Predicting anti-cancer molecule activity using machine learning algorithms

José Santos, Ata Amini, Michael Sternberg, Stephen Muggleton
Imperial College

27th March 2008

Problem

Will the addition of a fixed quantity of a compound into a tumour cell line stop it from growing ?

Motivation

- Growing number of molecules available make it inviable to do in vitro experimentation (it is expensive and time consuming to do these in vitro experiments. E.g. NCI can only evaluate up to 3.000 compounds a year)

- Important real problem for benchmarking machine learning classification algorithms

National Cancer Institute Dataset

- ~4000 unique compounds (i.e. small molecules up to 100 atoms)
- 60 tumour cell lines (i.e. groups of cells from a given tissue type. e.g. Prostate, Breast, Colon)
- For each pair <compound, cell line> we have a boolean variable (true if compound concentration causes at least 50% cell growth inhibition, false otherwise)

Compound information

- Fragment counts

  A compound is converted to a set of fragments. Considering the molecule as a graph, the set of fragments of a molecule are its components of size 1

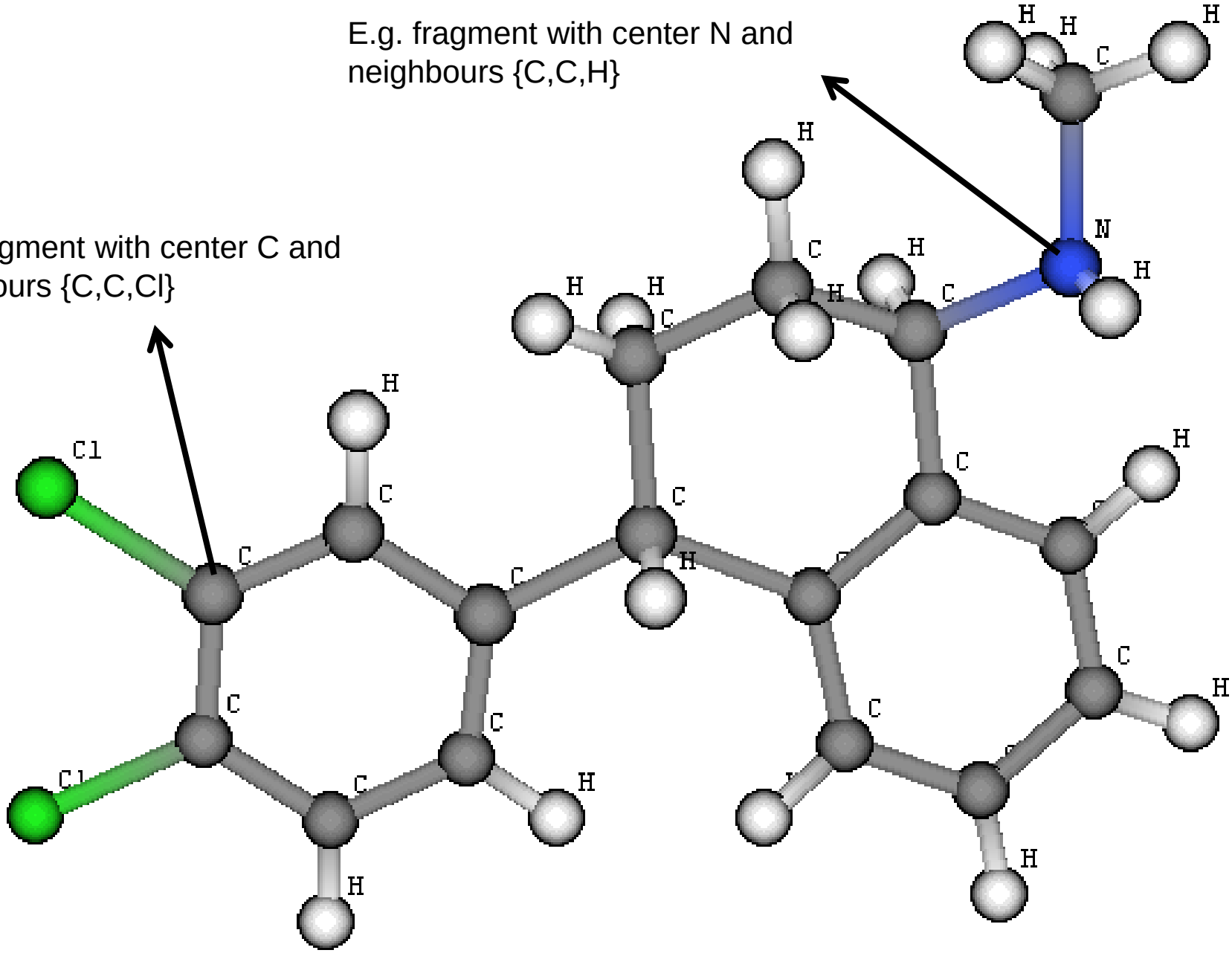
- Molecular weight and Octanol water coefficients (logp)

- Binary target

  Is the compound active for a given cell line?

