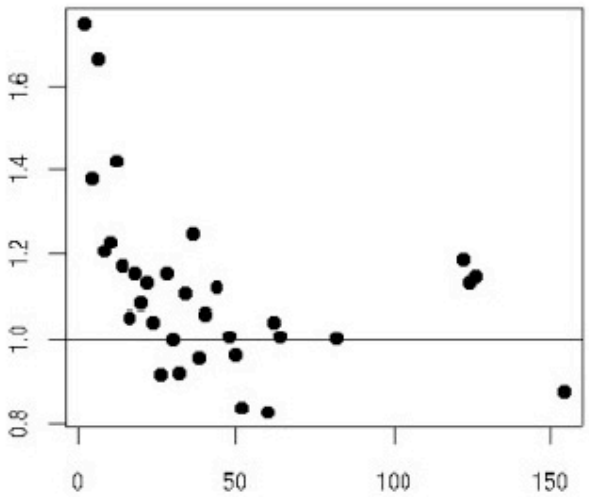
Enzymes



GPCRs



Ion channels



Relative improvement : hierarchy vs Dirac

(Jacob and V., *Bioinformatics*, 2008)

Conclusion

- SVM offer state-of-the-art performance in many chemo- and bio-informatics applications
- The kernel trick is useful to
  - Work implicitly with **many features** without computing them (*2D fragment kernels)*
  - Work with **similarity measures** that cannot be derived from descriptors (*optimal alignment kernel)*
  - Relax the need for **discretization** (*3D pharmacophore kernel)*
  - Work in a **product space** (*chemogenomics*)
- Promising direction:
  - More kernels / Multiple kernel learning
  - Collaborative filtering in product space

Thank you !

Collaborators:
P. Mahé, L. Jacob, V. Stoven, B. Hoffmann

References :
`http://cbio.ensmp.fr/~jvert`

Open-source kernels for chemoinformatics:
`http://chemcpp.sourceforge.net`