Neural-Symbolic Integration

A selfcontained introduction

INTERNATIONAL CENTER
FOR COMPUTATIONAL LOGIC

AIFB

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:

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} $._{pto}$

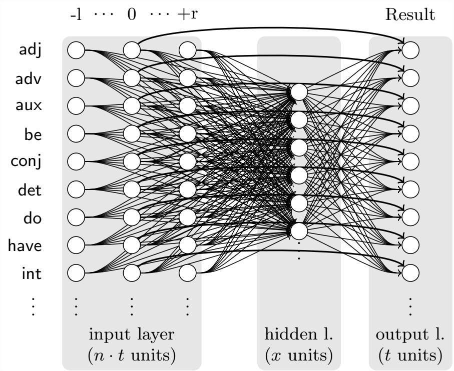
- Part of the dictionary extracted from the corpus:

Word	P_{adj}	P_{adv}	P_n	P_v	P_{prep}	...
right	42%	25%	33%			
following	40%			35%	25%	
beat	50%		8%	42%		
outside	17%	33%			50%	
grant			75%	25%		

- 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?



Wordclass Tagging: Background Knowledge 1

- Consider the following simple grammar:

$$s \rightarrow pto, np, vp, pto \qquad vp \rightarrow v, np \qquad optrel \rightarrow that, vp$$
$$vp \rightarrow v \qquad np \rightarrow pn \qquad \ldots$$

- Unfolding this grammar five times yields 88 sentences:

$$[pto, det, n, that, v, det, n, v, det, n, that, v, pn, pto],$$
$$[pto, det, n, that, v, det, n, v, det, n, that, v, pto],$$
$$[pto, det, n, that, v, pn, v, det, n, that, v, pn, pto], \quad \ldots$$

- Most frequent triples and their propositional rules:

$$[pto, det, n] \quad \rightsquigarrow \quad det \leftarrow pto_{-1}, n_{+1}$$
$$[det, n, that] \quad \rightsquigarrow \quad n \leftarrow det_{-1}, that_{+1}$$
$$[n, that, v] \quad \rightsquigarrow \quad that \leftarrow n_{-1}, v_{+1}$$

Wordclass Tagging: Background Knowledge 2

- *Exception rule*: "A noun? preceded by an auxiliary verb? is actually a verb!"
- Encoding as propositional rule:

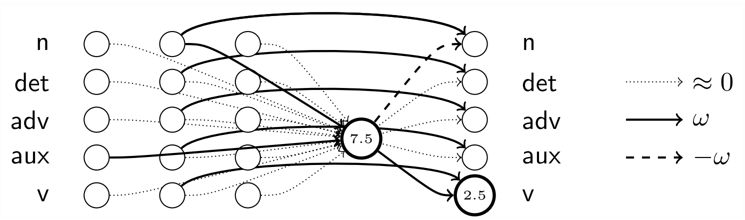
$$v \leftarrow aux_{-1}, n_0$$

- Known as e.g. "transformation rules"
- Generated by linguists or extracted from corpus

Wordclass Tagging: Embedding Rules

- Following the propositional core method we embedded rules into the network:

$$v \leftarrow aux_{-1}, n_0$$

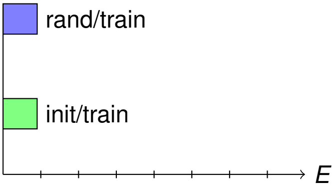


Wordclass Tagging: Results

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

Wordclass Tagging: Results

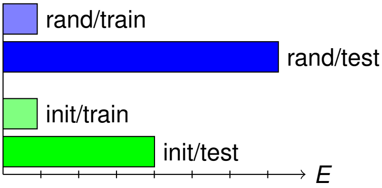
- To avoid overfitting, training data is split into parts
  - training data
  - test / validation data
- Error on training and test corpus:



Wordclass Tagging: Results

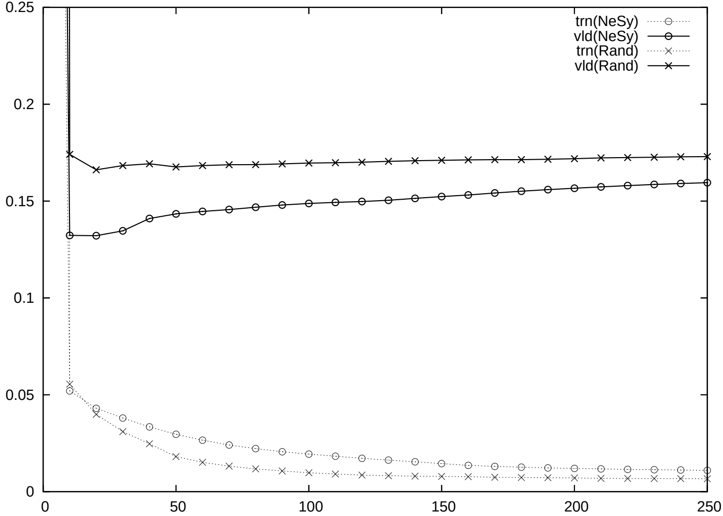
- To avoid overfitting, training data is split into parts
  - training data
  - test / validation data

