Machine learning for cancer informatics and drug discovery

Jean-Philippe.Vert@mines.org

Mines ParisTech / Institut Curie / INSERM U900

ENS Paris, October 13, 2010
Team’s goal

Develop new mathematical/computational models and tools, in particular machine learning, to contribute to:

1. Diagnosis, prognosis and predictive models
2. Identification of important pathways and new drug targets
3. Identification of new drugs
Example: DNA microarray to monitor the transcriptome

Principle of cDNA microarray assay for gene expression (after Gibson & Muse 2002)

Red = "up-regulation"
Green = "down-regulation"
Black = constitutive expression
Question:
Diagnosis / Prognosis from genome / transcriptome

- What class of tumour is that?
- What is the risk of metastasis in 5 years?
- Will drug XXX have an effect?
Question: Identification of new regulations / therapeutic targets

- Which genes are regulated by XXX?
- Which proteins interact with YYY?
Questions:
Virtual screening and QSAR

-Can this small molecule inhibit XXX?
-Is it toxic? Is it drug-like?
Today’s lectures

1. How to make an « intelligent » tool to predict biological properties? We will study the machine learning approach.

2. Then, we will survey several particular applications in bioinformatics and drug discovery
Part 1:
The machine learning approach
Apple or pear?
We need a classifier or predictor.
How to make a predictor?

1. Experiment-based
2. Knowledge-based, « intelligent design »
3. Data-based, « machine learning »
Experiment-based classification

• Design experiments to answer the question
• Carry out the experiment
• The best approach if possible
• However: not always possible, takes time and money
Knowledge-based predictor

• Based on shape, texture, color, …
• Usually difficult to engineer
• Can not be used for other problems (e.g., discriminate strawberries vs grapes)
Data-based predictor = learning
Another way to look at it
Some properties of data-based predictors

• Needs a database of labeled examples
• Does not always provide a simple rule (« black-box »)
• The more data the better!
• The algorithm can be quite generic
Ok, but there is no apple is bioinformatics!

• Sure, but:
  – There are many data
  – Many problems can be formulated as that of « learning a predictor from data »
  – It is often difficult to design knowledge-based predictors (no clear biological theory, noise, large number of features…)
Example: diagnosis/prognosis from microarray data

- Cancer type
- Future evolution
Example: virtual screening

- Activity
- Toxicity
Example: gene annotation

- Localization
- Function
- Structure

Secreted proteins:
- MASKATLLLAFDTLLFATCIARHQQQOQQQQCQLQONIEA...
- MARSSLFTFLCLAVFINGCLSQIEQQSPWFGQGSEVW...
- MALHTVLMSSLPMLEAQNPEHANITIGEPITNETLGWL...

Non-secreted proteins:
- MAPPSVFAEPQAQPVLVFKLIADFREDPDRPKVNLGVG...
- MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDLILVG...
- MSISESYAKEIKTAFQFTDFPIEFGQEDFLPIIGNP...

MAHSKMQN...
Other examples

• Predict function from structure
• Predict splicing sites
• Predict binding sites
• Predict regulated genes
• …
Summary

- Patterns X (image/sequence/structure/…)
- Label Y (binary here, but can be more general)
- We want to build a predictor Y=f(X)
- For this we need a training set of (X,Y) pairs
- We need an algorithm that estimates the predictor f from the training set
- We can then use the predictor to make predictions on new patterns X by f(Y)
My first machine learning algorithm: nearest neighbour

- Define a similarity measure $s(X,X')$ between patterns
- For a new pattern $X$, predict as label $f(X)$ the label of the most similar pattern in the training set
Nearest neighbours

• Very simple to implement
• Good baseline method
• Simple extension: make a majority vote of the k nearest neighbors (k-NN)
Other popular algorithms

- Decision trees, random forests
- Fisher linear discriminant
- Artificial neural networks (ANN)
- Logistic regression
- Boosting
- Support vector machines (SVM)
Linear classifier (simple case)
Which one is better?
Vapnik’s answer: margin
Vapnik’s answer: margin
Vapnik’s answer: margin
The best: largest margin
Support vectors
Implementation

\[
\max_f \{\text{margin}(f)\} \iff \min_f \left\{ \frac{1}{\text{margin}(f)} \right\}
\]

• The problème 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
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.
Some limitations

- What if the data are not vectors?
- What if instead I have a way to measure a distance between patterns (e.g., alignment of sequences, structures, …)
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:

\[ f(X) = \sum_{i \in SV} w_i \|X_i - X\|^2 \]
An interesting generalization

- 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:
  \[
  f(X) = \sum_{i \in SV} w_i d(X_i, X)^2
  \]
Technical details

• This will work very well if the distance $d(X,X')$ satisfies some mathematical conditions («conditionally positive definiteness »)

• If not there still exist tricks to make it work
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
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: summary

• You need a training set of labeled patterns, i.e., of (X,Y) pairs
• You need a distance d(X,X’) between patterns
• You need to choose the parameter C (e.g., cross-validation)
• You plug this into any SVM implementation to train a predictor
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
- Need pairwise distance / similarity as input instead of vectors / fingerprints
Part 2
Applications of machine learning in bioinformatics and drug discovery