Connectionist Computing
COMP 30230/41390

Gianluca Pollastri

office: E0.95, Science East.

email: gianluca.pollastri@ucd.ie

Credits

- **Geoffrey Hinton, University of Toronto.**
  - **borrowed some of his slides for "Neural Networks" and "Computation in Neural Networks" courses.**

- **Ronan Reilly, NUI Maynooth.**
  - **slides from his CS4018.**

- **Paolo Frasconi, University of Florence.**
  - **slides from tutorial on Machine Learning for structured domains.**

Lecture notes on Brightspace

- **Strictly confidential...**
- **Slim PDF version will be uploaded later, typically the same day as the lecture.**
- **If there is demand, I can upload onto Brightspace last year's narrated slides.. (should be very similar to this year's material)**

Books

- No book covers large fractions of this course.

- Parts of chapters 4, 6, (7), 13 of Tom Mitchell's "Machine Learning"
- Parts of chapter V of Mackay's "Information Theory, Inference, and Learning Algorithms", available online at:

http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html

- Chapter 20 of Russell and Norvig's "Artificial Intelligence: A Modern Approach", also available at:

http://aima.cs.berkeley.edu/newchap20.pdf

- More materials later..

Marking

- **3 landmark papers to read, and submit a 10-line summary on Brightspace about: each worth <u>6-7%</u>**
- **a connectionist model to build and play with on some sets, write a report: <u>30%</u>**
- **Final Exam in the RDS (<u>50%</u>)**