E.g. fragment with center N and neighbours {C,C,H}

E.g. fragment with center C and neighbours {C,C,Cl}



Data example (in Prolog)

```
cell_line(cColon-205).
cell_line(cBreast-54).
…
fragment(mNSC_9993, fC_C_C_Cl, 2).
fragment(mNSC_9993, fN_C_C_H, 1).
…
logp(mNSC_9993,4.02).
mweight(mNSC_9993,256).

active(mNSC_9993, cColon-205).
:-active(mNSC_9993, cBreast-54).
…
```

Machine learning algorithms

- Decision Trees (C5.0)
- Inductive Logic Programming (Progol)
- Support Vector Machines (LIB SVM)

Decision trees: C5.0

- Tree built in a greedy way choosing at each time the feature that maximizes the information gain
- Very fast (in this problem ~1 min per cell line)
- Generated model is a small set of propositional rules
- Good accuracy but slightly worse (-4%) than Support Vector Machines

Sample rule:

if(mweight > 425 and fC_C <= 2) then active = true (for cell line cColon-205)

Inductive Logic Programming: Progol

- A* top-down search, with an information compression heuristic , in an hypothesis space defined by mode declarations
- Very slow (in this problem ~5 hours per cell line, and even so not exaustive)
- Generated model is a set of first order logic rules.
- Accuracy identical to C5.0 (the issue with this problem is that fragment counts have very little relational potential)

Sample rule:

active(A, cColon-205*) :- fragment(A, fN_C, B), fragment(A, fC_C_H, C), C<B.*

Support Vector Machines: LIB SVM

- Observations are separated by an hyperplane maximally separating the two classes in a high dimension space
- Fast (in this problem ~5 minutes per cell line)
- Generated model are the support vectors (meaningless for human understanding)
- High accuracy (~73% in this problem, the default being 54%)
- Can use a generic kernel (e.g. RBF) or a user defined kernel function (i.e. a function that computes the similarity between two objects)

Comparison with other work

We compare our results with similar work from Baldi et al 2005.

- Their method was SVMs with a specific kernel (tanimoto coefficient) over the same dataset

- Our SVM used a generic RBF kernel with fragment counts, molecular weight and logp as features

Some classification results per cell line

Cell Line	Default	Baldi et al 05	C5.0	Lib SVM
786-0	52.25	72.62	67.94	73.42
A498	51.21	71.81	67.70	73.25
A549	50.91	72.38	68.26	73.41
ACHN	50.84	73.47	66.78	74.03
BT-549	50.36	71.99	69.11	73.97
CAKI-1	52.09	72.34	67.15	71.96
CCRF-CEM	63.71	72.36	70.40	75.09
COLO-205	53.31	72.67	69.41	71.71

Overall classification results

Algorithm	Default	Baldi et al	C5.0	LIB SVM
Avg. Accuracy	54.25%	72.29%	68.75%	72.98%
Std Deviation	3.48%	0.98%	1.46%	1.18%

P-value for a paired (over the 60 cell lines) t-test, with the null hypothesis being that the Baldi et al and LIB SVM classifiers have the same accuracy, is 1E-8.

Generalizing fragment counts

So far we have considered as fragments graph components of size 1 (as has been done in related literature).

What happens if we use other component sizes?

Surprisingly if instead of using fragment counts we simply use atom counts the accuracies are almost as good for SVMs and identical for DTs

		LIB SVM		C5.0	
Comp. size	# Features	Avg. Acc.	Std. Dev.	Avg. Acc.	Std. Dev.
0	~60	72.14	1.32	68.86	1.41
1	~1000	72.65	1.07	68.78	1.47
2	>10.000	71.54	1.10	67.81	1.45

Average accuracy per algorithm per component size

	Component size		
Algorithm	0 with 1	0 with 2	1 with 2
LIB SVM	6.0E-06	4.2E-05	1.5E-18
C5.0	5.1E-01	5.4E-12	5.0E-12

P-values

Conclusions

- Marginally better accuracy than previous work

- Fragment counts are not as important as previously thought

- Although SVMs generate the most accurate models, such models are close to useless for important practical tasks like drug design.

- For such tasks descritive, rather than just predictive, models like ILP or decision trees are required

Possible improvement directions

Apply ILP using proper relational background knowledge (e.g. distance between atoms or fragments)

For this to be practical, as the hypothesis space is even bigger, ILP systems much faster than existing ones are needed