Deep Learning

Lecture 9

Fernando Perez-Cruz
based on Thomas Hofmann lectures

Swiss Data Science Center
ETH Zurich and EPFL – datascience.ch

November 19, 2018

Overview

1. Memory Units
2. Differentiable Memory
3. Attention
4. Recursive Networks

Section 1

Memory Units

Long-Term Dependencies

Sometimes: important to model long-term dependencies $\Longrightarrow$ network needs to memorize features from the distant past

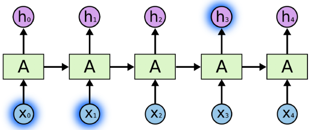
Recurrent network: hidden state needs to preserve memory
Conflicts with short-term fluctuations and vanishing gradients.

Conclusion: difficult to learn long-term dependencies with standard recurrent network (see DL Section 10.7)

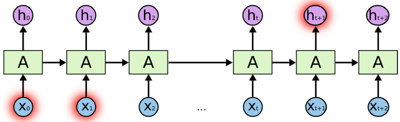
Popular remedy: gated units

RNNs Limitations

Short Term: "the clouds are in the *sky*"



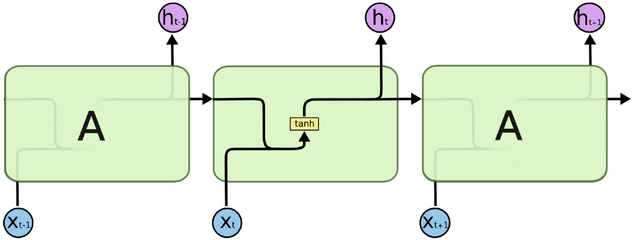
Long Term: "I grew up in France ... I speak fluent *French*"



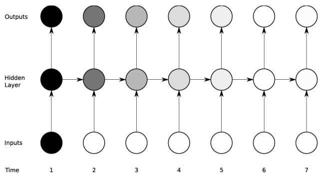
from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

RNNs Limitations

RNNs classic architecture forgets easily



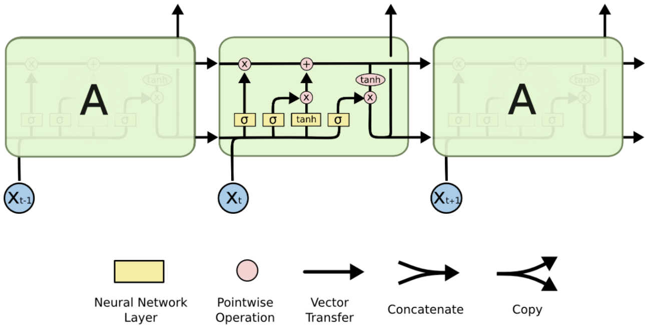
The repeating module in a standard RNN contains a single layer.



from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

LSTM: Overall Architecture

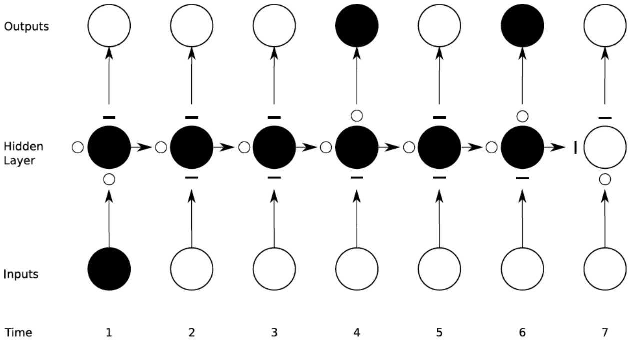
Long-Short-Term-Memory: complex unit for memory management



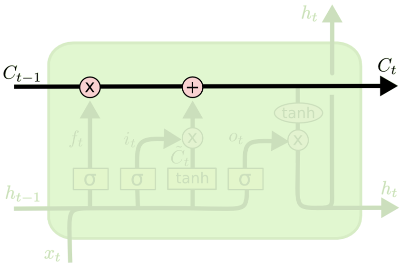
from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

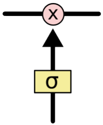
LSTM: Forgetting remembering

Long-Short-Term-Memory: Remembering information for long time and forgetting it fast.