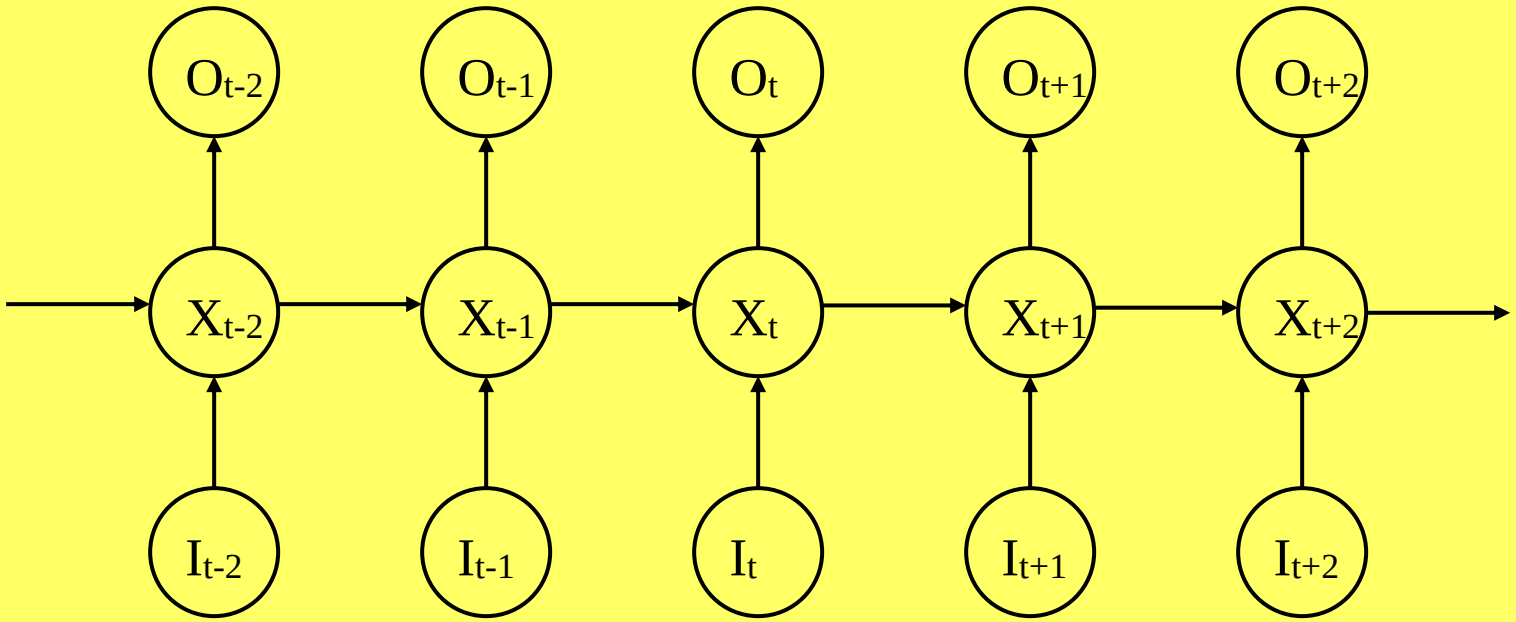
Programming assignment

- **Implement a Multi-Layer Perceptron, test it.**

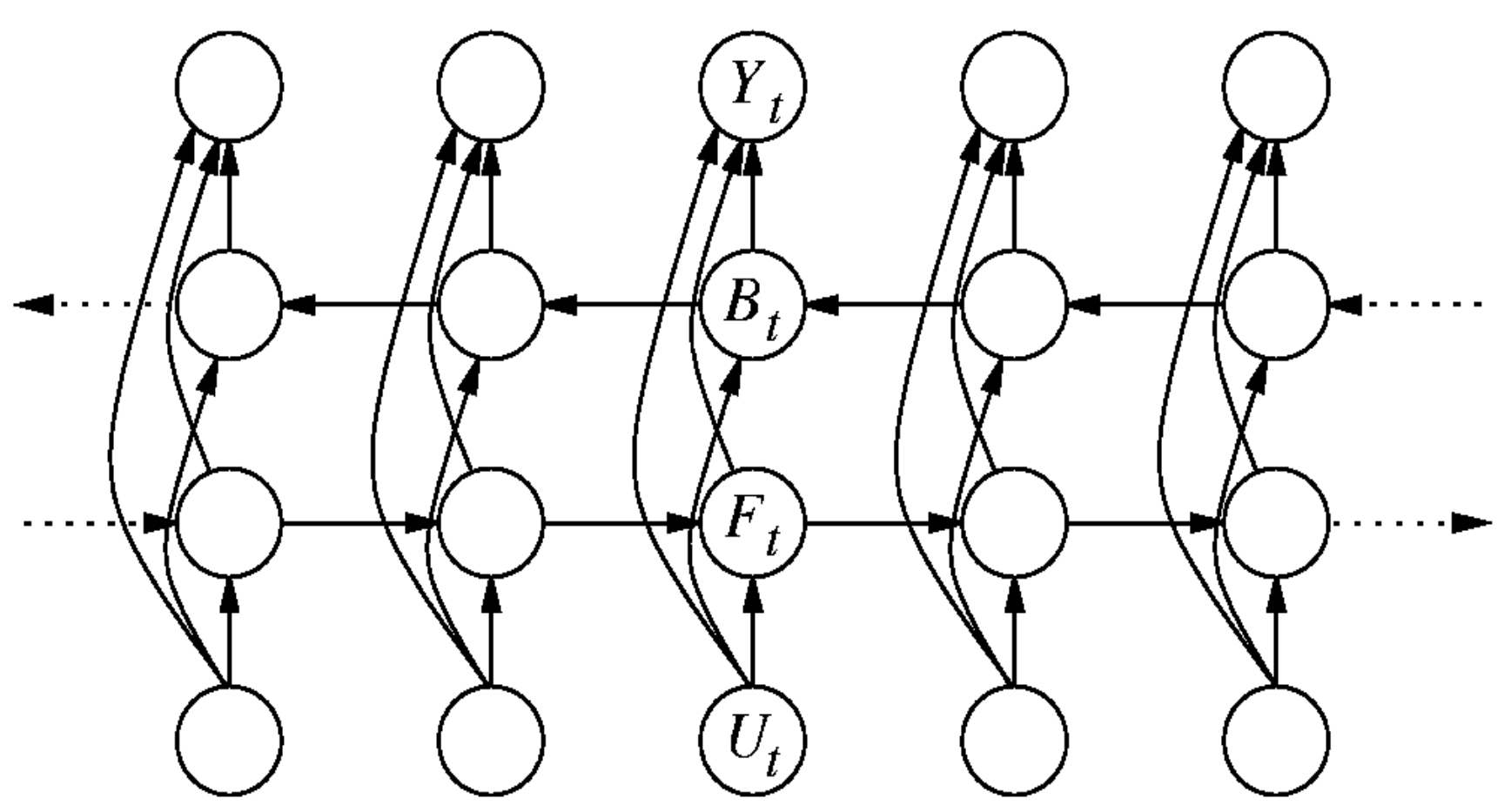
- **The description on Brightspace.**

- **Submit through Brightspace code and test results by Dec the 5th at 23:59, any time zone of your choice (Baker Island?).**
- **30% of the overall mark**
- **One third of a grade down every day late, that is: if you deserve an A and you're 1 day late you get an A-, 2 days late a B+, etc.**

Recurrent Neural Networks

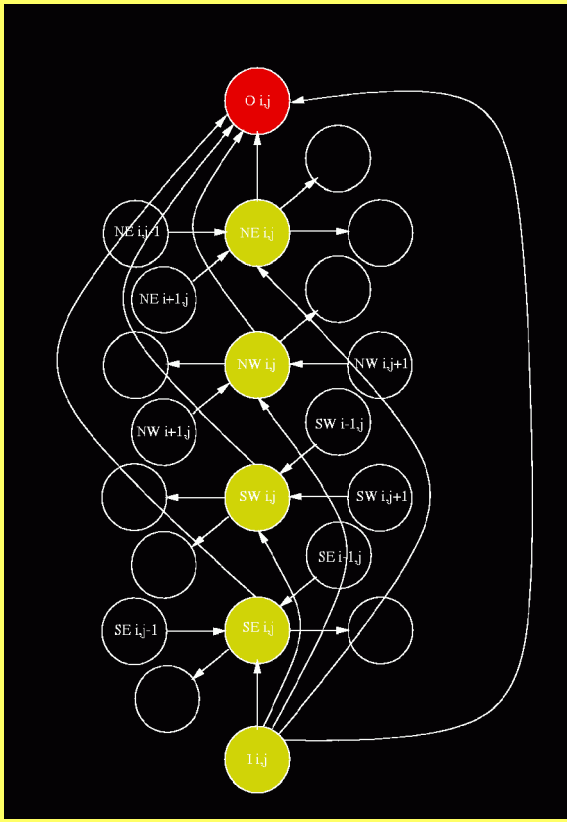


Bidirectional Recurrent Neural Networks (BRNN)



