Horn-Clause Neural Networks

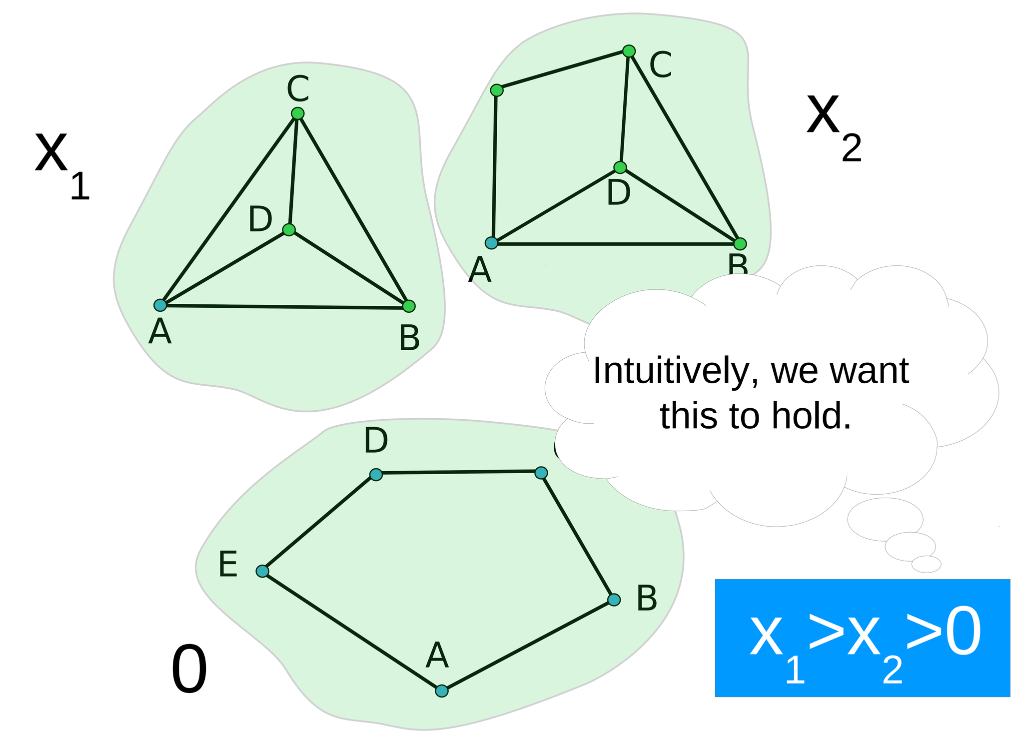
Vojtěch Aschenbrenner[1] & Ondřej Kuželka[2]

[1]CTU in Prague, [2]KU Leuven

A simple FOL-based neural network model – an example program:

1: brightTetrahedron(A,B,C,D) :- brightTriangle(A,B,C), brightTriangle(A,B,D), brightTriangle(B,C,D), brightTriangle(A,C,D).
1: brightTriangle(X,Y,Z) :- bright(X), bright(Y), bright(Z), edge(X,Y), edge(Y,Z), edge(Z,X).
0.1: brightTriangle(X,Y,Z) :- bright(W), bright(X), bright(Y), bright(Z), edge(X,W), edge(W,Y), edge(Y,Z), edge(Z,X). /* A rectangle is almost a triangle too :) */
0.1: bright(blue).
1: bright(green).
2: bright(yellow).
2: bright(white).

Examples of '*brightTetrahedrons(A,B,C,D)*':



Semantics:

1. ***Value*** *of a ground fact* is a parameter (the number on the left)
2. ***Output*** *of a true ground Horn clause* $C = h$:- $b_1,...,b_k$ is given as:

$$\textbf{output}(C) = \textbf{sigmoid}(\textbf{value}(b_1)+...+\textbf{value}(b_k)-k)$$

3. ***Output*** *of a false ground clause is 0.*
4. ***Value*** *of a ground atom A with predicate h* is given as follows:
   - Let $Cs = \{C_1,...,C_m\}$ be the set of all Horn clauses with h in the head.
   - Let $Gr(C_i)$ denote the set of all true groundings of C_i with the ground atom A in the head.
   - Then

$$\textbf{value}(A) = \textbf{sigmoid}(w_{C1}\,\textbf{max}_{\overline{C}1 \text{ in } Gr(C1)}\textbf{output}(\overline{C}_1)+...+w_{Ck}\,\textbf{max}_{\overline{C}k \text{ in } Gr(Ck)}\textbf{output}(\overline{C}_k)+w_0^A)$$

5. ***Value*** *of an atom A* is the maximum of the values of its groundings.

