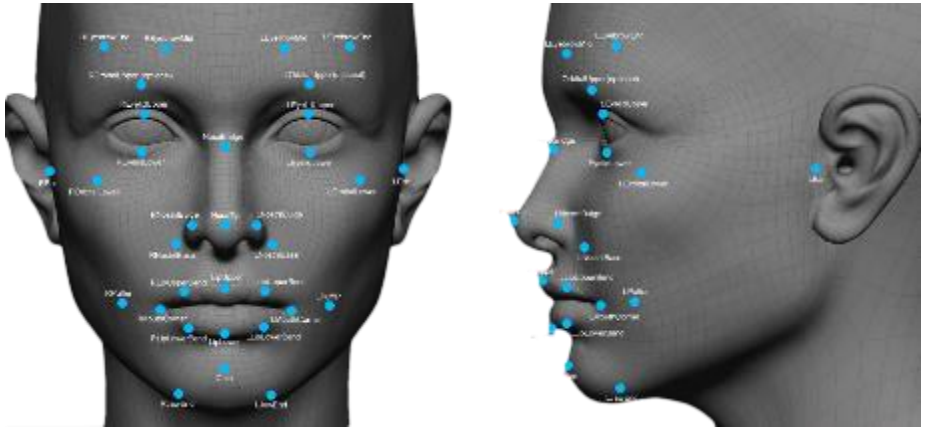


Recurrent neural networks. Part 2

Lecture # 11 of MIPT course: Introduction to machine learning

Speaker: Alexey O. Seleznev

PhD in Computational Chemistry

5VISION TEAM

Moscow

2017

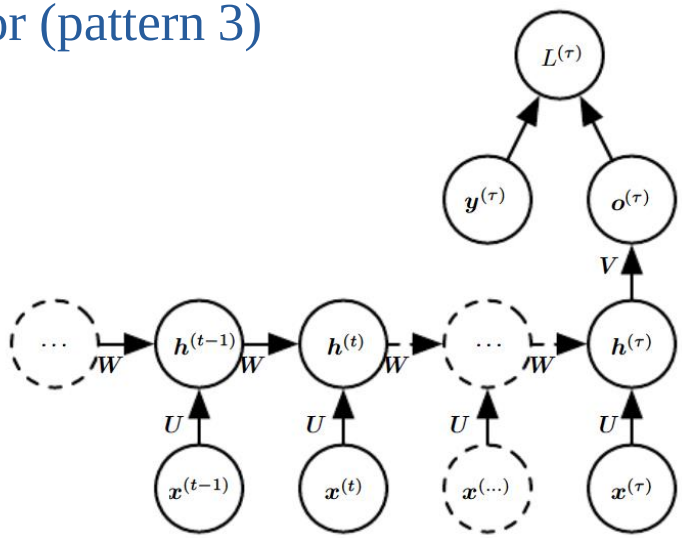
OUTLINE

- Sequence-to-sequence models
- Bidirectional RNNs
- Deep RNNs
- Soft-attention mechanism
- Discussion on assignment #3
- References

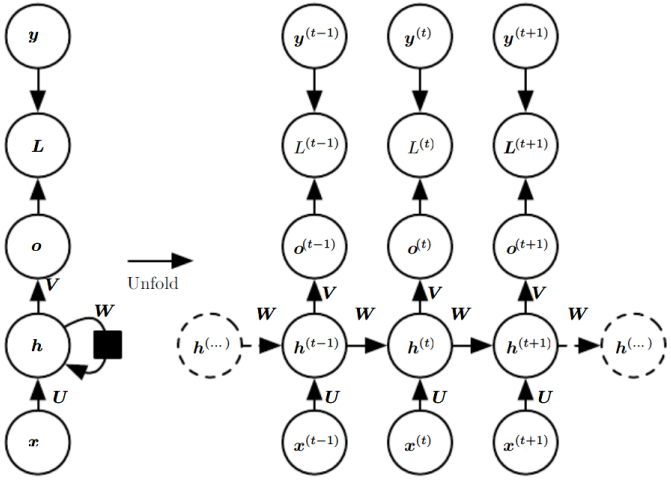
Sequence-to-Sequence Models

Sequence-to-Sequence Models

- We have seen that RNN can map an input sequence to
  - a fixed-size vector (pattern 3)