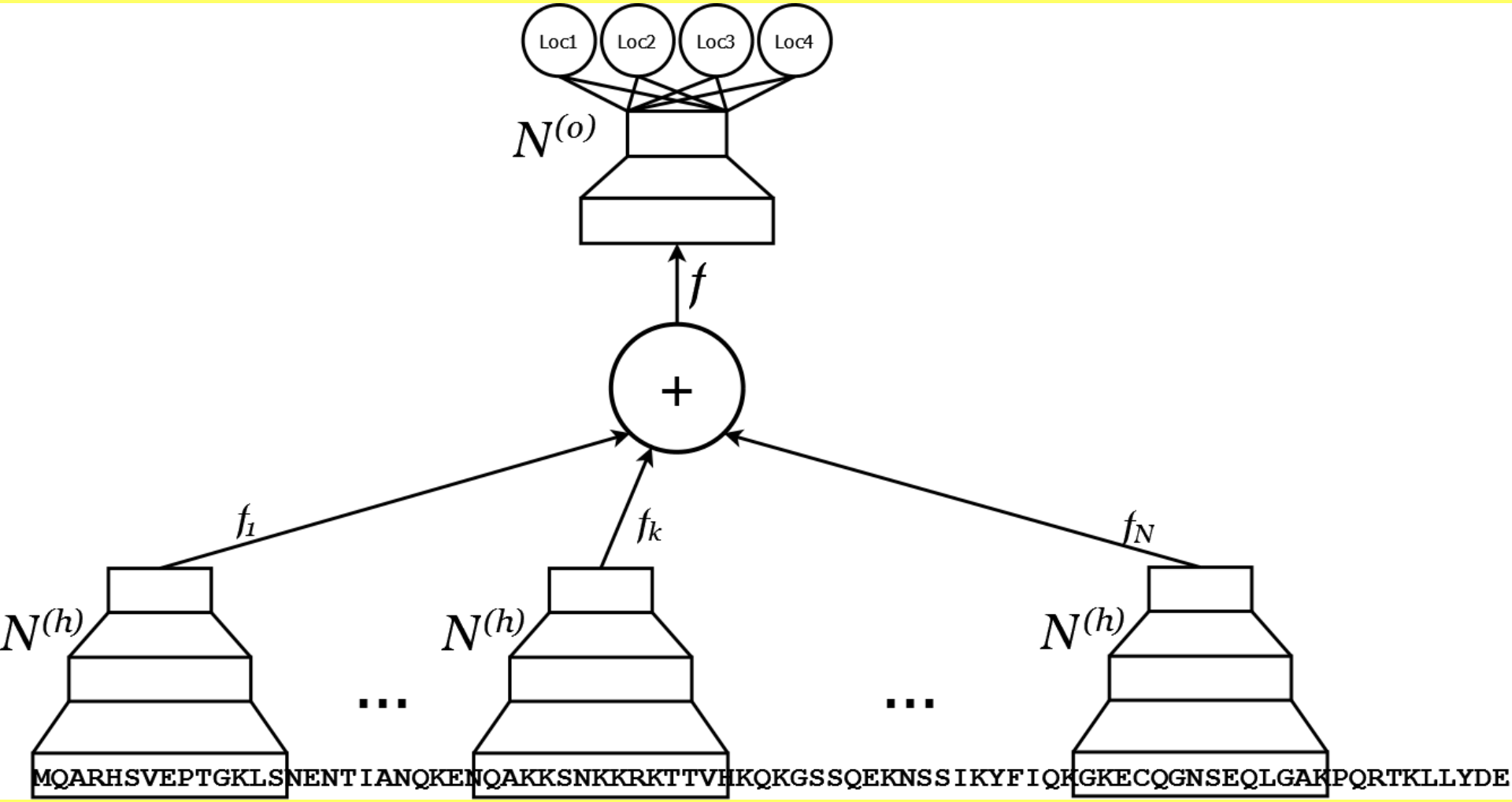
2D RNNs

$$
\begin{cases}
O_{ij} = \mathcal{N}_O(I_{ij}, H^{NW}_{i,j}, H^{NE}_{i,j}, H^{SW}_{i,j}, H^{SE}_{i,j,}) \\
H^{NE}_{i,j} = \mathcal{N}_{NE}(I_{i,j}, H^{NE}_{i-1,j}, H^{NE}_{i,j-1}) \\
H^{NW}_{i,j} = \mathcal{N}_{NW}(I_{i,j}, H^{NW}_{i+1,j}, H^{NW}_{i,j-1}) \\
H^{SW}_{i,j} = \mathcal{N}_{SW}(I_{i,j}, H^{SW}_{i+1,j}, H^{SW}_{i,j+1}) \\
H^{SE}_{i,j} = \mathcal{N}_{SE}(I_{i,j}, H^{SE}_{i-1,j}, H^{SE}_{i,j+1})
\end{cases}
$$



Pollastri & Baldi 2002, *Bioinformatics*
Baldi & Pollastri 2003, *JMLR*

N to 1 Neural Networks



Mooney, Wang & Pollastri, *Bioinformatics*, 2011

Full, general graphs

- **Can we extend this process to any shape and map?**

- **Yes we can, but let me tell you how through a story.**

Small molecules

- **Small organic compounds that are bioactive, but are not proteins, DNA, RNA, etc.**

QSAR

- **Quantitative Structure-Activity Relationship**
- **Based on the observation that what a molecule does depends on its (chemical? 3D?) structure**
- **Computationally, predict Activity or Properties of small molecules, e.g. toxicity, carcinogenicity, biodegradability, anti-cancer activity, etc.**

Why QSAR?

- **Screening computationally is *a lot* cheaper than screening experimentally.**
- **You have 10,000 compounds that may (or may not) do a job. If you can prune 9,950 of them by computational means you have saved *a lot* of time and $$$.**
- **(Designing 1 drug costs ~2bln and takes 13.5 years on average..)**

Traditional QSAR framework

$$Activity = F(structure)$$

$$F() = g(t())$$

- **t() encodes a structure (graph, or 3D, or both) into a fixed-width array of descriptors of a molecule.**
- **g() maps this array into the property of interest.**
- **Finding t() is the real challenge. Once there is a good t(), the problem is solved.**

Finding t()

- **For many domains you can <u>ask experts</u> to tell you what makes a good carcinogen/ toxic molecule/ biodegradable molecule etc.**

Finding t(): asking experts

- But that's work..
- And then you have to endure meeting them over a Scotch at 11am.
- Besides, the expert will be inherently fallible, and dated.

