Deep Learning for NLP

Stephen Pulman

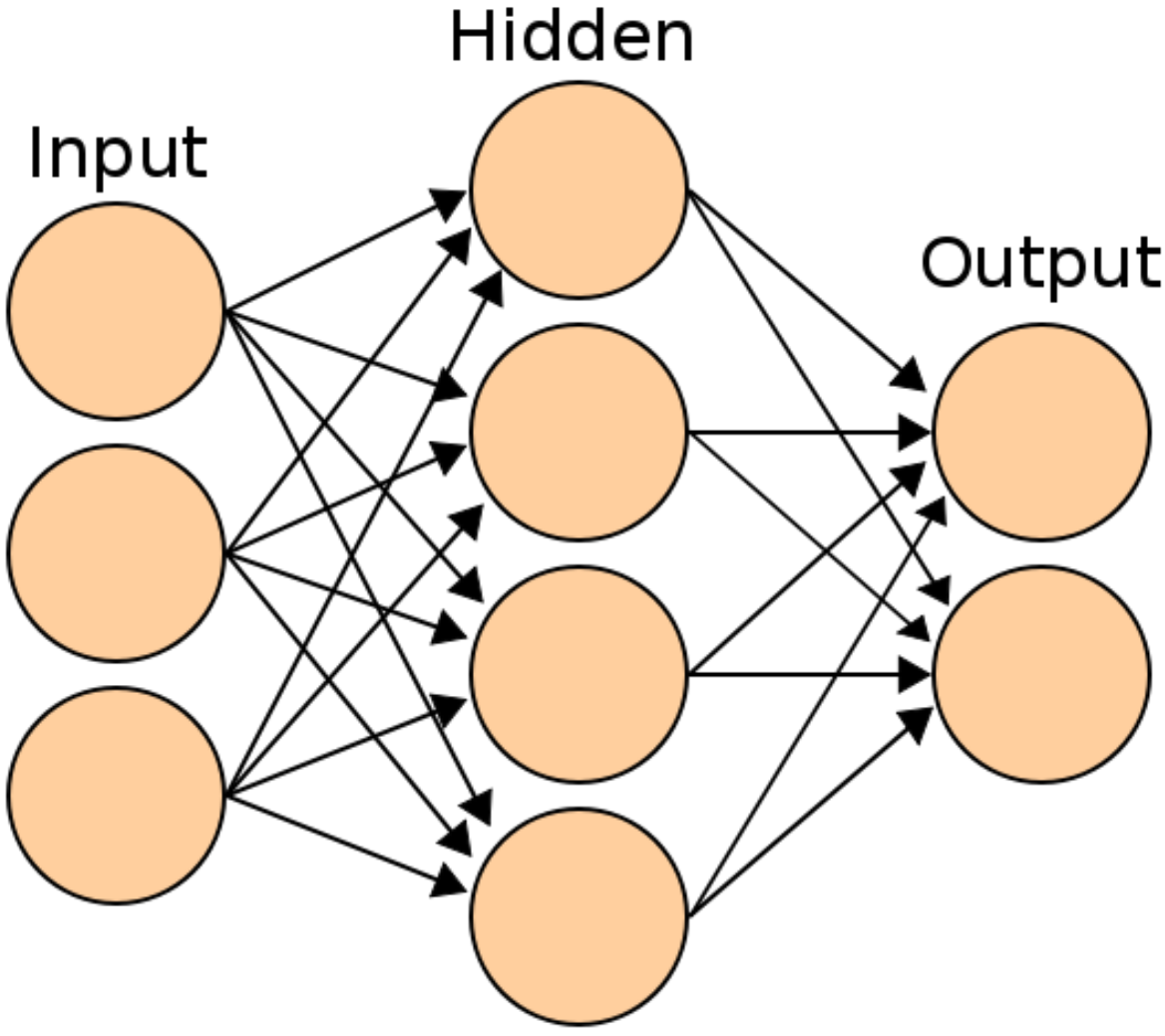
Dept. of Computer Science, Oxford University
and TheySay Ltd
stephen.pulman@cs.ox.ac.uk, stephen.pulman@theysay.io
@sgpulman

NYC Sentiment Symposium, March 6th 2014

TheySay

What is deep learning?