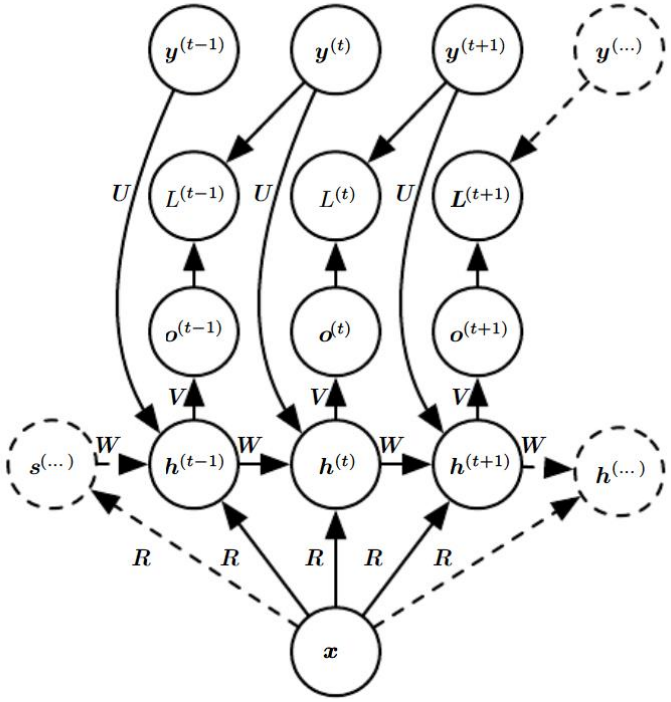
  - an output sequence of the same length (pattern 1 or 2)



Goodfellow et al (book)

Sequence-to-Sequence Models

- RNN can also map a fixed-size vector to a sequence (e.g. image capturing)

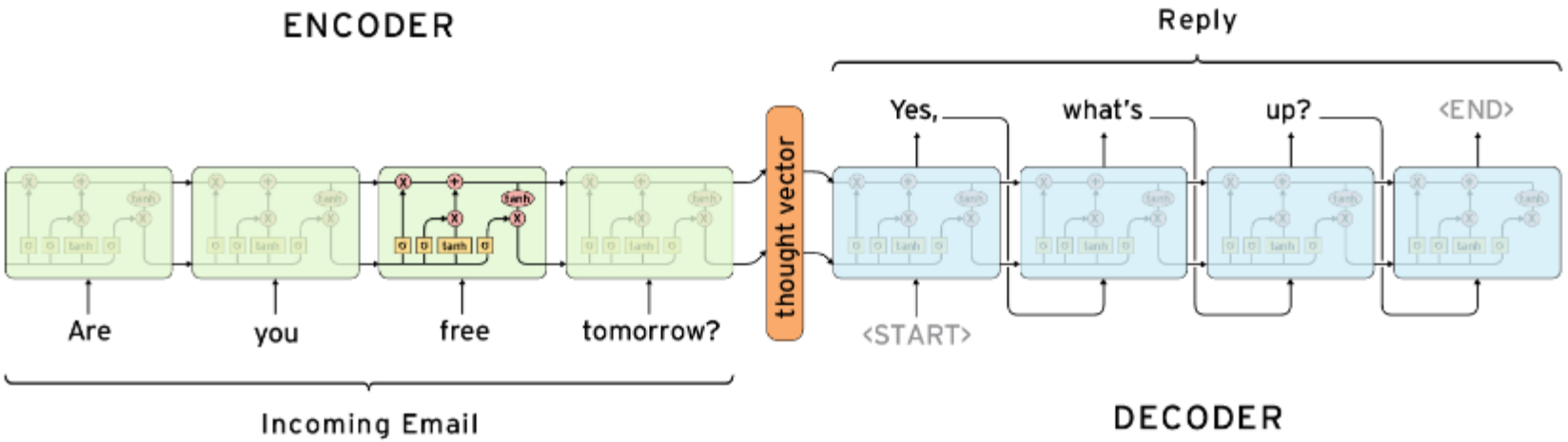


Goodfellow et al (book)

- Below we discuss how RNN can be trained to map an input sequence to an output sequence (**sequence-to-sequence**) of not necessarily the same length. That is of great importance in e.g. speech recognition and machine translation