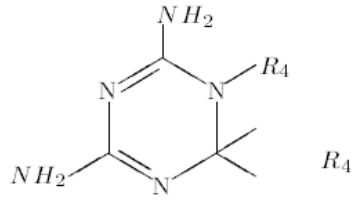
Finding t() (2)

- **You can toss every t() you have in your arsenal at it and pick the best (trial and error).**
- **But that's cheating, really, unless you have 1G molecules to prevent you from fooling yourself into thinking your tail is your average.**
- **And we ain't got 1G molecules (though I've been told we apparently do).**

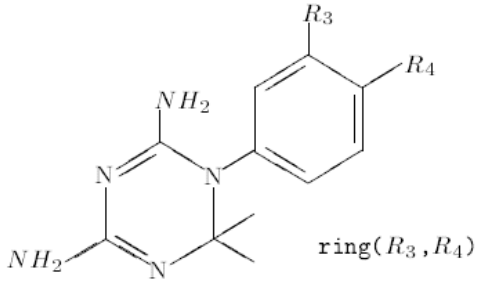
Methods for QSAR (on ~1 slide)

- **Linear regression (on what features?)**
- **ANN (same?)**
- **Recursive NN (but need a DAG!)**

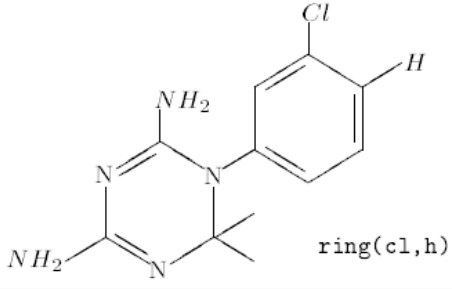
(a) Basic triazine template



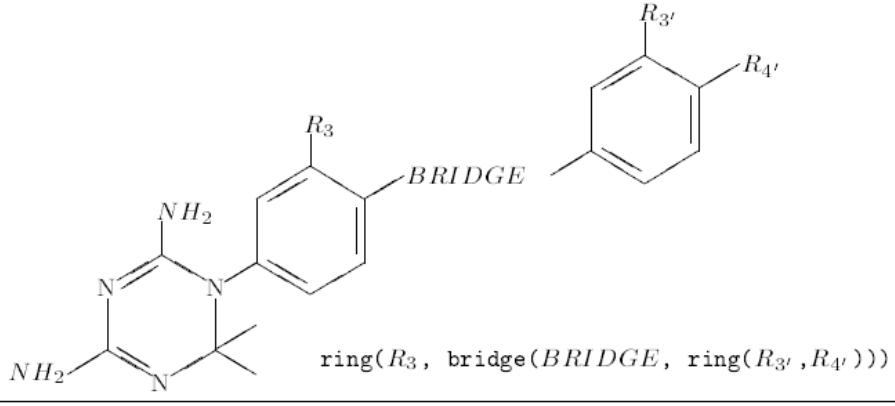
(b) Triazine template with one phenyl ring



(c) Example drug 159: $3-Cl, 4-H$



(d) Triazine template with two phenyl rings and bridge



Schmitt & Goller 1998

A lazy computer scientist's viewpoint

- **Molecules are Undirected Graphs**
- **In principle, the property of interest may be determined by any part of the graph**
- **Context matters, a lot**
- **No direction is, in principle, more important**
- **We are too lazy to learn about molecules**

Besides, variable

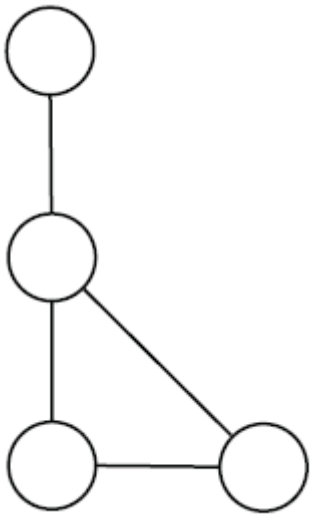
- **Molecular graphs are of *variable* size.**
- **We have to come up with models that deal with that, or still endure those 11am Scotches.**

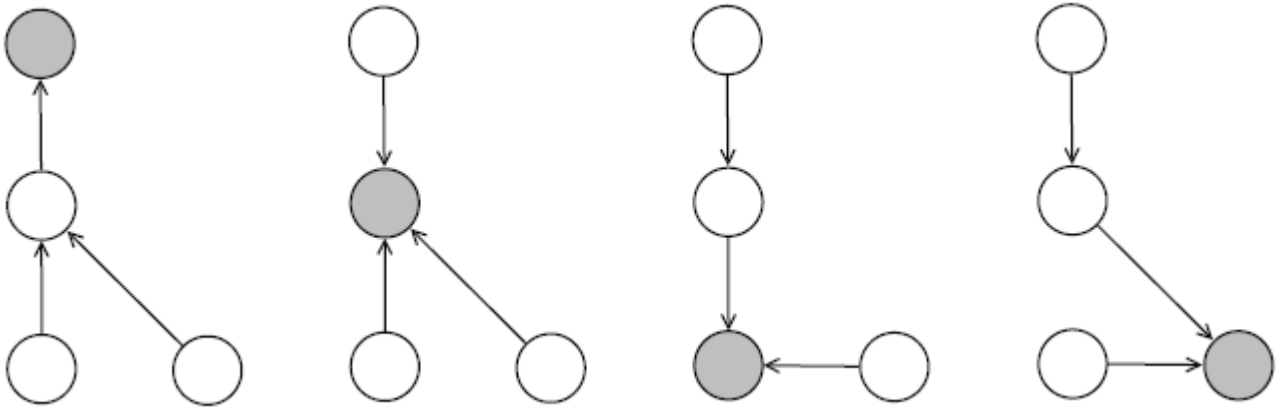
A lazy computer scientist's viewpoint (2)