Familiar three layer neural network:



Properties

- Each unit connected via weights to every unit in next layer.
- Input layer is vector of feature values.
- Hidden units output goes through non-linearity like *tanh* or *sigmoid*.
- Output layer goes through 'softmax' to provide probability distribution over output values.
- Training via 'backpropagation':
  i.e. error (= difference between current and target output values) propagated back through network to adjust weights.

Deep learning

Problems with traditional NN:

- In principle many layers, in practice 3+ cannot be trained because of the 'vanishing gradient' problem.
- Input features usually hand-crafted.

Promise of 'deep learning':

- Unsupervised pre-training can help learn useful features by discovering regularities in the data.
- Multiple layers trained separately, corresponding to successive levels of abstraction.
- Advances in hardware: most computation in NN is matrix/vector operations so efficient GPUs can be used.
- Multi-core machines and clusters are cheap(ish), although parallelizing some algorithms is not trivial.

Pre-training in NLP

Most common kind of semi-supervised pre-training method is 'word embeddings':[1]

- Create randomly initialised vectors for each word (varying lengths, say 50).
- Create training data by concatenating vectors for 5-gram (positive example) and same 5-gram but with a randomly different word substituted for one word (negative example).
- Example: ...photographer visits Syrian refugees in...
  vs. ...photographer visits Syrian refugees quark...
- Use traditional NN to compute score for each, training objective: *score(pos)* $>$ *score(neg)*.
- Simultaneously propagate weight changes to word vectors.
- Resulting vectors implicitly represent contextual and cluster properties of words.

[1] Turian et al., 2010, http://www.newdesign.aclweb.org/anthology/P/P10/P10-1040.pdf, Mikolov et al., 2013, http://aclweb.org/anthology/N/N13/N13-1090.pdf

Some interesting properties

Clustering - nearest neighbours within vector space to:

France:	Austria, Belgium, Germany, ...
scratched:	nailed, smashed, punched, scraped, slashed,...
Xbox:	Amiga, Playstation, MSX, iPod,...

Analogy - subtract one vector, then add another, then find nearest neighbour:

Paris	− France	+ Italy	= Rome
sushi	− Japan	+ Germany	= bratwurst
but:		+ France	= tapas
and:		+ USA	= pizza!
France	− Sarkozy	+ Berlusconi	= Italy
		+ Merkel	= Germany

So these word embeddings seem to really capture something meaningful.

SENNA

Collobert [2] uses such embeddings as pre-training for a general NN model which achieves excellent results on standard NLP tasks like:

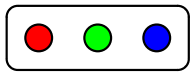
- Part Of Speech tagging
- NP chunking
- Named Entity Recognition
- Semantic Role Labelling

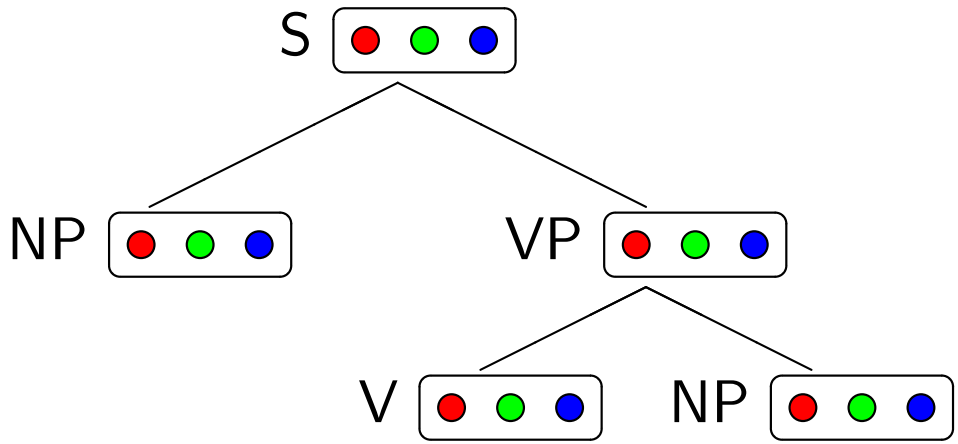
Task	Benchmark	SENNA	Previous best
POS	97.24	97.29	Toutanova et al. (2003)
NP	94.29	94.32	Sha and Pereira (2003)
NER	89.31	89.59	Ando and Zhang (2005)
SRL	77.92	75.49	Koomen et al. (2005)