LSTM: Flow of Information



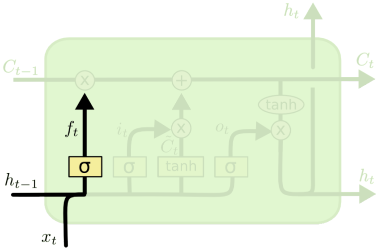


Information propagates along the chain like on a conveyor belt.

Information can flow unchanged and is only selectively changed (vector addition) by σ-gates.

from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

LSTM: Forget Gate

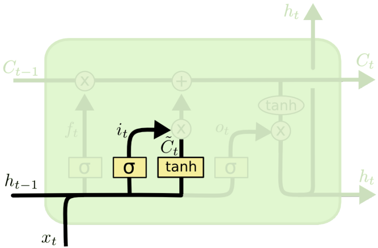


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Keeping or forgetting of stored content?

from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

LSTM: Input $\rightarrow$ Memory Value



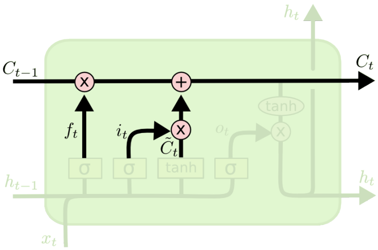
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Preparing new input information to be added to the memory.

from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

LSTM: Updating Memory

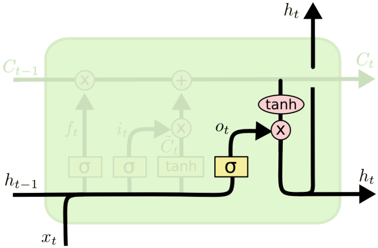


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Combining stored and new information.

from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

LSTM: Output Gate



$$o_t = \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Computing output selectively.

from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

Long Short-Term Memory Units

Mathematically, simple! (back to our notation)

Affine functions for input processing (1), input gate (2), forget gate (3), output gate (4):

$$F^\kappa = \sigma \circ \bar{F}^\kappa, \quad \bar{F}^\kappa = \mathbf{W}^\kappa \mathbf{h}^{t-1} + \mathbf{U}^\kappa \mathbf{x}^t + \mathbf{b}^\kappa, \quad \kappa \in \{1, 2, 3, 4\}$$

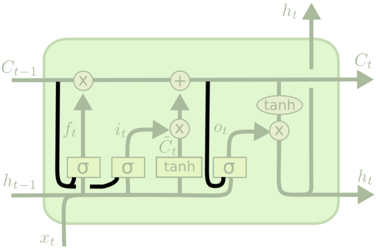
Next state (with pointwise multiplications)

$$\mathbf{C}^t = F^3(\dots) \odot \underbrace{\mathbf{C}^{t-1}}_{\text{self}} + F^2(\dots) \odot \underbrace{F^1(\dots)}_{\text{net in}}$$

Sigmoid output

$$\mathbf{h}^t = F^4(\dots) \odot \tanh(\mathbf{C}^t)$$

LSTM With Peepholes

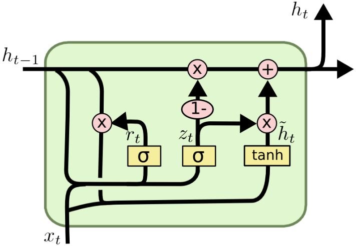


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

Adding peephole connections (Gers & Schmidhuber, 2000).

from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

Gated Memory Units



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Memory state = output. Modifications to logic. (Cho et al, 2014).

