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
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:

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

• 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

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

- 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)

```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

**Representation**
- Vectors / Fingerprints
- *Tanimoto* Inner product « Kernel »
- Pairwise distance / similarity

**Discrimination**
- Neural Net
- LDA
- Decision trees
- PLS, …
- SVM
- Kernel PLS
- Kernel LDA
- …
2D fragment kernels (walks)

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

$$\phi_1(x) = \left( \#(C), \#(O), \#(N), \ldots \right)^T$$
$$\phi_2(x) = \left( \#(C-C), \#(C=O), \#(C-N), \ldots \right)^T$$ etc...

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

$$K_{2D}(x, x') = \sum_{d=1}^{\infty} \lambda(d) \phi_d(x)^T \phi_d(x').$$

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)

\[ \lambda A + \lambda^2 A^2 + \lambda^3 A^3 + \ldots = (I - \lambda A)^{-1} - I. \]
Extension: subtree patterns

« All subtree patterns »

\[ I(v, n+1) = \sum_{R \subseteq N(v)} \prod_{v' \in R} \lambda_t(v, v') I(v', n) \]


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:

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

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:

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

![Diagram showing points in a 3D space with annotations x1, x2, and x3]
Removing discretization artifacts

3D Fingerprint

3D Fuzzy Fingerprint

3D Kernel
From the fingerprint kernel to the pharmacophore kernel

\[ K(x, y) = \sum_{t \in T_d} \phi_t(x)\phi_t(y) \]

\[ = \sum_{t \in T_d} \left( \sum_{p_x \in P(x)} 1(\text{bin}(p_x) = t) \right) \left( \sum_{p_y \in P(y)} 1(\text{bin}(p_y) = t) \right) \]

\[ = \sum_{p_x \in P(x)} \sum_{p_y \in P(y)} 1(\text{bin}(p_x) = \text{bin}(p_y)) \]

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

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

<table>
<thead>
<tr>
<th>Kernel</th>
<th>BZR</th>
<th>COX</th>
<th>DHFR</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D (Tanimoto)</td>
<td>71.2</td>
<td>63.0</td>
<td>76.9</td>
<td>77.1</td>
</tr>
<tr>
<td>3D fingerprint</td>
<td>75.4</td>
<td>67.0</td>
<td>76.9</td>
<td>78.6</td>
</tr>
<tr>
<td>3D not discretized</td>
<td>76.4</td>
<td>69.8</td>
<td>81.9</td>
<td>79.8</td>
</tr>
</tbody>
</table>

Towards *in silico* chemogenomics
In silico Chemogenomics

Chemical space

Target family
Fingerprint for a (target,molecule) pair?

\[ \Phi_{\text{tar}}(t) = \{ \text{Sequence, Structure, Evolution, Expression, ...} \} \]

\[ \Phi_{\text{lig}}(c) = \{ \text{2D, 3D, Pharmacophore, MW, logP, ...} \} \]

\[ \Phi(c, t) = ??? \]
Fingerprint for a (target, molecule) pair?

\[ \Phi_{tar}(t) = \begin{cases} 
\text{-Sequence} \\
\text{-Structure} \\
\text{-Evolution} \\
\text{-Expression} \\
\ldots 
\end{cases} \]

\[ \Phi_{lig}(c) = \begin{cases} 
\text{-2D} \\
\text{-3D} \\
\text{-Pharmacophore} \\
\text{-logP, ...} 
\end{cases} \]

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

\[ 10^6 \quad \quad 10^3 \quad \quad 10^3 \]
Similarity for (target,molecule) pairs