Sequence-to-Sequence Models

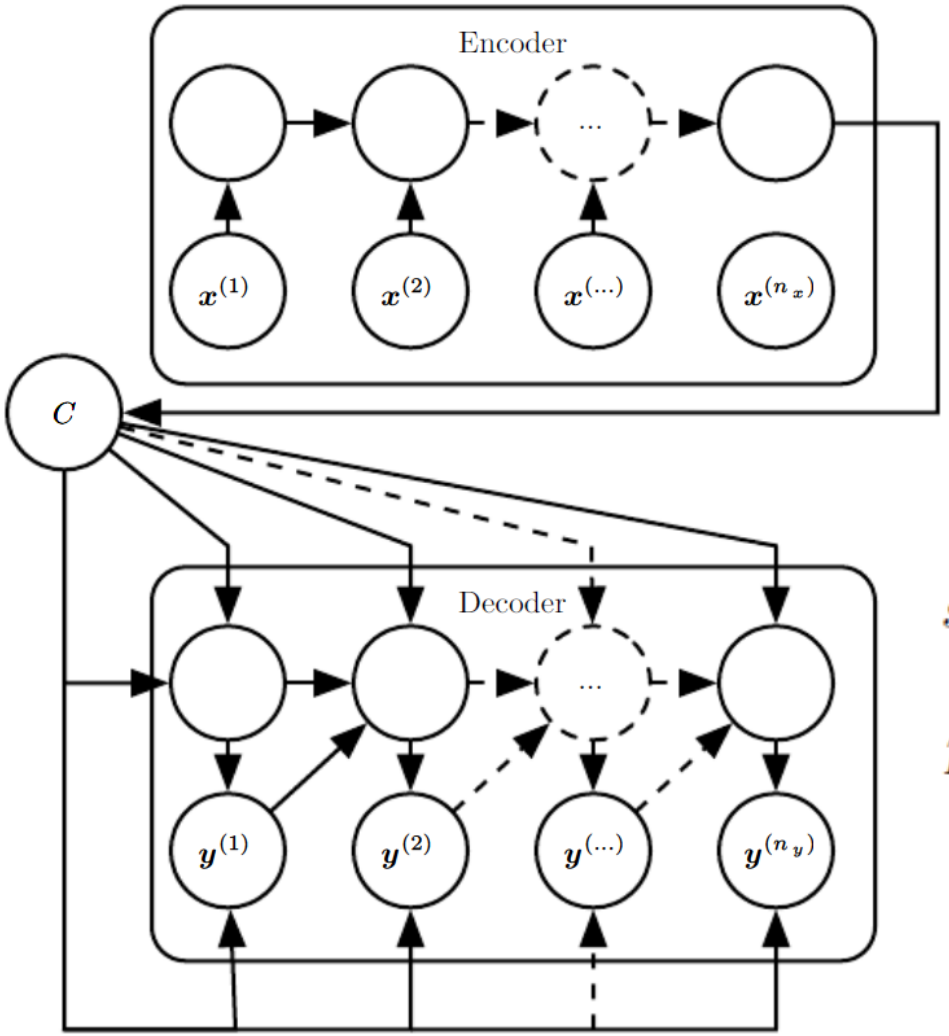
- The **idea of a sequence-to-sequence model**:
  - An encoder (or reader) RNN processes the input sequence **x** and returns the context C, usually a simple function of its final hidden state
  - A decoder (or writer) RNN is conditioned on C to generate the output sequence **y**
  - The two RNNs are trained jointly to maximize the average conditional probability $\log P(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n_y)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n_x)})$



http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/

Sequence-to-Sequence Models

- A more general scheme of sequence-to-sequence model



$$h_t = f\left(x_t, h_{t-1}\right)$$

$$c = q(\{h_1, ..., h_{n_x}\})$$

$$s_{t+1} = g(y_t, s_t, c)$$

$$p(\mathbf{y}) = \prod_{t=1}^{n_y} p(y_t | s_t, y_{t-1}, c)$$

Goodfellow et al (book)

