Neural-Symbolic Integration
A selfcontained introduction

Sebastian Bader    Pascal Hitzler

ICCL, Technische Universität Dresden, Germany
AIFB, Universität Karlsruhe, Germany
Outline of the Course

- Introduction and Motivation
- The Core Method for Propositional Logic
- Applications of the Propositional Core Method
- The Core Method for First-Order Logic
- More on First-Order & other Perspectives
Outline:

Initialising Networks

Introduction
NeSy Tagging System
Results
Conclusions

Guiding Backprop
Neuro-Symbolic Word Tagging

- Joint work with:
  - Nuno Marques (UNL)
  - Vitor Rocio (UNL)
  - Steffen Hölldobler (ICCL)

- “Neuro-Symbolic Word Tagging”
  (Marques, Bader, ea., 2007)
Word Tagging

Word Tagging (Part-of-speech tagging) is the process of assigning grammatical tags (like noun, verb, etc.) to a word depending on its definition and its context.

- Task: Construct a tagging function from an annotated corpus and background knowledge.
- Tagging function: Maps a word (together with its context) to some wordclass or to a distribution over all tags.
Wordclass Tagging: Example

Sentence from an annotated corpus:

"pto I prp am be not adv prepared v to prep grant v bail n to prep any pri of prep them prp "pto ,pto said v the det magistrate n .pto

Part of the dictionary extracted from the corpus:

<table>
<thead>
<tr>
<th>Word</th>
<th>$P_{adj}$</th>
<th>$P_{adv}$</th>
<th>$P_n$</th>
<th>$P_v$</th>
<th>$P_{prep}$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>right</td>
<td>42%</td>
<td>25%</td>
<td>33%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>following</td>
<td>40%</td>
<td>25%</td>
<td></td>
<td></td>
<td>35%</td>
<td>25%</td>
</tr>
<tr>
<td>beat</td>
<td>50%</td>
<td></td>
<td>8%</td>
<td></td>
<td>42%</td>
<td></td>
</tr>
<tr>
<td>outside</td>
<td>17%</td>
<td>33%</td>
<td></td>
<td></td>
<td>50%</td>
<td></td>
</tr>
<tr>
<td>grant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>75%</td>
<td>25%</td>
</tr>
</tbody>
</table>

Susanne corpus (used in experiments): > 150k words
Neural Networks for Wordclass Tagging

- Context of $n$ words
- Represent word by its probability vector $p_i \in \mathbb{R}^t$
- Compute the resulting vector $p \in \mathbb{R}^t$
- Use the tag with maximum value

¿ How to use background knowledge?
Consider the following simple grammar:

\[ s \rightarrow pto, np, vp, pto \]
\[ vp \rightarrow v, np \]
\[ optrel \rightarrow that, vp \]
\[ vp \rightarrow v \]
\[ np \rightarrow pn \]

Unfolding this grammar five times yields 88 sentences:

\[ [pto, det, n, that, v, det, n, v, det, n, that, v, pn, pto], \]
\[ [pto, det, n, that, v, det, n, v, det, n, that, v, pto], \]
\[ [pto, det, n, that, v, pn, v, det, n, that, v, pn, pto], \]

Most frequent triples and their propositional rules:

\[ [pto, det, n] \quad \leadsto \quad det \leftarrow pto_{-1}, n_{+1} \]
\[ [det, n, that] \quad \leadsto \quad n \leftarrow det_{-1}, that_{+1} \]
\[ [n, that, v] \quad \leadsto \quad that \leftarrow n_{-1}, v_{+1} \]
Wordclass Tagging: Background Knowledge 2

- **Exception rule:** “A noun? preceded by an auxiliary verb? is actually a verb!”
- Encoding as propositional rule:
  \[ v \leftarrow aux_{-1}, n_0 \]
- Known as e.g. ”transformation rules”
- Generated by linguists or extracted from corpus
Following the propositional core method we embedded rules into the network:

\[ v \leftarrow aux_{-1}, n_0 \]
Wordclass Tagging: Results

- To avoid overfitting, training data is split into parts
  - training data
  - test / validation data
Wordclass Tagging: Results

- To avoid overfitting, training data is split into parts
  - training data
  - test / validation data