[2] Collobert et al. 2011 `http://ronan.collobert.com/pub/matos/2011_nlp_jmlr.pdf`

Parsing

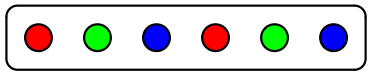
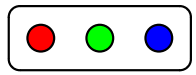
Several groups[3] have shown that combining these word embeddings with parsing leads to improved performance on a number of tasks like paraphrase detection, or sentiment analysis.

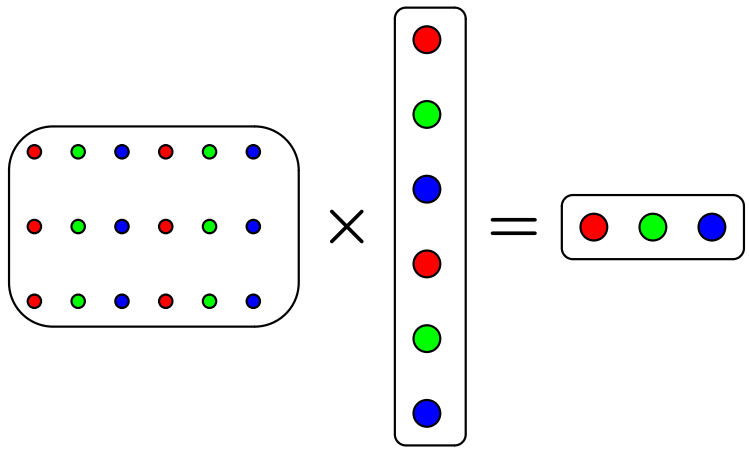
Socher's 'Compositional Vector Grammar' parser: given some training data, and a standard PCFG, represent words and, via composition, phrases, as vectors  .



[3] Hermann and Blunsom, 2013 http://aclweb.org/anthology//P/P13/P13-1088.pdf
Socher et al 2013a http://aclweb.org/anthology//P/P13/P13-1045.pdf,
Socher et al 2013b http://aclweb.org/anthology//D/D13/D13-1170.pdf

Parsing

To compute VP's vector, we concatenate those of V and NP to give  , and learn from the training data a weight matrix specific to the rule **VP** $\rightarrow$ **V NP** or **S** $\rightarrow$ **NP VP** which combines them in the 'right' way to form a  :



- In effect a separate NN for each syntax rule, using the usual methods: backpropagation, non-linearities, etc.
- Scores can be combined with rule probabilities.
- Resulting system used to filter output of the Stanford Parser
- Improves accuracy from 86.6% to 90.4%

Words of caution

This all looks great, but...

- Lots of hype about deep learning, which has been naively reported in the press.
- Is this setting us up for another AI winter?
- Improvements via DL are generally tiny, and often DL not the best performing systems (e.g. Berkeley and Charniak parsers are more accurate than Stanford's)
- In fact, traditional rule based methods currently best for some tasks: e.g. NER (91.77% vs. 89.59%)[4] and for coreference resolution.[5]

[4] Chiticariu et al. 2013 http://aclweb.org/anthology//D/D13/D13-1079.pdf

[5] Lee et al. 2013 http://aclweb.org/anthology//J/J13/J13-4004.pdf