Sequence-to-Sequence Models

- What to do if C dimension is too small to properly summarize a long input sequence?
  - Bahdanau et al (2014) suggested a solution where the model does not encode a whole input sentence into a single fixed-length vector. Instead, it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding

$$c = q(\{h_1, ..., h_{n_x}\}) \longrightarrow c_t = \sum_{j=1}^{n_x} \alpha_{tj} h_j$$

  - This frees the model from having to squash all the information of a source sentence, regardless of its length, into a fixed-length vector.
  - We will look at this model in greater detail after introducing soft-attention and bidirectional RNNs

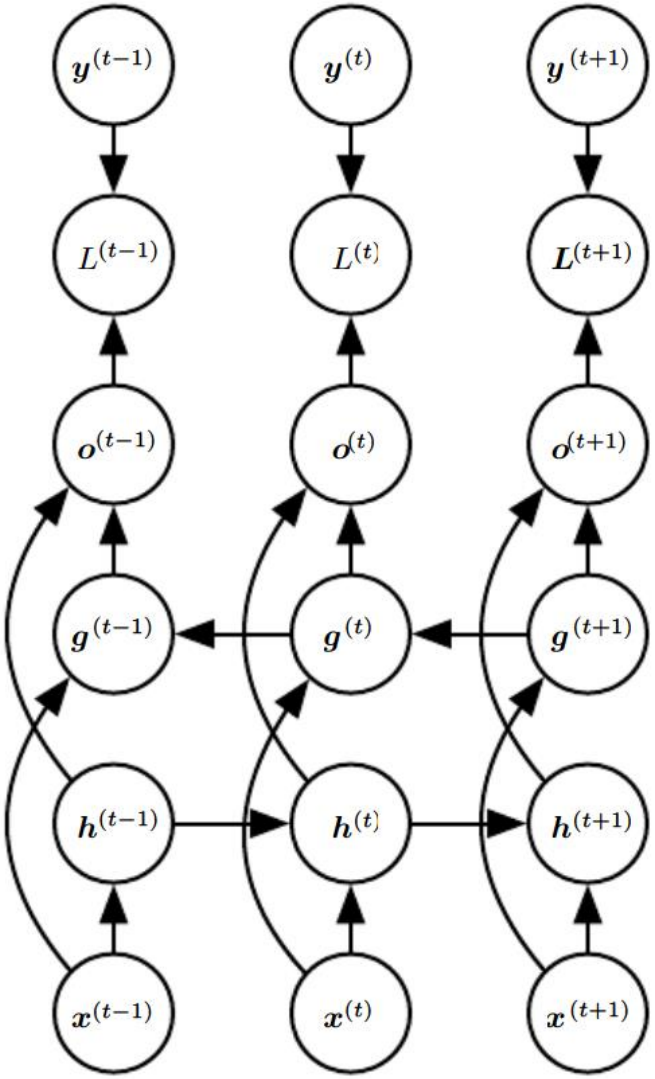
Bidirectional RNNs

Bidirectional RNNs

- The structure of RNN implies that the state at time t captures only information from the past and the present input.
- In many applications, however, we want to obtain a prediction that reflects the content of the **whole** input sequence
- Example: to correctly interpret a word having several meanings, one may need to know the next one “from the future”
- Bidirectional RNNs (Schuster and Paliwal, 1997) combine an RNN that moves forward through time with another RNN that moves backward through time.
- This allows to incorporate information from both the future and the past into the output. Nonetheless, the most important are the values around time t.

Bidirectional RNNs

- Typical architecture:



Goodfellow et al (book)

Bidirectional RNNs

- Some notes on BiRNNs:
  - It is not necessary that two RNNs in BiRNN are of the same type (simple RNN, LSTM, GRU)
  - The dimension of output is the sum of two RNNs hidden states dimensions
  - Hidden states dimensions can be different