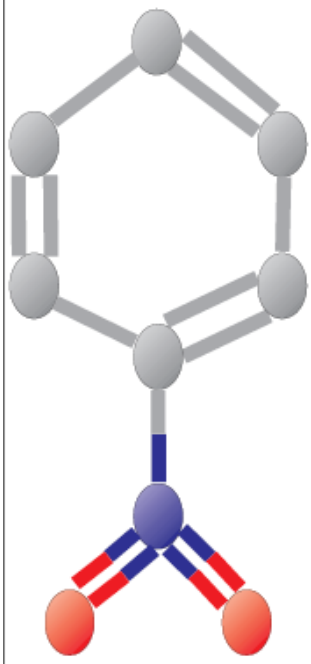
- Molecules are Undirected Graphs
- In principle, the property of interest may be determined by any part of the graph
- Context matters, a lot
- No direction is, in principle, more important
- We are too lazy to learn about molecules

- **Factorise a UG into as many DAGs as it takes so that each node becomes a root**
- **If N nodes, this obviously makes N DAGs (but small molecules are small!)**

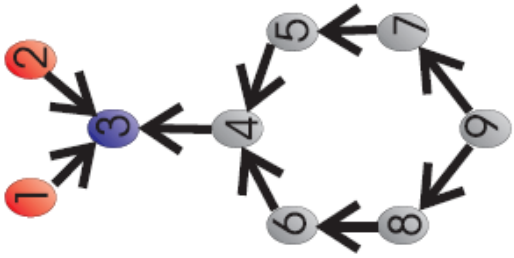
Example



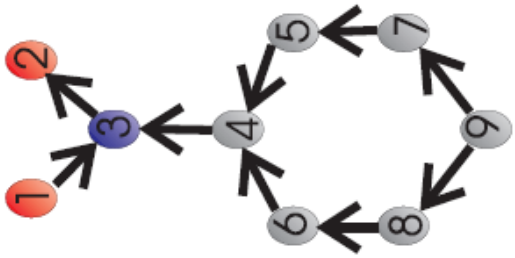




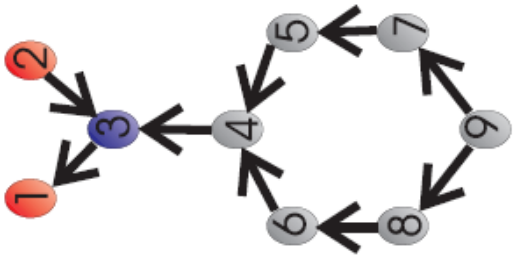
Undirected graph



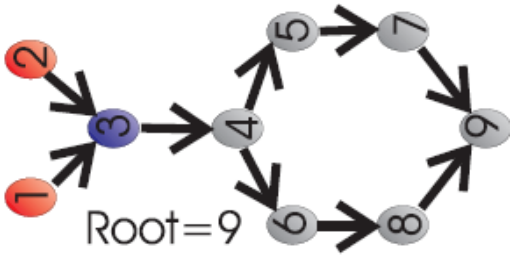
Root=3



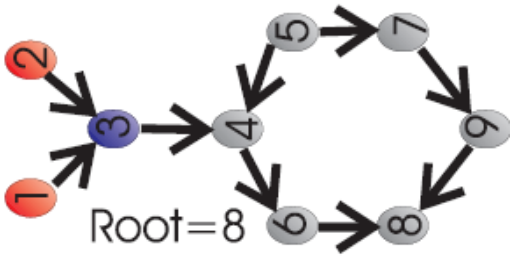
Root=2



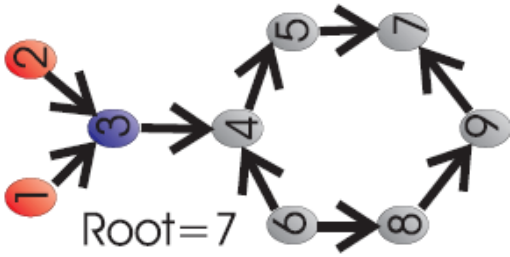
Root=1



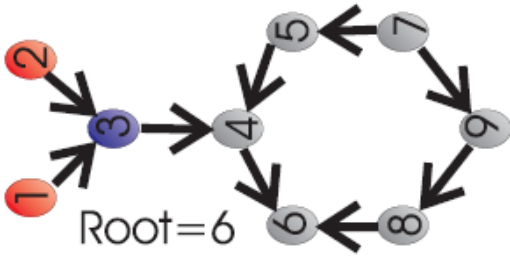
Root=9



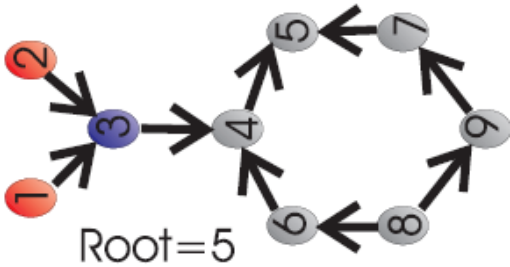
Root=8



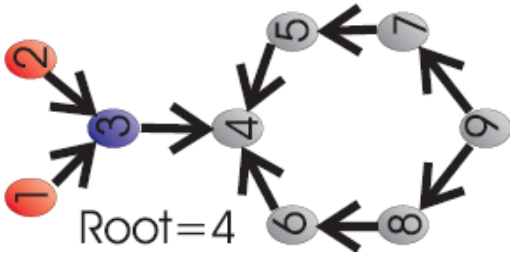
Root=7



Root=6



Root=5



Root=4

What we have done (3)

- **The state of each vertex in the k-th DAG is a function of the atom (and potentially anything else we know) at that vertex, and the states of the children:**

$$\mathbf{G}_{v,k} = \mathcal{M}^{(G)}\left(i_v, \mathbf{G}_{ch^1_{[v,k]}}, \ldots, \mathbf{G}_{ch^n_{[v,k]}}\right)$$