Convex combination of old and new information.

from Christopher Olah: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

Gated Memory Units

GRUs and LSTMs can learn active memory strategies: what to memorize, overwrite and recall when.

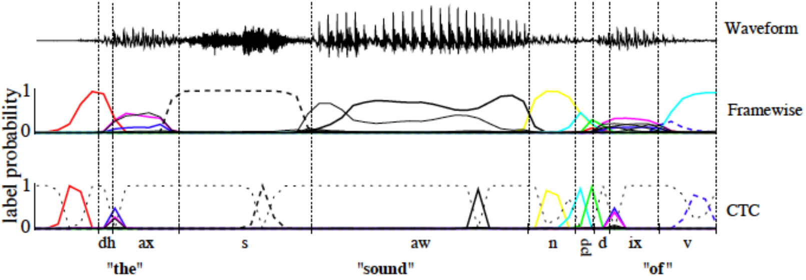
Successful use cases

- handwriting recognition
- speech recognition (also: Google)
- machine translation
- image captioning

Notoriously difficult to understand what units learn...

Resource-hungry. Slow in learning.

Unsegmented Sequences



Connectionist Temporal Classification

Allows to estimate the sequences of unsegmented data.

- Simple model:

$$p(\boldsymbol{\pi}|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}$$

where π_t is a distribution over all posible labels + *blank*

- Map from many to one:

$$p(\boldsymbol{\ell}|\mathbf{x}) = \sum_{\pi \in \mathrm{B}^{-1}(\boldsymbol{\ell})} p(\boldsymbol{\pi}|\mathbf{x})$$

Removing repeated symbols and blanks: $\boldsymbol{\ell} = aab$

$$\mathrm{B}(a - ab-)$$
$$\mathrm{B}(-aa - -abb)$$

Forward Backward for CTC

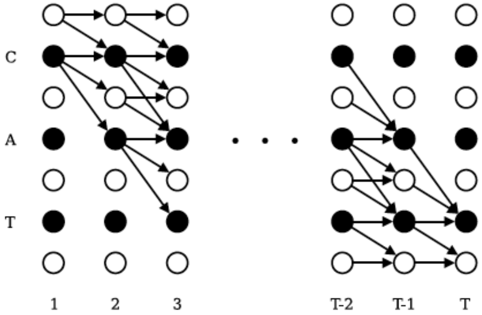


Figure 7.2: **CTC forward-backward algorithm.** Black circles represent labels, and white circles represent blanks. Arrows signify allowed transitions. Forward variables are updated in the direction of the arrows, and backward variables are updated against them.

Language Modeling

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

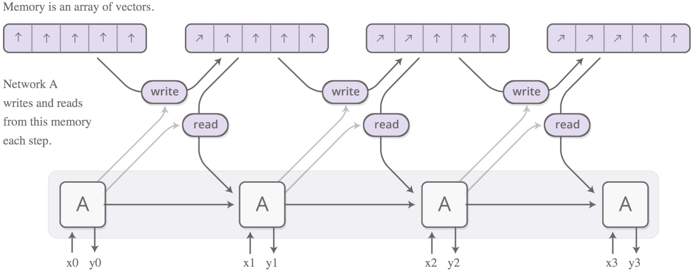
from Jozefowicz et al, 2016

- evaluation on corpus w/ 1B words
- number of parameters can be in the 100Ms or even Bs!
- ensembles can reduce perplexity to ≈ 23 (best result 06/2016)