Deep RNNs

Deep RNNs

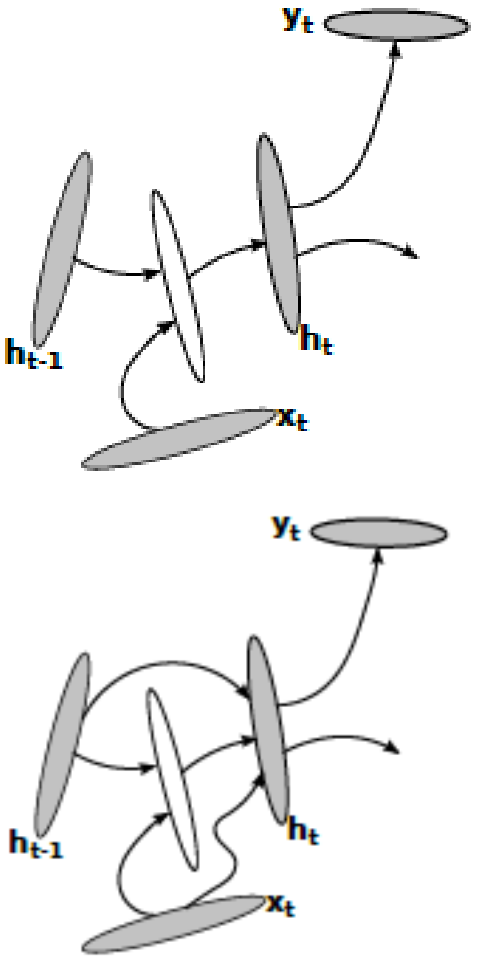
- Depth for feedforward models increases their expressiveness through distributed representation (do you remember what it is?)
- For RNNs, depth is ambiguous
  - They are already deep: when unfolded out in time, they have indefinitely many layers
  - But the primary function of the latter is to introduce memory, not hierarchical processing
  - RNNs learn to select what information they need to pass onwards, and what they need to discard
- What are the possible approaches to extending an RNN into a real deep RNN?

Deep RNNs

- Deep input-to-hidden networks
  - Motivation: higher-level representations of input should make it easier for RNN to learn the temporal structure between successive time steps (the same idea as for deep feedforward networks)
- Deep hidden-to-output networks
  - Motivation: disentangle the factors of variation in the hidden state of RNN, making it easier to predict the output (efficient summary of history)
- Deep hidden-to-hidden networks
  - The idea is to add dense layers between time steps to process the fixed-length hidden state

Deep RNNs

- A potential gain: hidden state can increase its adaptivity to quickly changing modes of input, while still preserving a useful summary of the past
- A potential problem: difficult to train due to the increase in the number of layers
- Possible solution: introduce shortcut connections



$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}) = \phi_h\left(\mathbf{W}_L^\top \phi_{L-1}\left(\mathbf{W}_{L-1}^\top \phi_{L-2}\left(\cdots \phi_1\left(\mathbf{W}_1^\top \mathbf{h}_{t-1} + \mathbf{U}^\top \mathbf{x}_t\right)\right)\right)\right)$$

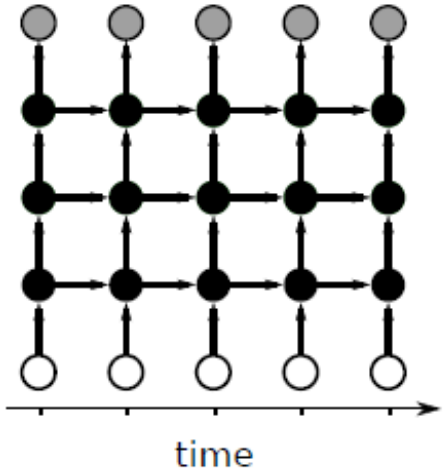
Pascanu et al (2014)

Deep RNNs

- Stack of hidden states (the most important one!)
  - The idea is to stack multiple recurrent hidden layers on top of each other

$$\mathbf{h}_t^{(l)} = f_h^{(l)}(\mathbf{h}_t^{(l-1)}, \mathbf{h}_{t-1}^{(l)}) = \phi_h\left(\mathbf{W}_l^\top \mathbf{h}_{t-1}^{(l)} + \mathbf{U}_l^\top \mathbf{h}_t^{(l-1)}\right)$$

  - Motivation: process the time series at several time scales (e.g. lower layers analyze short-term dependences, while higher levels concentrate on long-term dependences)



Break 5 min

Soft-Attention Mechanism

Soft-Attention Mechanism

- The idea:
  - Replace a set of vectors containing different pieces of information with either one of them (hard attention) or linear combination of them (soft attention) so that the most relevant information (as of the current time step) is preserved
- Motivation behind attention mechanism:
  - Reduce the number of optimized parameters without losing the ability to transmit relevant information
  - Visualize what and where the agent's attention is focusing on when making decisions
- To explain how soft attention works, let us return to the example of sequence-to-sequence model for machine translation

