Recurrent neural networks

Machine Learning

Natalia Khanzhina

07.10.2019

Lecture plan

- Sequences and time series
- Recurrent neural networks
- RNNs with memory
- More connections
- Word representation

- The presentation is prepared with materials of
  - D. Polykovsky and K. Khrabrov “Neural networks in machine learning”
  - A. Ng “Recurrent neural networks”
- Slides are available online: **goo.gl/RihS9R**

Lecture plan

- Sequences and time series
- Recurrent neural networks
- RNNs with memory
- More connections
- Word representation

Application area

- Time series
- Natural language
- Speech
- Dynamical systems
- Images and videos

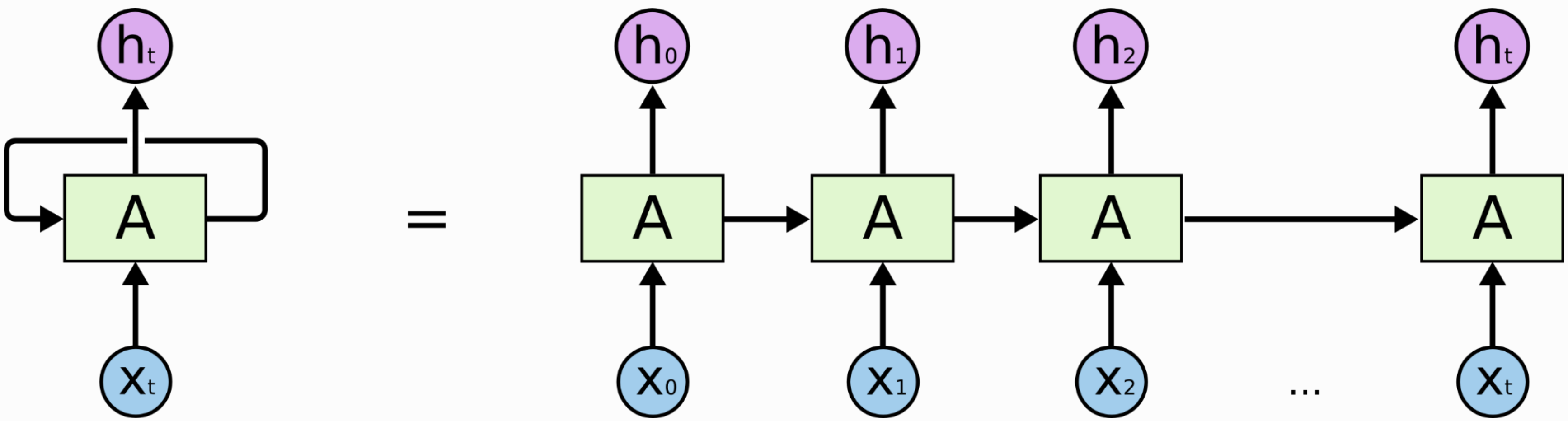
- In general, is state-of-the art for sequence processing

Sequence processing methods

- Spectral
- Time
- Time-frequency

Recurrent neural network

- Network with loops or unrolled network without loops



Lecture plan

- Sequences and time series
- Recurrent neural networks
- RNNs with memory
- More connections
- Word representation

Feedforward NN

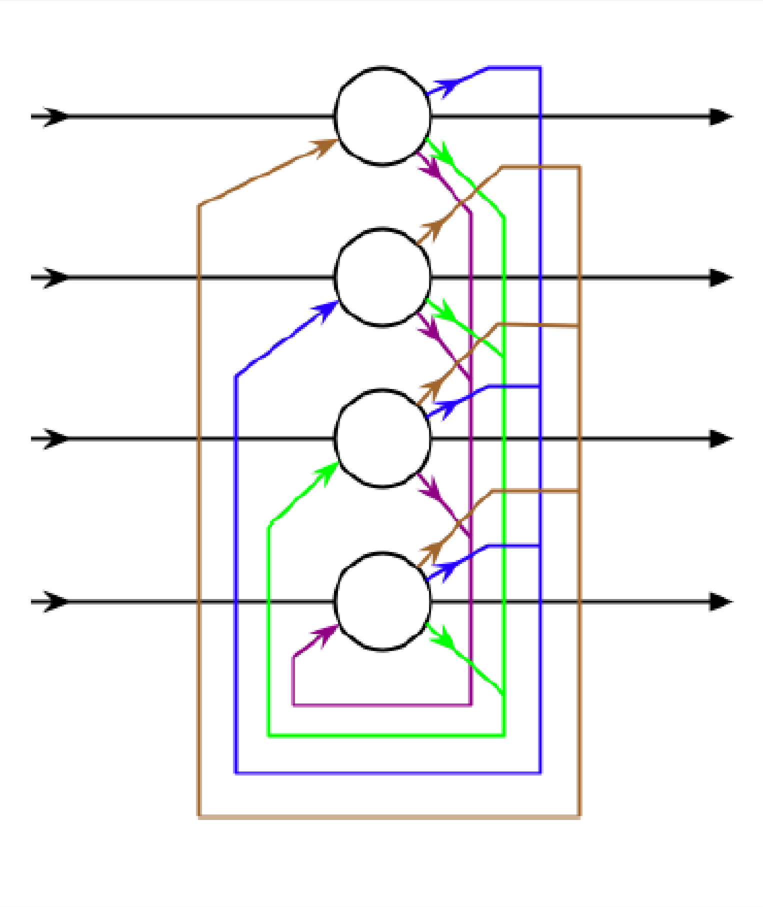
- Several theorems that FNN approximates any function
- FNN allows decomposition to apply chain rule for gradient computation
- Widely used