Section 2

Differentiable Memory

Neural Turing Machine: Architecture

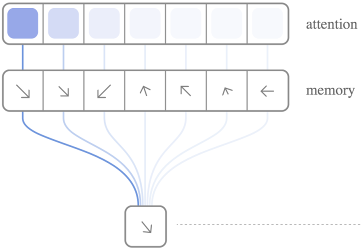


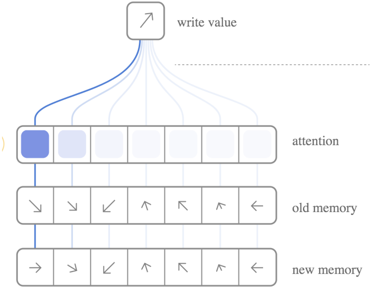
RNN controls an external memory bank

Reminiscent of Turing machine, but: each cell $M_i \in \mathbb{R}^d$

from Olah & Carter, 2016: `http://distill.pub/2016/augmented-rnns`

Neural Turing Machine: Differentiable Memory





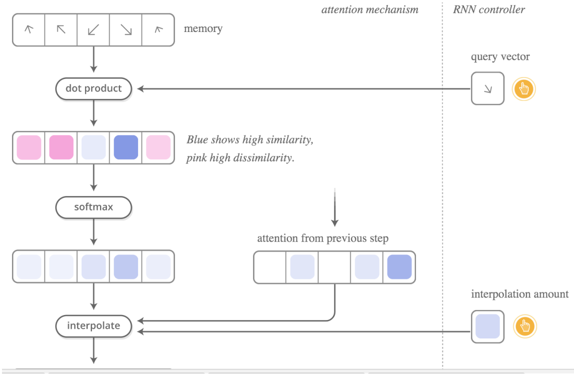
Compute attention distribution $(\alpha_i)_i$, $\alpha_i \geq 0$ s.t. $\sum_i \alpha_i = 1$.

Read out expected memory content: $r \leftarrow \sum_i \alpha_i M_i$.

Write uses weights $(\beta_i)_i$, $\beta_i \in [0;1]$, $M_i \leftarrow (1-\beta_i)M_i + \beta_i w$.

from Olah & Carter, 2016: `http://distill.pub/2016/augmented-rnns`

Neural Turing Machine: Memory Controller (1 of 2)

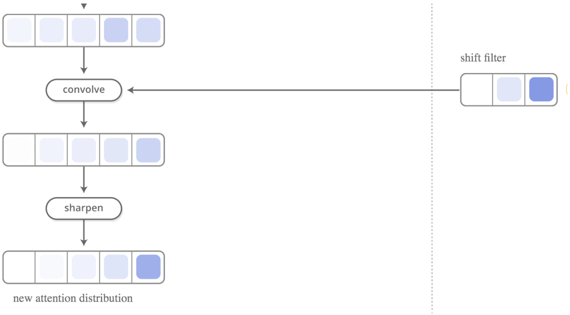


Associative memory access with query (key) $q \in \mathbb{R}^d$.

Cells are scored via softmax.

from Olah & Carter, 2016: `http://distill.pub/2016/augmented-rnns`

Neural Turing Machine: Memory Controller (2 of 2)



Ability to shift relative to content-selected locations (convolution).

Additional accentuation to sharpen attention distribution.

from Olah & Carter, 2016: `http://distill.pub/2016/augmented-rnns`

Differentiable Memory: Discussion

- NTM architectures can learn loops and simple programs. However, no real-world applications.
- Other architectures:
  - Neural random access machines (Kurach et al, 2015)
  - Differentiable data structures like stacks and queues (Grefenstette et al., 2015; Joulin & Mikolov, 2015)
- Direction of future research

Section 3

Attention

Attention Mechanisms

Simple way to overcome some challenges of RNN-based memorization: attention mechanism

Selectively attend to inputs or feature representations computed from inputs.

RNNs: learn to encode information relevant for the future.

Attention: select what is relevant from the past in hindsight!

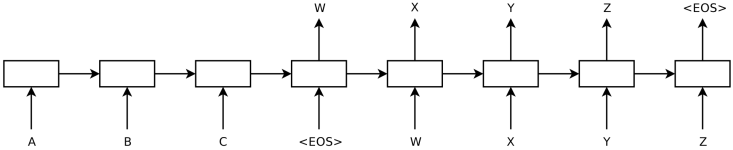
- both ideas can be combined

Sequence to Sequence Learning

Let us take an important example: sequence to sequence learning.