$$K_{\text{target}}(t, t') = \begin{cases} \text{-Sequence} \\ \text{-Structure} \\ \text{-Evolution} \\ \text{-Expression} \\ \ldots \end{cases}$$

$$K_{\text{ligand}}(c, c') = \begin{cases} \text{-2D} \\ \text{-3D} \\ \text{-Pharmacophore} \\ \text{-logP, ...} \end{cases}$$

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

<table>
<thead>
<tr>
<th>$K_{\text{tar}} \backslash K_{\text{lig}}$</th>
<th>2D Tanimoto</th>
<th>3D pharmacophore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirac</td>
<td>$86.2 \pm 1.9$</td>
<td>$84.4 \pm 2.0$</td>
</tr>
<tr>
<td>multitask</td>
<td>$88.8 \pm 1.9$</td>
<td>$85.0 \pm 2.3$</td>
</tr>
<tr>
<td>hierarchy</td>
<td>$93.1 \pm 1.3$</td>
<td>$88.5 \pm 2.0$</td>
</tr>
<tr>
<td>binding pocket</td>
<td>$90.3 \pm 1.9$</td>
<td>$87.1 \pm 2.3$</td>
</tr>
</tbody>
</table>

Orphan GPCRs setup

<table>
<thead>
<tr>
<th>$K_{\text{tar}} \backslash K_{\text{lig}}$</th>
<th>2D Tanimoto</th>
<th>3D pharmacophore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirac</td>
<td>$50.0 \pm 0.0$</td>
<td>$50.0 \pm 0.0$</td>
</tr>
<tr>
<td>multitask</td>
<td>$56.8 \pm 2.5$</td>
<td>$58.2 \pm 2.2$</td>
</tr>
<tr>
<td>hierarchy</td>
<td>$77.4 \pm 2.4$</td>
<td>$76.2 \pm 2.2$</td>
</tr>
<tr>
<td>binding pocket</td>
<td>$78.1 \pm 2.3$</td>
<td>$76.6 \pm 2.2$</td>
</tr>
</tbody>
</table>

(Jacob et al., *BMC Bioinformatics*, 2008)
Influence of the number of known ligands

Number of ligands / GPCR

Performance improvement (hierarchy vs Dirac)

(Jacob et al., BMC Bioinformatics, 2008)
Screening of enzymes, GPCRs, ion channels

**Data:** KEGG BRITE database, redundancy removed

**Enzymes**
- 675 targets
- 524 molecules
- 1218 interactions
- 1218 negatives

**GPCRs**
- 100 targets
- 219 molecules
- 399 interactions
- 399 negatives

**Ion channels**
- 114 targets
- 462 molecules
- 1165 interactions
- 1165 negatives

(Jacob and V., Bioinformatics, 2008)
## Results (mean AUC)

### 10-fold CV

<table>
<thead>
<tr>
<th>$K_{tar}$ \ Target</th>
<th>Enzymes</th>
<th>GPCR</th>
<th>Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirac</td>
<td>0.646 ± 0.009</td>
<td>0.750 ± 0.023</td>
<td>0.770 ± 0.020</td>
</tr>
<tr>
<td>Multitask</td>
<td>0.931 ± 0.006</td>
<td>0.749 ± 0.022</td>
<td>0.873 ± 0.015</td>
</tr>
<tr>
<td>Hierarchy</td>
<td><strong>0.955 ± 0.005</strong></td>
<td><strong>0.926 ± 0.015</strong></td>
<td><strong>0.925 ± 0.012</strong></td>
</tr>
<tr>
<td>Mismatch</td>
<td>0.725 ± 0.009</td>
<td>0.805 ± 0.023</td>
<td>0.875 ± 0.015</td>
</tr>
<tr>
<td>Local alignment</td>
<td>0.676 ± 0.009</td>
<td>0.824 ± 0.021</td>
<td>0.901 ± 0.013</td>
</tr>
</tbody>
</table>

### Orphan setting

<table>
<thead>
<tr>
<th>$K_{tar}$ \ Target</th>
<th>Enzymes</th>
<th>GPCR</th>
<th>Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirac</td>
<td>0.500 ± 0.000</td>
<td>0.500 ± 0.000</td>
<td>0.500 ± 0.000</td>
</tr>
<tr>
<td>Multitask</td>
<td>0.902 ± 0.008</td>
<td>0.576 ± 0.026</td>
<td>0.704 ± 0.026</td>
</tr>
<tr>
<td>Hierarchy</td>
<td><strong>0.938 ± 0.006</strong></td>
<td><strong>0.875 ± 0.020</strong></td>
<td><strong>0.853 ± 0.019</strong></td>
</tr>
<tr>
<td>Mismatch</td>
<td>0.602 ± 0.008</td>
<td>0.703 ± 0.027</td>
<td>0.729 ± 0.024</td>
</tr>
<tr>
<td>Local alignment</td>
<td>0.535 ± 0.005</td>
<td>0.751 ± 0.025</td>
<td>0.772 ± 0.023</td>
</tr>
</tbody>
</table>

(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