Soft-Attention Mechanism

- The reader is presented by a BiRNN
- The writer is a common RNN with soft attention

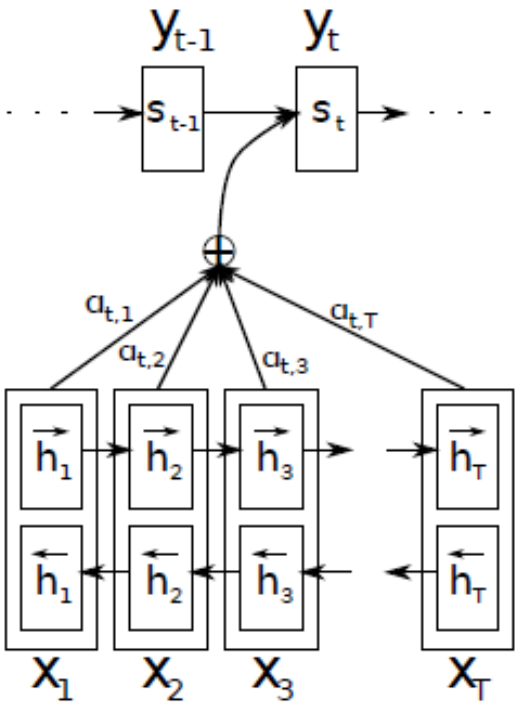
$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

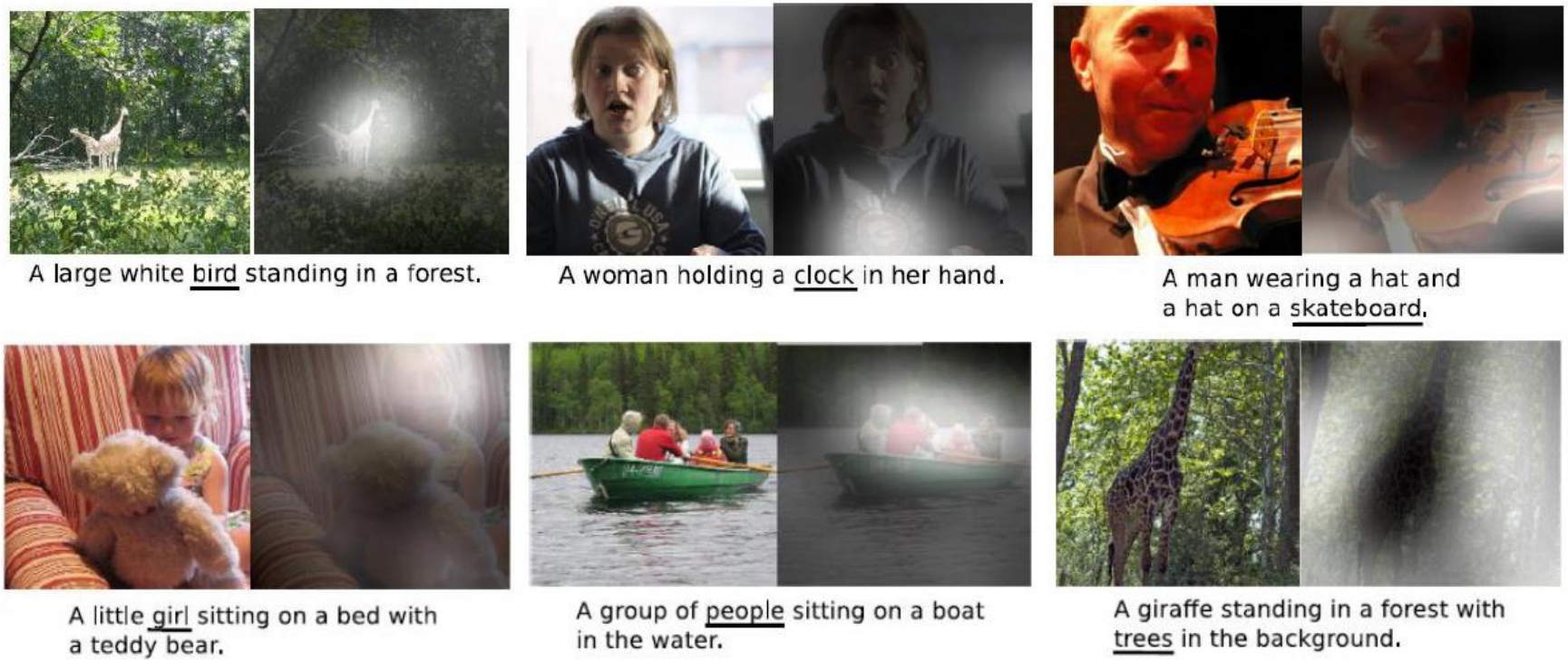
Bahdanau et al (2014)



