Sequence modeling - 2

Alex Allauzen

2017-12-05

Outline

Plan

Language model / Generative sequence model

Applications

Automatic Speech Recognition, Machine Translation, OCR, ...

The goal

Estimate the **non-zero** probability of a word sequence given a vocabulary

$$P(w_1^L) = P(w_1, w_2, ...w_L) = \prod_{i=1}^{L} P(w_i|w_1^{i-1}), \quad \forall i, w_i \in \mathcal{V}$$

with the n**-gram assumption**:

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i|w_{i-n+1}^{i-1}), \quad \forall i, w_i \in \mathcal{V},$$

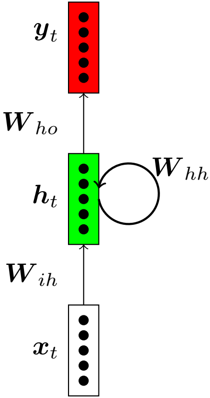
in the **recurrent** way

$$P(w_i|w_1^{i-1})$$

Plan

Recurrent network

Interlude introducing new architectures



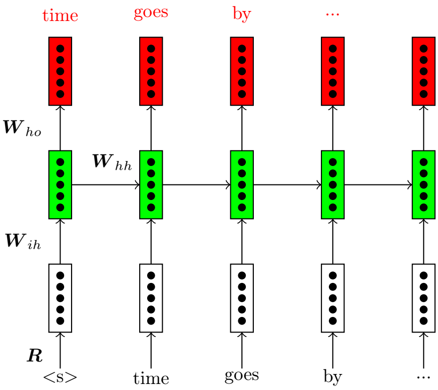
A dynamic system, at time t:

- maintains a hidden representation, the internal state: $\boldsymbol{h}_t$
- Updated with the observation of $\boldsymbol{x}_t$ and the previous state $\boldsymbol{h}_{t-1}$
- The prediction $\boldsymbol{y}_t$ depends on the internal state ($\boldsymbol{h}_t$)
- For a language model, $\boldsymbol{x}_t$ comes from word embeddings

The same parameter set is shared across time steps

Recurrent network language model

Unfolding the structure: a deep-network



At each step t

- Read the word $w_t \rightarrow \boldsymbol{x}_t$ from $\boldsymbol{R}$
- Update the hidden state $\boldsymbol{h}_t = f(\boldsymbol{W}_{ih}\boldsymbol{x}_t + \boldsymbol{W_{hh}h_{t-1}})$
- The word at $t+1$ can be predicted from $\boldsymbol{h}_t$:

$$\boldsymbol{y}_t = g(\boldsymbol{W}_{ho}\boldsymbol{h}_t)$$

- g is the softmax function over the vocabulary

Training recurrent language model

Training algorithm

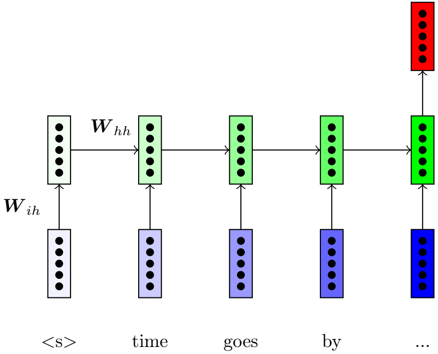
Back-Propagation through time or BPTT (Rumelhart et al.1986; Mikolov et al.2011):

- for each step t
  - compute the loss gradient
  - Back-Propagation through the unfolded structure

Inference

- Cannot be easily integrated to conventional approaches (ASR, SMT, ...)
- A powerful device for end-to-end system

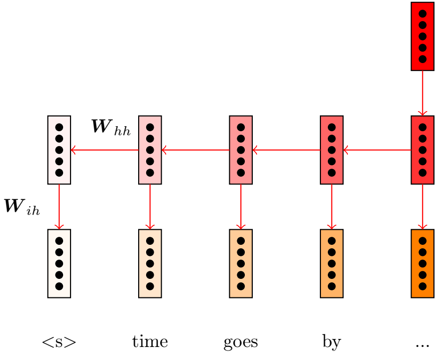
Short term memory



During inference :

- With the distance, the influence of observations reduces
- The memory is limited
- No way to keep/skip information