Recurrent NN

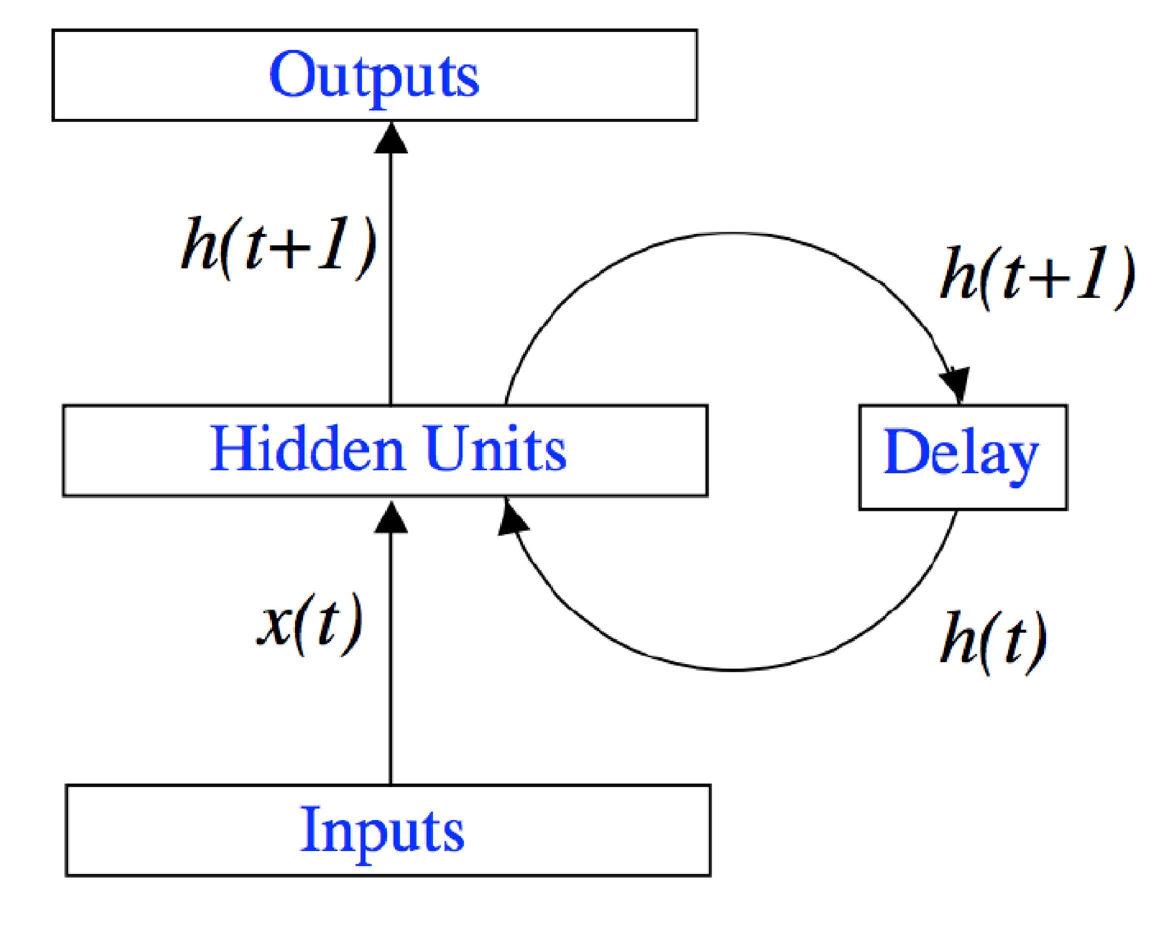
- Biological neural networks are recurrent
- RNN models a dynamic system
- Not as widespread and has a few conventional models / ways to learn them
- Any Turing machine can be represented as a fully connected RNN with sigmoid activation function (Siegelman and Sontag, 1991)