- a is an alignment model which scores how well the inputs around position j and the output at position i match
- a is parametrized by a simple feedforward neural network which is jointly trained with other components of the network

Soft-Attention Mechanism

- We have demonstrated the importance of soft-attention for reducing the number of parameters (without it, C should have had a huge dimension)
- What about visualization of attention regions?



Xu et al (2015)

Soft-Attention Mechanism

Discussion on Assignment #3

Discussion on Assignment #3

- Assignment objectives:
  - Given a set of time series describing dynamics of mean values of smartphone's accelerometer and gyroscope, predict the type of human activity (walking, sitting, staying, etc)
- Main challenge:
  - High level of noise in test data. The need for strong regularizer
  - Low level of noise in training data
- Types of regularizers:
  - L1/L2 norms (should be tried)
  - Dataset augmentation
  - Noise robustness
  - Early stopping

References

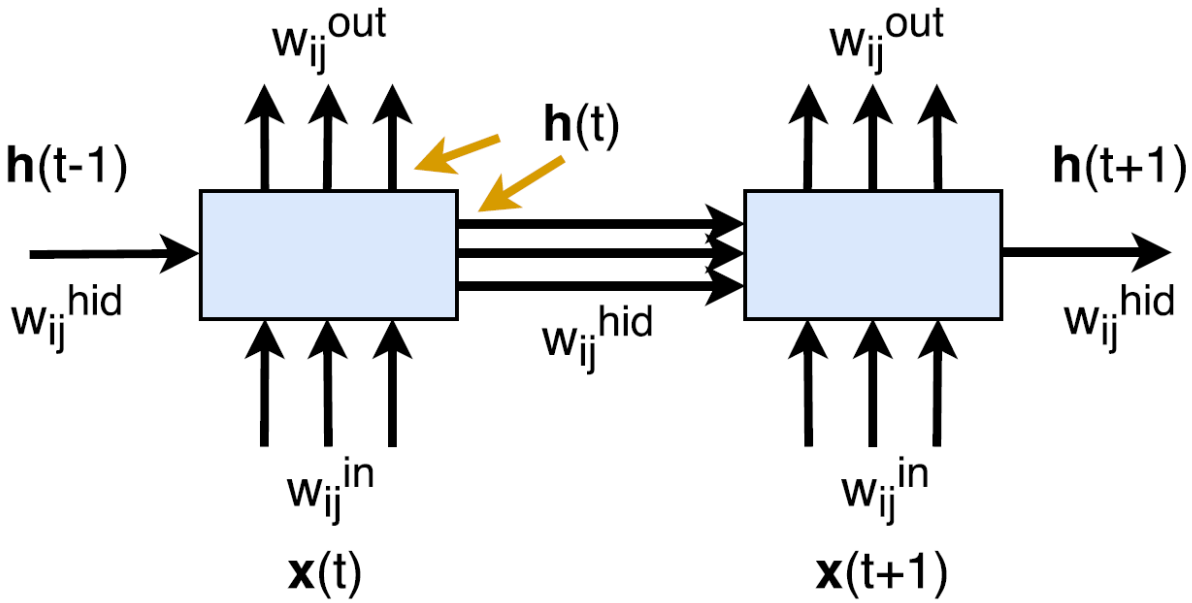
- I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. The MIT Press 2016
- D. Bahdanau, K. Cho. Y. Bengio *Neural Machine Translation by Jointly Learning to Align and Translate*. ArXiv 1409.0473 (2014)
- M. Schuster and K. Paliwal. *Bidirectional recurrent neural networks*. IEEE Transactions on Signal Processing, 45(11), 2673-2681 (1997)
- R. Pascanu, C. Gulcehre, K. Cho, Y. Bengio. *How to Construct Deep Recurrent Neural Networks?* ArXiv 1312.6026 (2014)
- K. Xu, J. Lei Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, Y. Bengio. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.* ArXiv 1502.03044