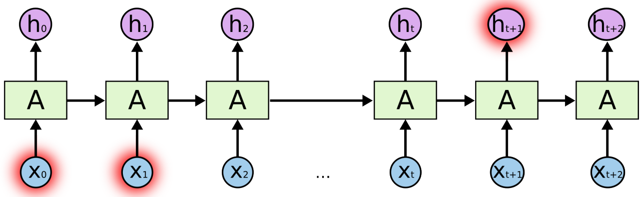
Vanishing gradient issue



During training:

- The gradient diminishes at each backward step
- No long term propagation of the gradient
- **It can also explode !**

The Problem of Long-Term Dependencies



ex: "I grew up in France… I speak fluent French"

- Recent observations hide the older ones (Bengio et al.1994)
- The vanishing (exploding) gradient is a real issue (Pascanu et al.2013)

Solutions

Improved optimization

- Gradient clipping (Pascanu et al.2013)
- Hessian-Free optimzation (Martens and Sutskever2012) or natural gradient (Desjardins et al.2013; Ollivier2015)

Modified unit

A recurrent network should be able to mitigate the observations *vs* its internal state:

- LSTM or Long-Short-Term-Memory cell (Hochreiter and Schmidhuber1997; Graves and Schmidhuber2009)
- Gated Recurrent Unit or GRU (Cho et al.2014)

Gradient clipping

A simple and efficient trick

Given a threshold γ, before each update:

- Compute the norm of the gradient (at each time step) : $||\nabla_{\boldsymbol{\theta}}||$
- If $||\nabla_{\boldsymbol{\theta}}|| > \gamma$:

$$\nabla_{\boldsymbol{\theta}} \leftarrow \frac{\gamma}{||\nabla_{\boldsymbol{\theta}}||}\nabla_{\boldsymbol{\theta}}$$

Challenges

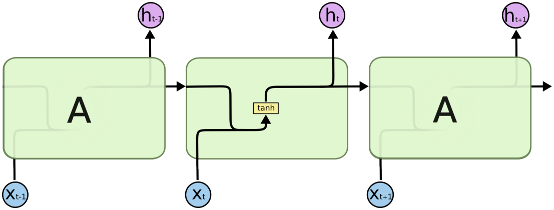
Share knowledge among similar words	☒
Keep/skip what is meaningful/meaningless	☒/☐
Long distance dependancies	☒/☐

Plan

Introduction to LSTM

The standard recurrent cell

- A recurrent cell is neural network layer
- Conveyor belt : the hidden state

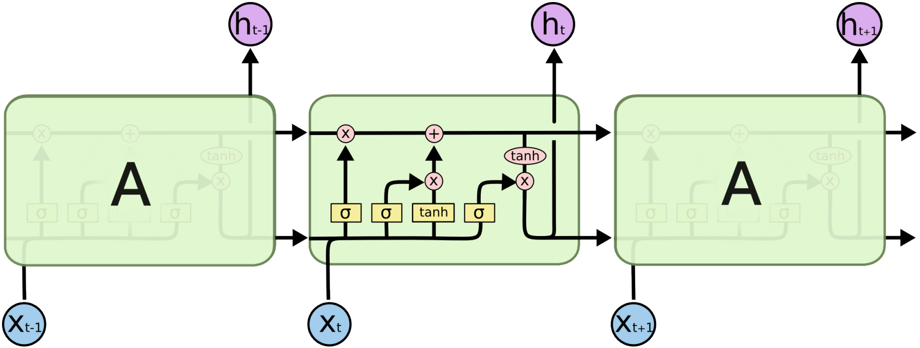


Lines carry a vectors

Based on `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Introduction to LSTM

The LSTM cell



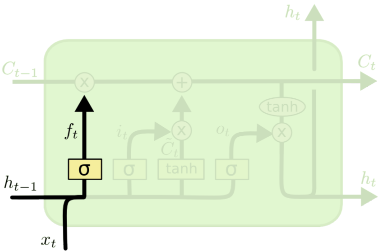
- LSTM introduces a second channel:
  the cell state
- The cell is now four neural layers, interacting in a very special way
- It acts as a memory
- Gates control the memory

Roadmap of inference

1. Memory organization
   1. what should be kept ?
   2. what shoud be updated ?
2. Update the cell state
3. Filter the state to provide the "hidden" state

LSTM : Control flow - 1

What should be forgotten from the previous cell state ?