- Error on training and test corpus:
Wordclass Tagging: Results

- To avoid overfitting, training data is split into parts
  - training data
  - test / validation data

- Error on training and test corpus:
Wordclass Tagging: Results

Evolution of errors:

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
Wordclass Tagging: Results

- Evolution of connection weights:
  - NeSy
  - Randomised

- "Structure" is more or less preserved, only small changes.
Wordclass Tagging: Results

- Evolution of connection weights:

  NeSy

  Randomised

  ”Structure” is more or less preserved, only small changes.
Wordclass Tagging: Conclusions

- Encoding of (available) background knowledge into connectionist systems
- NeSy System outperforms naive network on validation set, especially if only limited data is available (which is usually the case)
- The Core Methods provides a good framework
- The tagging problem is magnitudes bigger than any other NeSy problem so far: 3190 vs. 156610 examples
Outline:

Initialising Networks

Guiding Backprop

The "Tic-Tac-Toe" Classification Task
Rule Extraction
Embedding Rules into the Network
Rule Encoding Example
Experimental Evaluation
Discussion
Joint work with:
- Nuno Marques (UNL)
- Steffen Hölldobler (ICCL)

“Guiding Backprop by Inserting Rules”
(Marques, Bader, ea., 2008)
Introduction and Motivation

😊 Rule insertion prior to training can lead to faster convergence and to better results (e.g. (Towell.Shavlik:1994, Garcez.Broda.ea:2002)).

😊 In natural language processing, we can improve accuracy by inserting symbolic rules (Marques.Bader.ea:2007).

😊 Rules covering many training samples, are quickly acquired by back-propagation.

😊 Very specific rules embedded prior to the training will very likely be overwritten by newly learned rules.

► We can analyse the errors made by the network to obtain correcting rules.
A General Method for Guiding Backprop

1. Initialise the network.
2. Repeat until some stopping condition is satisfied:
   2.1 Train the network for a given number of cycles.
   2.2 Analyse the errors of the network to obtain correcting rules.
   2.3 Embed the rule(s) into the network.
UCI’s “Tic-Tac-Toe Endgame Data Set”

**Goal:** board is a win-situation for player X?

\[
\begin{array}{ccc}
X & O & \\
O & X & O \\
X & X & \\
\end{array}
\]
\[
\begin{array}{ccc}
X & X & O \\
O & O & X \\
X & O & X \\
\end{array}
\]
\[
\begin{array}{ccc}
X & O & X \\
O & O & \\
X & O & X \\
\end{array}
\]

- win
- no win
- no win
Obtain and Embedding Correcting Rules

Acquiring rules:
- Classification errors are positive or negative \((error > 0.5)\).
- Select cells and values where most samples agree.

**Example**

**rule:** \([b_{13}, b_3, o_{23}, x_{43}, x_{34}, x_{70}, o_7, b_1, b_1] \mapsto +\) (99 samples).

**template:** \([?, ?, o, x, x, x, ?, ?, ?] \mapsto +\) (covers 38 samples).

- Cells marked \(?\) are initialised to 0.0.
- Cells with equal/distinct input have weight \(\omega/\neg\omega\).
- Output connection is \(\omega/\neg\omega\) for positive/negative rules.
- Small random noise is added \([-0.05, 0.05])\).
Example for Encoding a Rule Template

Neuron for the rule $[?, ?, o, x, x, x, ?, ?, ?] \mapsto +$:

- Positive connections are solid lines
- Negative connections are dotted.
- Connections with small weights are omitted.
Mean Squared Error for Different Values of $\omega$

![Graph showing Mean Squared Error for Different Values of $\omega$]
Discussion

- For $\omega = 0$ network stops to improve at some point.
- For $\omega > 0$ network outperforms network $\omega = 0$.
- For $\omega = 5$ network did not learn very well.

Avoiding Local Minima
With errors on only a few samples, back-propagation can get into a local minima.

- For $\omega = 0$, the network gets stuck at some level.
- Our method can insert new units for those samples.
- Rules are not necessarily correct ($\omega = 5$ is too much).
- Some rules give better classification than their support.
Outline of the Course

- Introduction and Motivation
- The Core Method for Propositional Logic
- Applications of the Propositional Core Method
- The Core Method for First-Order Logic
- More on First-Order & other Perspectives