- **When there are no children, the recursion ends.**

What we have done (4)

- **Problem: there are N graphs, hence N roots, hence N vectors G (contexts around each atom). How to map them to 1 property? Add them up! If v_k is the root vertex in the k-th graph:**

$$\mathbf{G}_{structure} = \sum_{k=1}^{N} \mathbf{G}_{v_k,k}$$

- **All graphs compete to be in this global state: it is an adaptive form of compression, property driven!**

What we have done (5)

- **We map the global state into the property of interest (o) as:**

$$o = \mathcal{M}^{(O)}(G_{structure})$$

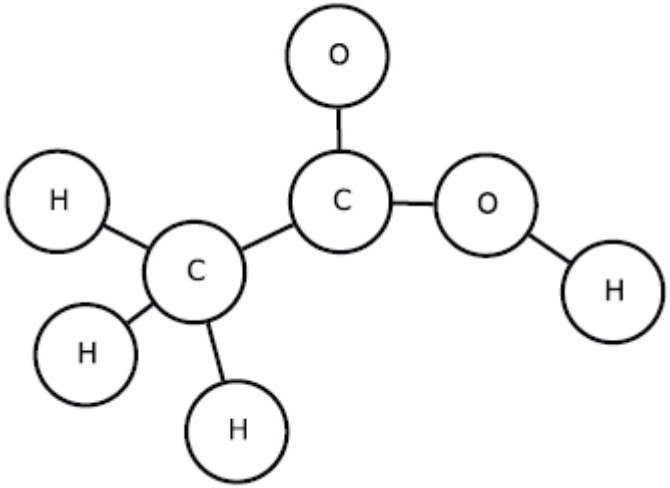
- **The molecule-to-features and features-to-output functions are modelled by Multi-Layer Perceptrons.**
- **We train the whole thing (all DAG's and all) by gradient descent.**

In plain language..

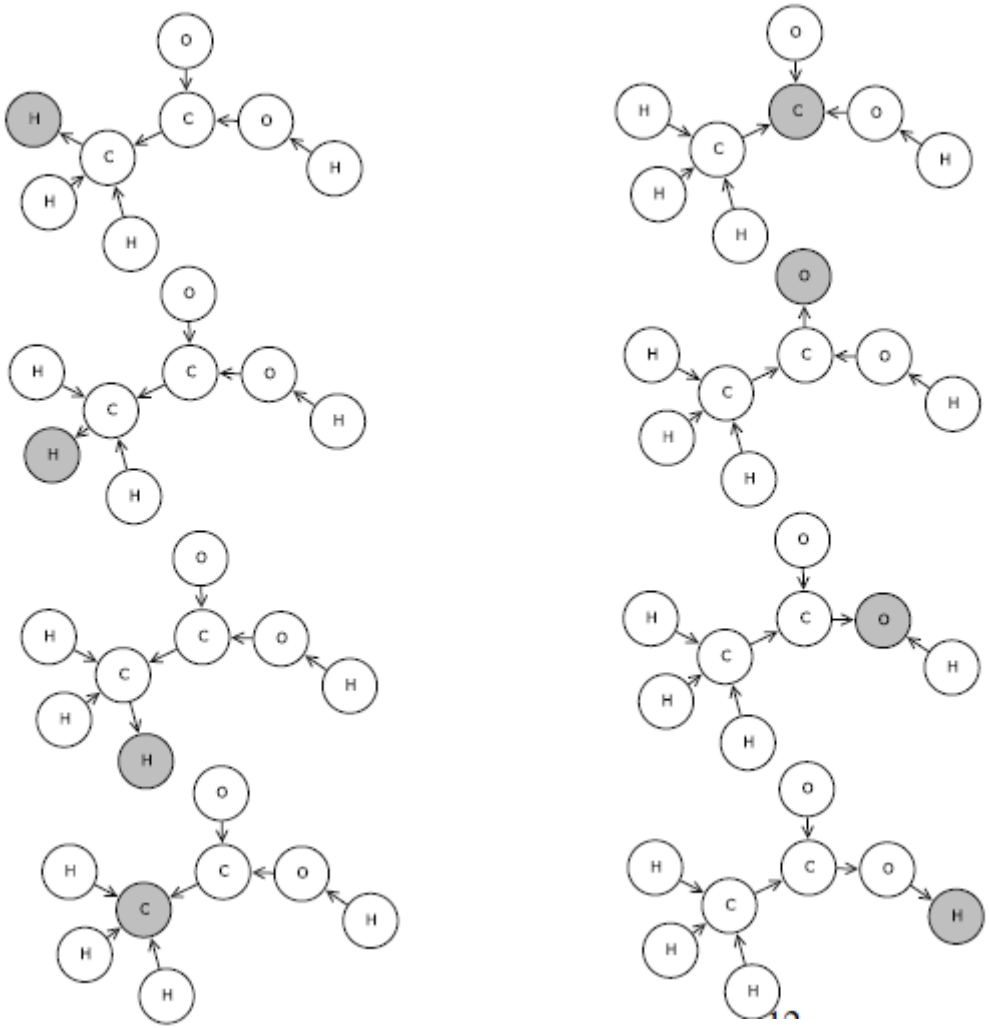
- **We look at the graph from the point of view of each node.**
- **We *add up* all these points of view.**
- **What we get is a fixed-width vector of real numbers that describes the graph.**
- **Based on this vector we predict the property.**
- **Note: what vector we get depends on the property we're trying to predict.**