Seminal paper: Sutskever, Vinyals & Le, 2014

Encoder-decoder architecture



Encode sequence (e.g. sentence) into vector

Decode sequence (e.g. translate) from vector (w/ output feedback)

- Hinton: "thought vectors"

RNN Encoder/Decoder

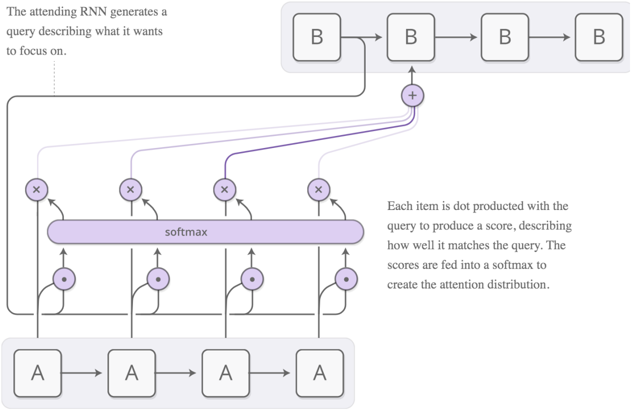
How to make this work? (Sutskever, Vinyals & Le, 2014)

- Deep LSTMs (multiple layers, e.g. 4)
- Different RNNs for encoding and decoding
- Beam search for decoding
- Reverse order of source sequence
- Ensemble-ing

Machine translation task

- State-of-the art results on WMT benchmarks
- however, traditional approaches: sentence aligment models (!)
- ... what is the equivalent in a neural architecture?

Seq2seq with Attention



from Olah & Carter, 2016: `http://distill.pub/2016/augmented-rnns`

Attend to the hidden state of the encoding RNN!

Seq2seq with Attention: MT Example

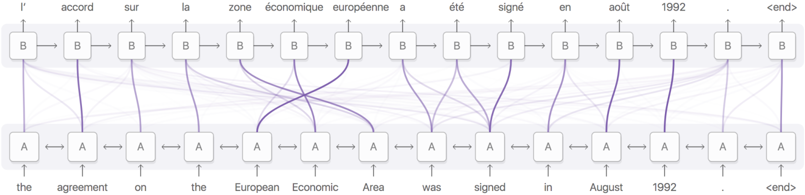


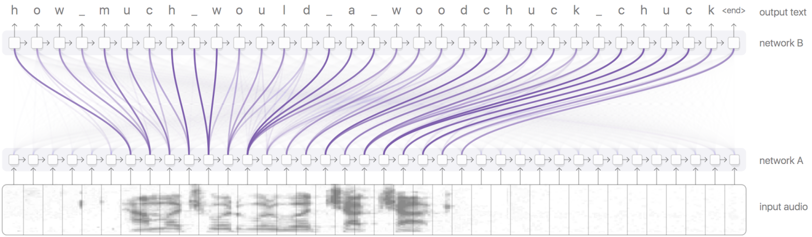
Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

from Olah & Carter, 2016: `http://distill.pub/2016/augmented-rnns`

Interpretable attention model (akin to alignments)

Bahdanau, Cho & Bengio, 2014

Seq2seq with Attention: Speech Recognition



from Olah & Carter, 2016: `http://distill.pub/2016/augmented-rnns`

Listen, Attend and Spell Model (Chan et al, 2015)

Attention is all you need?

- Remove the LSTM for the input.
- Self-Attention for the input and output.
- Positional Encoding: Sinusoids.
- Single hidden layer Feed forward NN.