Hopfield NN

- Represent associative memory
- Networks show interesting behavior, they may become stable, oscillate or show deterministic chaotic behavior.

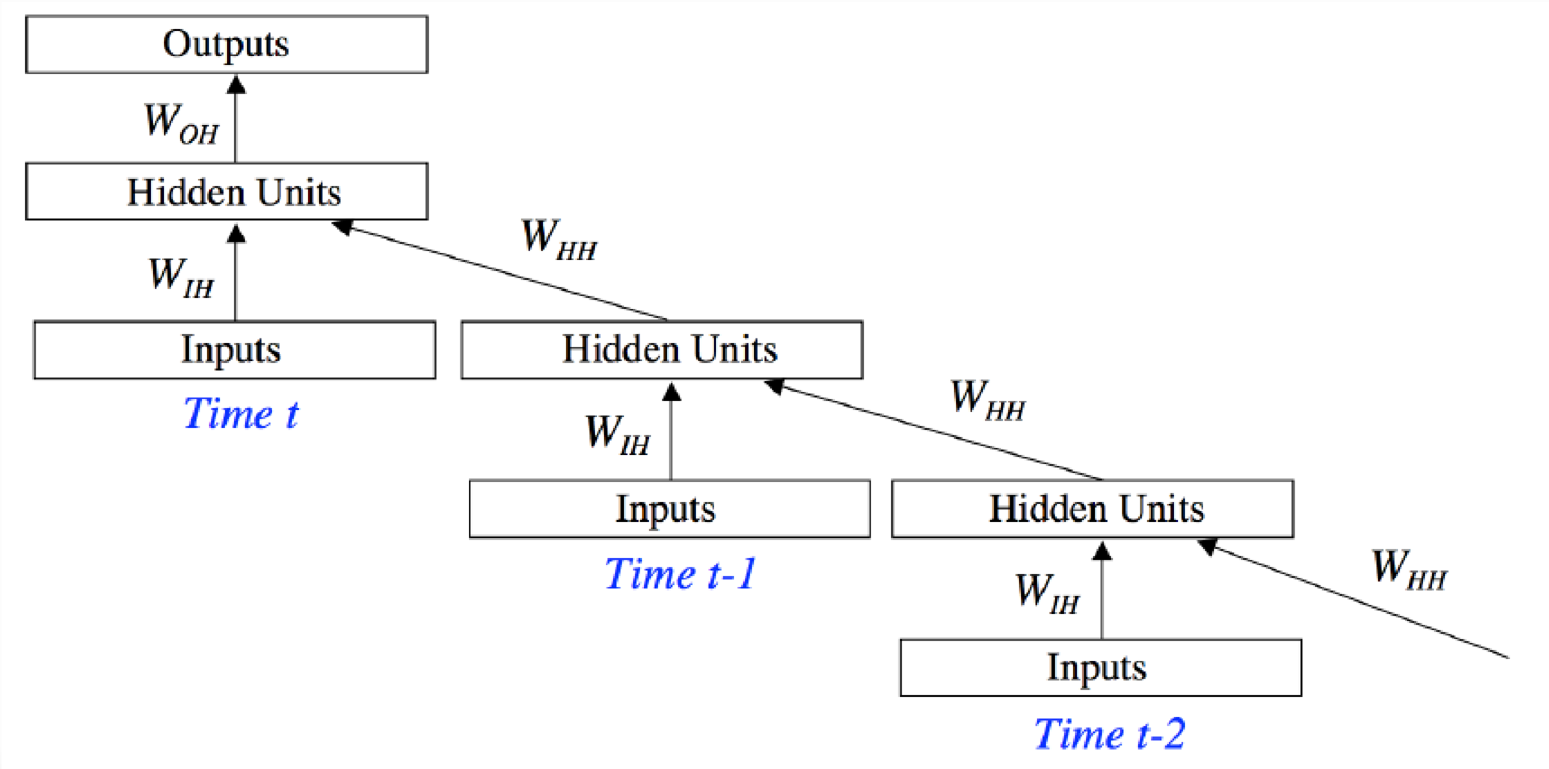


Backpropagation through time



Unfolded RNN

- Limiting the maximum length, we can perform backpropagation through time



Weights sharing

The problem is that the weights must remain the same

Backpropagation can be easily changed to obtain this

If we want $w_i = w_j$, then $w_i^{(0)} = w_j^{(0)}$ and $\Delta w_i^{(k)} = \Delta w_j^{(k)}$ $\forall k$ should be satisfied.