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

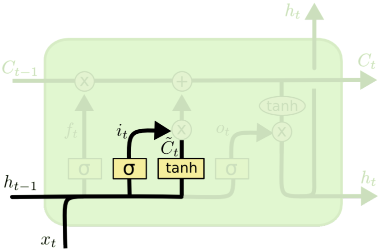
Action

The sigmoid (forget gate) answers for each component:

- 1: to keep it,
- 0 to forget it, or
- a value in-between to mitigate its influence

LSTM : Control flow - 2

What should be taken into account ?



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

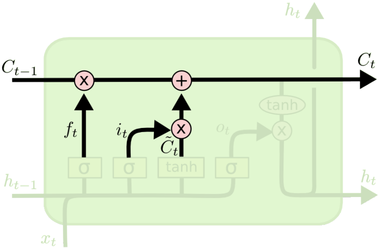
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Actions

- Create the update $\tilde{C}_t$ of the cell state
- and its contribution i_t (the input gate with a sigmoid activation)

LSTM : Control flow - 3

Write the new state



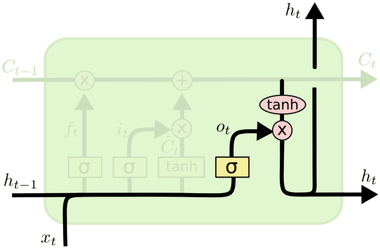
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Actions

- Merge the old cell state modified by the forget gate
- with the new input

LSTM : Control flow - 4

Write the new hidden state



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

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

Actions

- Decide what parts of the (filtered) cell state to output o_t
- Compute the hidden state

LSTM summary

A special kind of recurrent architecture

The internal cell state allows the model to:

- keep information in memory
- select the relevant outputs
- to reset or mitigate the long-term memory

Consequences

- An efficient model of sequences
- Overcome the vanishing gradient issue
- Very promising results in generation

Variants

- Gated recurrent units (GRU) (Cho et al.2014) or a more complicated one (Gers and Schmidhuber2000)
- Some recent comparisons (Józefowicz et al.2015; Greff et al.2015)

Plan

Challenges

Share knowledge among similar words	⊠
Keep/skip what is meaningful/meaningless	⊠⊠/☐
Long distance dependancies	⊠⊠/☐

Sequence modeling

n-gram models

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i | w_{i-n+1}^{i-1}), \quad \forall i, w_i \in \mathcal{V},$$

- A sliding window of fixed size $(n-1)$
- n can be wide
- A kind of 1-d convolution

Recurrent models

$$P(w_i | w_1^{i-1})$$

- The hidden state accumulates the memory of the past
- Difficult to optimize : exploding/vanishing gradient
- Long range dependancies are still an issue

Y. Bengio, P. Simard, and P. Frasconi.
1994.
Learning long-term dependencies with gradient descent is difficult.
Trans. Neur. Netw., 5(2):157–166, March.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio.
2014.
Learning phrase representations using rnn encoder–decoder for statistical machine translation.
In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Guillaume Desjardins, Razvan Pascanu, Aaron Courville, and Yoshua Bengio.
2013.
Metric-free natural gradient for joint-training of boltzmann machines.
In *International Conference on Learning Representations (ICLR'2013)*.

F.A. Gers and J. Schmidhuber.
2000.
Recurrent nets that time and count.
In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 189–194 vol.3.

Alex Graves and Juergen Schmidhuber.
2009.
Offline handwriting recognition with multidimensional recurrent neural networks.
In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber.
2015.
Lstm: A search space odyssey.
arXiv preprint arXiv:1503.04069.

Sepp Hochreiter and Jürgen Schmidhuber.
1997.
Long short-term memory.
Neural Comput., 9(8):1735–1780, November.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever.
2015.
An empirical exploration of recurrent network architectures.
In *Proceedings of the International Conference of Machine Learning (ICML)*, pages 2342–2350.

James Martens and Ilya Sutskever.
2012.

Training deep and recurrent networks with hessian-free optimization.

In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 479–535. Springer.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur.

2011.

Extensions of recurrent neural network language model.

In *Proceedings of ICASSP*, pages 5528–5531.

Yann Ollivier.

2015.

Riemannian metrics for neural networks i: feedforward networks.

Information and Inference.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio.

2013.

On the difficulty of training recurrent neural networks.

In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Proceedings*, pages 1310–1318. JMLR.org.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams.

1986.

Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. chapter Learning internal representations by error propagation, pages 318–362. MIT Press, Cambridge, MA, USA.