- Error on training and test corpus:



Wordclass Tagging: Results

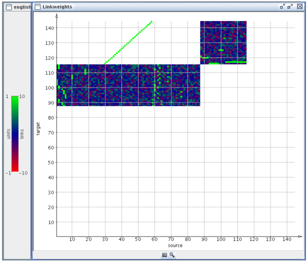
- Evolution of errors:



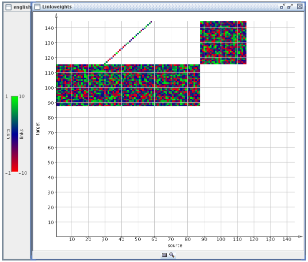
Wordclass Tagging: Results

- Evolution of connection weights:

NeSy



Randomised

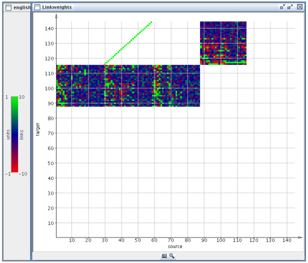


- ”Structure” is more or less preserved, only small changes.

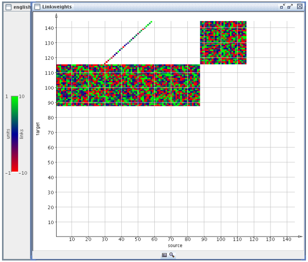
Wordclass Tagging: Results

- 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:

Neuro-Symbolic Word Tagging

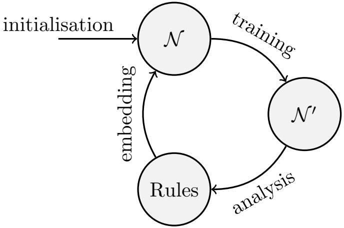
- Joint work with:
  - Nuno Marques (UNL)
  - Steffen Hölldobler (ICCL)
- "Guiding Backprop by Inserting Rules"
  (Marques, Bader, ea., 2008)

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"

(Asuncion.Newman:2007)

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

X		O
O	X	O
X		X

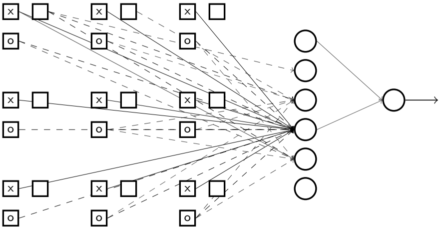
win

X	X	O
O	O	X
X	O	X

no win

X	O	X
O	O	
X	O	X

no win



Obtain and Embedding Correcting Rules

Acquiring rules:

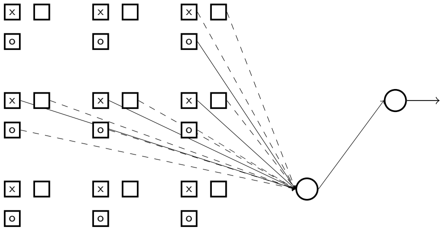
- Classification errors are positive or negative (*error* > 0.5).
- Select cells and values where most samples agree.

Example

rule: $[\mathrm{b}_{13}, \mathrm{b}_3, \mathrm{o}_{23}, \mathrm{x}_{43}, \mathrm{x}_{34}, \mathrm{x}_{70}, \mathrm{o}_7, \mathrm{b}_1, \mathrm{b}_1] \mapsto +$ (99 samples).
template: $[?, ?, \mathrm{o}, \mathrm{x}, \mathrm{x}, \mathrm{x}, ?, ?, ?] \mapsto +$ (covers 38 samples).

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

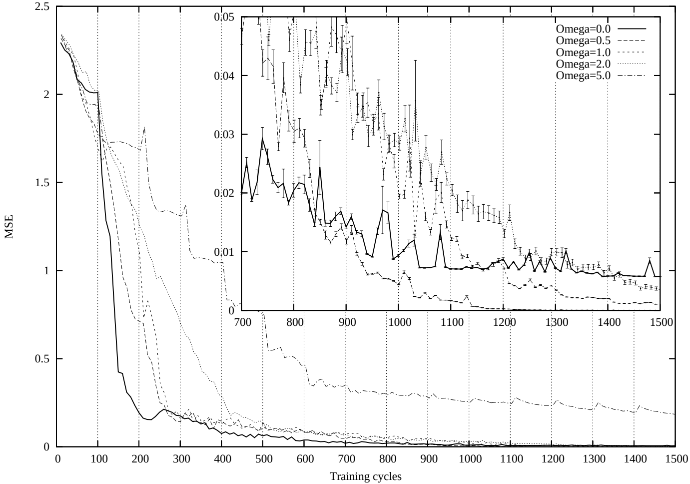
Example for Encoding a Rule Template



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

- Positive connections are solid lines
- Negative connections are dotted.
- Connections with small weights are omitted.

Mean Squared Error for Different Values of ω



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