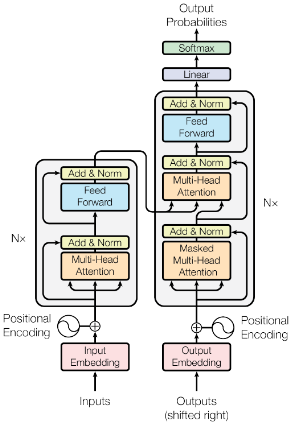


Figure 1: The Transformer - model architecture.

Vaswani et al, 2017: Attention is all you need.

Memory Networks

Memory networks (Weston et al, 2014; Kumar et al, 2015)

- operate over large data corpus (e.g. text documents)
- given question, learn to infer answers (QA retrieval)
- simplest form: memory is not altered
- recursive associative recall: given query q, find best matching memory cell i; use M_i and $\mathbf{x}$ as new key; repeat

Memory Networks

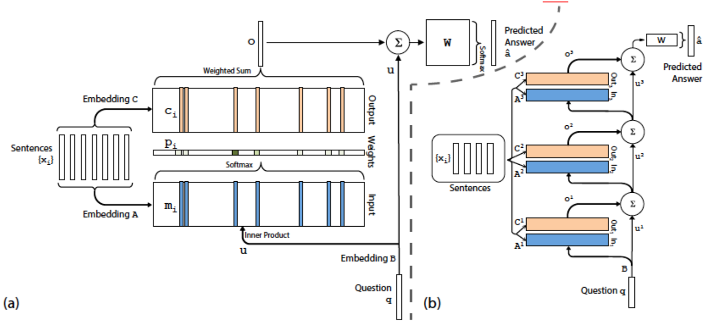


Figure 1: (a): A single layer version of our model. (b): A three layer version of our model. In practice, we can constrain several of the embedding matrices to be the same (see Section 2.2).

Sukhbaatar et al, 2015: End-To-End Memory Networks.

Memory Networks: Example

Sam walks into the kitchen.	Brian is a lion.	Mary journeyed to the den.
Sam picks up an apple.	Julius is a lion.	Mary went back to the kitchen.
Sam walks into the bedroom.	Julius is white.	John journeyed to the bedroom.
Sam drops the apple.	Bernhard is green.	Mary discarded the milk.
Q: Where is the apple?	Q: What color is Brian?	Q: Where was the milk before the den?
A. Bedroom	A. White	A. Hallway

Sukhbaatar et al, 2015: End-To-End Memory Networks.

Section 4

Recursive Networks

Recursive Networks

Recurrent networks = linear chain structure

Recursive networks = tree structure

- more flexibility in representing "grammar" like models
- depth efficient $\mathbf{O}(\log n)$ instead of $\mathbf{O}(n)$

Where does tree structure come from?

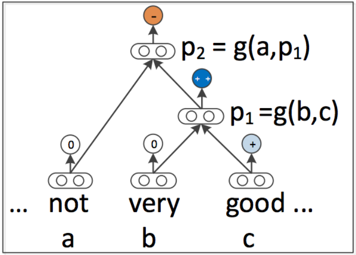
- produced by (reliable) parsing
- formal languages (programs, queries) etc.
- natural languages: syntactic parsing

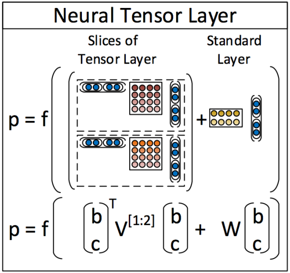
Recursive Networks

Basic composition operation

- recurrent network: $\mathbf{h}^t = F(\mathbf{h}^{t-1}, \mathbf{x}^t)$
- recursive network: $\mathbf{h}^{\mathsf{n}} = F(\mathbf{h}^{\mathsf{n.left}}, \mathbf{h}^{\mathsf{n.right}})$

Learn composition function $F : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$, which is then applied at each inner node of the tree.





Sentiment Analysis

Application of recursive networks: sentiment analysis.

Socher et al. 2013: Stanford sentiment Treebank.