Overall example

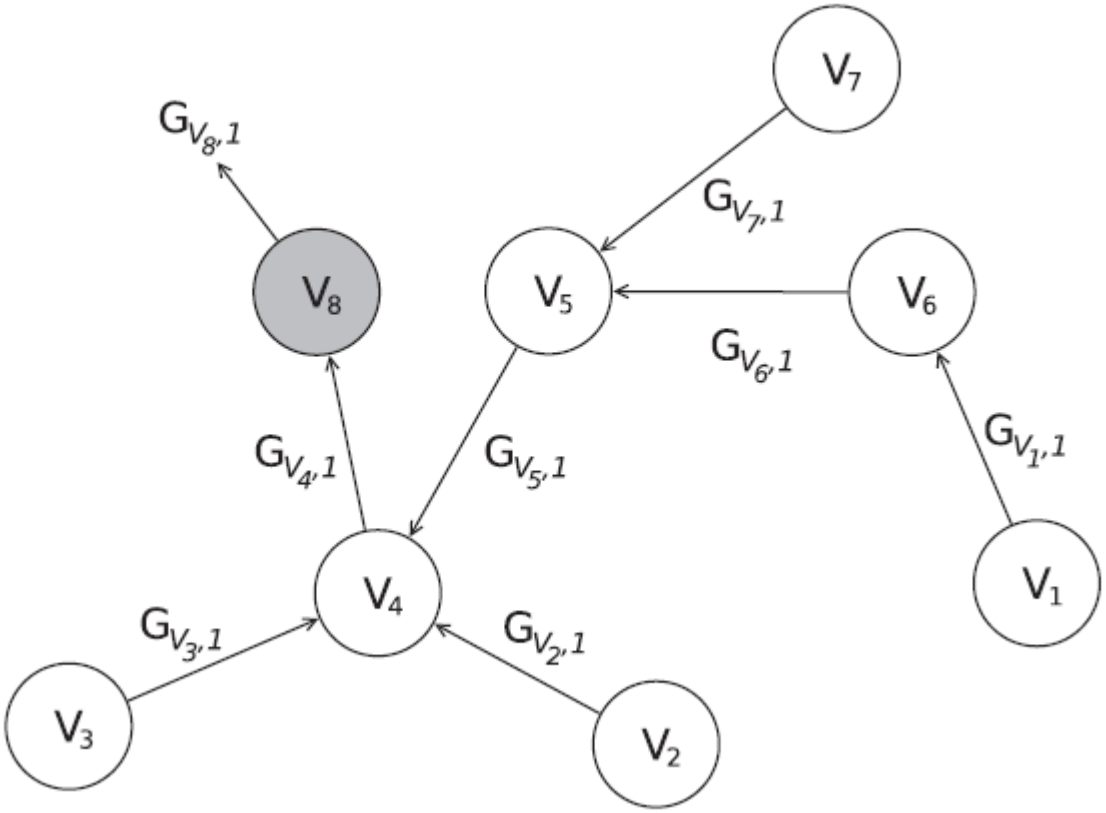
- **UG of acetic acid**

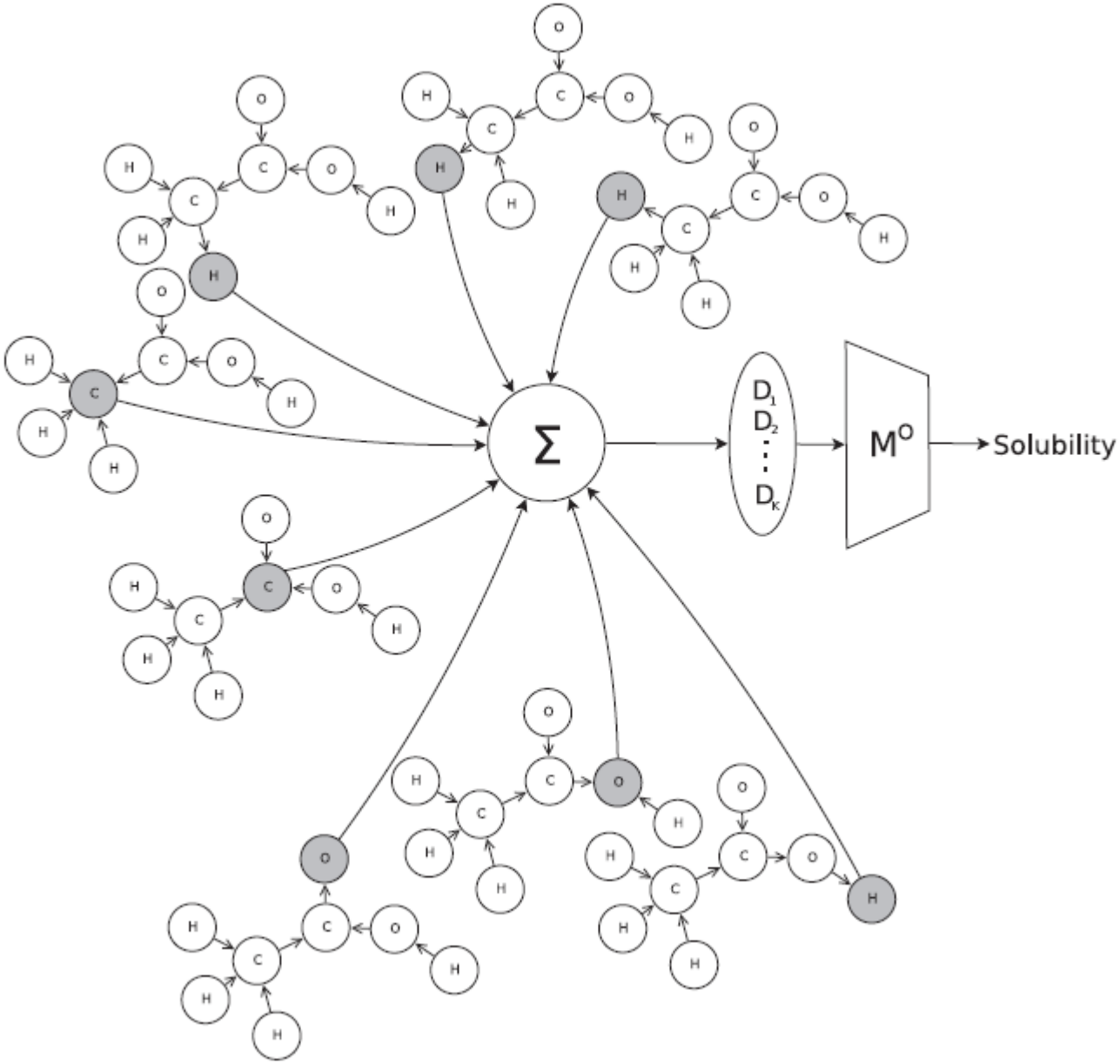


Factorising..