General Information

- Next Seminar:
  - Main topics:
    - Discussion on assignments
    - What would you like me to tell?
  - Schedule: 24 May, Wednesday 18:30
- Assignments:
  - Submissions for assignments #2 and #3 should be made through Kaggle

Backpropagation Through Time

- Backpropagation through time is similar to traditional backpropagation for an unfolded RNN (now n -> t):



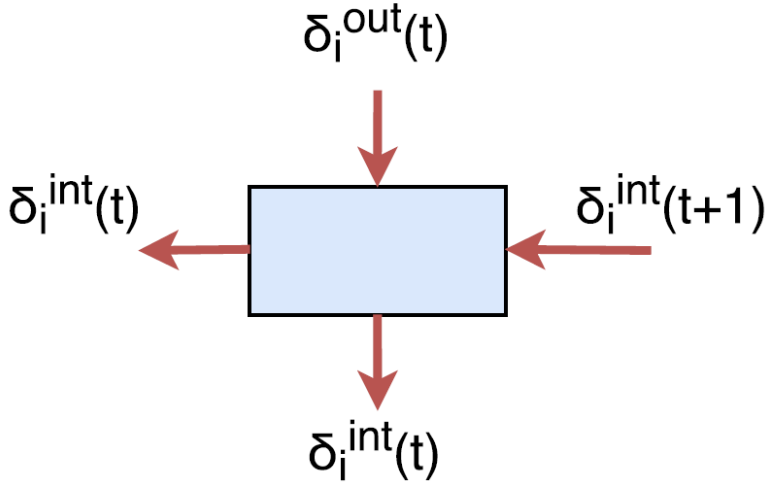
- Forward pass. Compute activations of internal and output units

$$x_i^{l+1}(t) = f^{out}\left(\sum_{j=1}^{K(l+1)} w_{ij}^{out} h_j(t)\right)$$

$$h_i(t) = f^{int}\left(\sum_{j=1}^{K(l)} w_{ij}^{in} x_j^l(t) + \sum_{j=1}^{K(l+1)} w_{ij}^{hid} h_j(t-1)\right)$$

Backpropagation Through Time

- Backward pass



$$\delta_i^{out}(t) = \frac{\partial Loss(t)}{\partial f}\frac{\partial f(u)}{\partial u}|_{u=z_i^{out}(t)} \quad \text{or}$$

$$\delta_i^{out}(t) = \delta_i^{l+1}(t) = \sum_{j=1}^{K(l+2)} \delta_j^{l+2}(t) w_{ji}^{l+2} \frac{\partial f^{out}(u)}{\partial u}|_{u=z_i^{out}(t)}$$

$$\delta_i^{int}(t) = \sum_{j=1}^{K(l+1)} \delta_j^{int}(t+1) w_{ji}^{hid} \frac{\partial f^{int}(u)}{\partial u}|_{u=z_i^{int}(t+1)} + \sum_{j=1}^{K(l+1)} \delta_j^{out}(t) w_{ji}^{out}$$

Backpropagation Through Time

- Weights update

$$w_{ij}^{out} = w_{ij}^{out} + \mu \sum_{t=1}^{T} \delta_i^{out}(t) h_j(t)$$

$$w_{ij}^{in} = w_{ij}^{in} + \mu \sum_{t=1}^{T} \delta_i^{int}(t) x_j(t) \frac{\partial f^{int}(u)}{\partial u}|_{u=z_i^{int}(t)}$$

$$w_{ij}^{hid} = w_{ij}^{hid} + \mu \sum_{t=1}^{T} \delta_i^{int}(t) h_j(t-1) \frac{\partial f^{int}(u)}{\partial u}|_{u=z_i^{int}(t)}$$

❖ In contrast to BP, BPTT has:

- Sum over t in loss function (vs mean over n in BP)
- Slow convergence
- Acute exploding & vanishing gradient problem