This can be seen as a template for feed-forward neural networks.
The ground network can be constructed as follows from a logic program with weights and a query atom (*in practice, we use an optimized algorithm which utilizes caching, branch-and-bound, forward-checking etc., this is just for illustration*):

```
Procedure constructNetworkForClause(C = h:-b_1, ..., b_k)
best := NULL
'OuterLoop': For each grounding θ of C
        node := a neuron with no inputs and bias equal to -k
        For i = 1, ..., k
            subnetwork = ConstructNetworkForAtom(bθ_i)
            If subnetwork == NULL Or subnetwork.evaluate() == 0
                Continue to 'OuterLoop'
            Else
                Connect subnetwork to node (weight 1)
            EndIf
        EndFor
        If node is better than best
            best := node
        EndIf
EndFor
Return best
```

```
Procedure constructNetworkForGroundAtom(a = p(c1,...,ck)

node := a neuron with no inputs and bias w_0^p
For each clause C = p(X1,...,Xk) :- b1, ..., b_m
    Let θ be minimal such that p(X1,...,Xk)θ = p(c1,...,ck)
    subnetwork = constructNetworkForClause(C)
    If subnetwork.evaluate() != 0
        Connect subnetwork to node with the weight specified for C in the program
    EndIf
EndFor
Return node
```

Parameter learning:

Parameter learning is done by repeating the following steps:

1. Construct neural networks for every program $H+e_i$ where H is a hypothesis e_i is a learning example
2. Check if the stopping criterion is met and if so, finish.
3. Perform online backpropagation for a given number of steps for each of the networks (updating the shared weights – note that the networks for different examples in the dataset can be different but they share some weights).

Some preliminary experiments:

Experiments were performed on chemical data. The structure was selected so that the program would have to induce soft clusterings of atom and bond types relevant for the respective datasets.

w_{toxic1}: toxic :- bond(A1,A2,B1), bond(A2,A3,B2), atg1(A1), atg2(A2),atg3(A3), bg1(B1), bg2(B2).
w_{toxic2}: toxic :- bond(A1,A2,B1), bond(A2,A3,B2), atg1(A1), atg2(A2),atg3(A3), bg1(B1), bg3(B2).
w_{toxic3}: toxic :- bond(A1,A2,B1), bond(A2,A3,B2), atg1(A1), atg2(A2),atg3(A3), bg2(B1), bg3(B2).
...
...
w_{atg11}: atg1(X) :- atm(X,carbon)
w_{atg12}: atg1(X) :- atm(X,hydrogen)
w_{atg13}: atg1(X) :- atm(X,nitrogen)
...
...
w_{atg21}: atg2X) :- atm(X,carbon)
w_{atg22}: atg2(X) :- atm(X,hydrogen)
w_{atg23}: atg2(X) :- atm(X,nitrogen)
...
...
...quite large network!

Already with this simple model, we were able to obtain competitive accuracies to nFOIL for PTC and Mutagenesis.

Future work:

1. Experiments with datasets where the ability to construct useful soft concepts (clusters) is expected to be useful

2. Structure learning

3. Make it deep

References:
V. Aschenbrenner, (supervisor O. Kuzelka): Deep Relational Learning with Predicate Invention, MSc Thesis, CTU in Prague, 2013

Acknowledgement:
Part of this work was done while VA and OK were with CTU in Prague. OK is supported by Jan Ramon's ERC Starting Grant 240186 'MiGraNT.