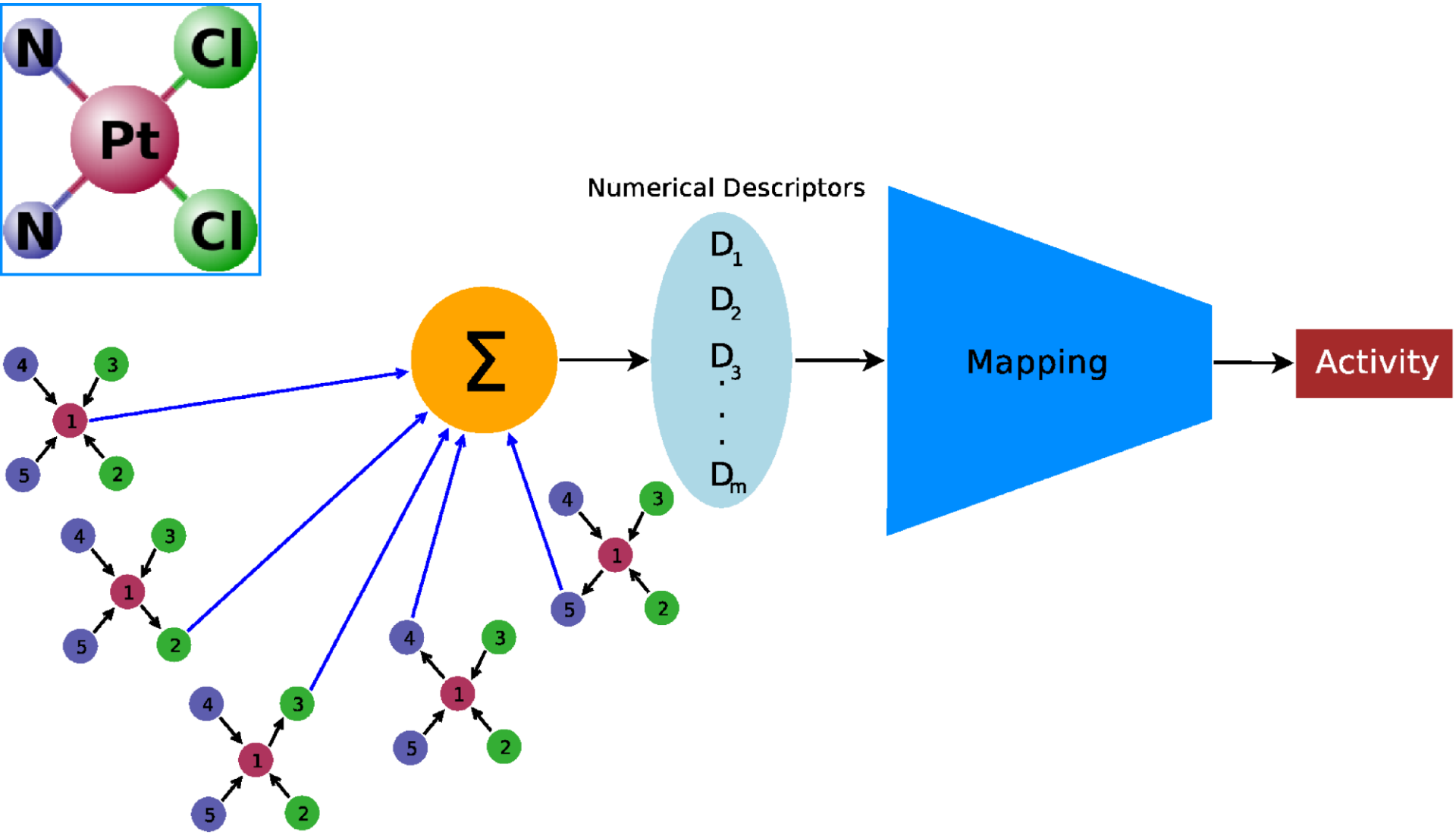


One graph





A.Lusci, G.Pollastri, P.Baldi,
"Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules",
***Journal of Chemical Information and Modeling*, 2013**


N
Cl
Pt
N
Cl
Numerical Descriptors
D_1
D_2
D_3
D_m
Σ
Mapping
Activity

Training

- **Model is a feed-forward neural net: backpropagation.**
- **But it's deep (paths of variable depth) and the gradient vanishes, even when the graphs are small.**
- **Clip the gradient on both sides.. If it's smaller than X, make it X. If it's bigger than Y, make it Y.**
- **Massive speedup in training times.**

UG-RNN

- **Do they work?**
- **Yes they do, better than anything else.**
- **And they *automatically* find the features.**
- **They are DEEP, of course, so they are clever.**

- **But they are hard to train.**

Summary: NN for structured data

- No need to design features. This is good:
  - no extra work, experts, morning imbibing, etc.
  - and, more importantly, less risk of overfitting the test set by trying 1000 different combinations of hyperparameters
- Some of these models are a bit temperamental and need special (deep) learning techniques to work