$$\Delta w_i^{(k)} = \Delta w_j^{(k)} := \frac{\delta L}{\delta w_i} + \frac{\delta L}{\delta w_j}$$

RNN analysis

Advantages

- Can represent not just functions, but systems
- Is a part of deep learning evaluation framework

Disadvantages

- Requires a lot of time to be trained
- Vanishing / exploding gradient
- Always changes previous signal, which makes them forget things

Lecture plan

- Sequences and time series
- Recurrent neural networks
- RNNs with memory
- More connections
- Word representation

Long term and short term memory

Long-term memory represented as a vector, containing slowly changing information about what we have learned previously

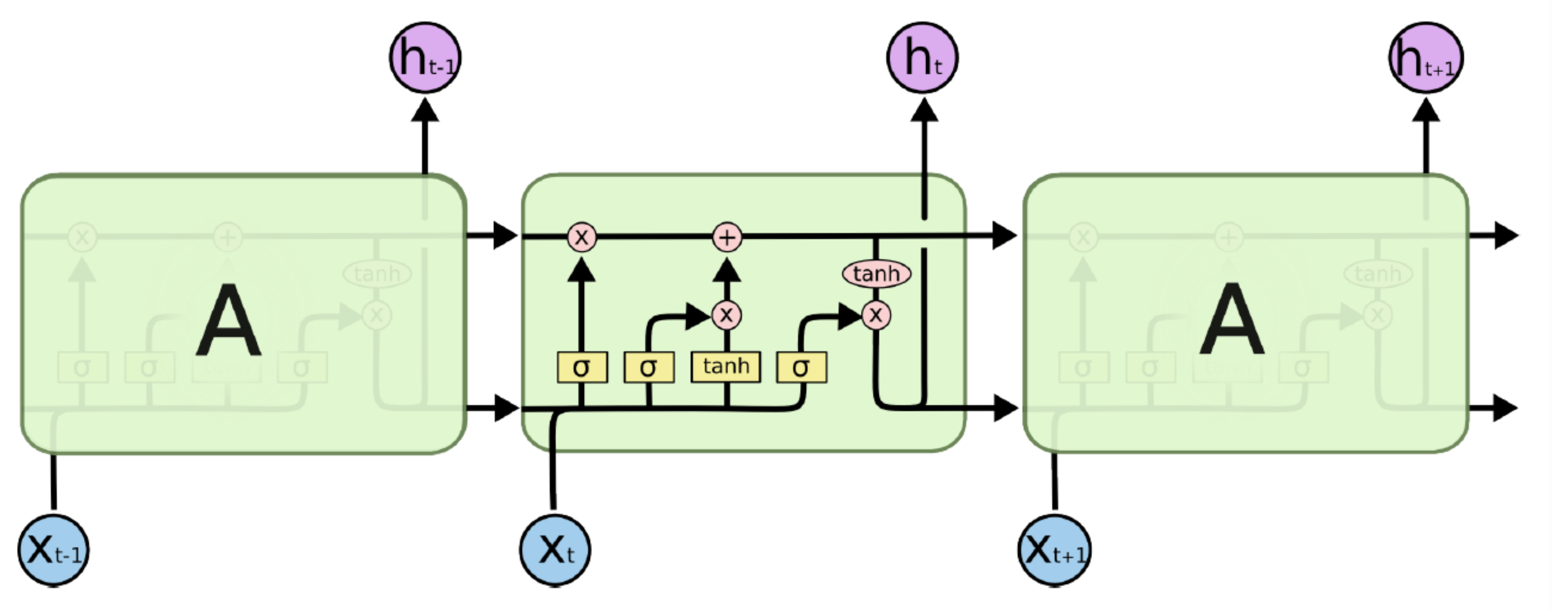
Short-term memory is with hidden states

Long-short term memory (LSTM) combines these two

LSTM

Memory unit uses LM input and entry to process them in SM

Connections between memory units are linear



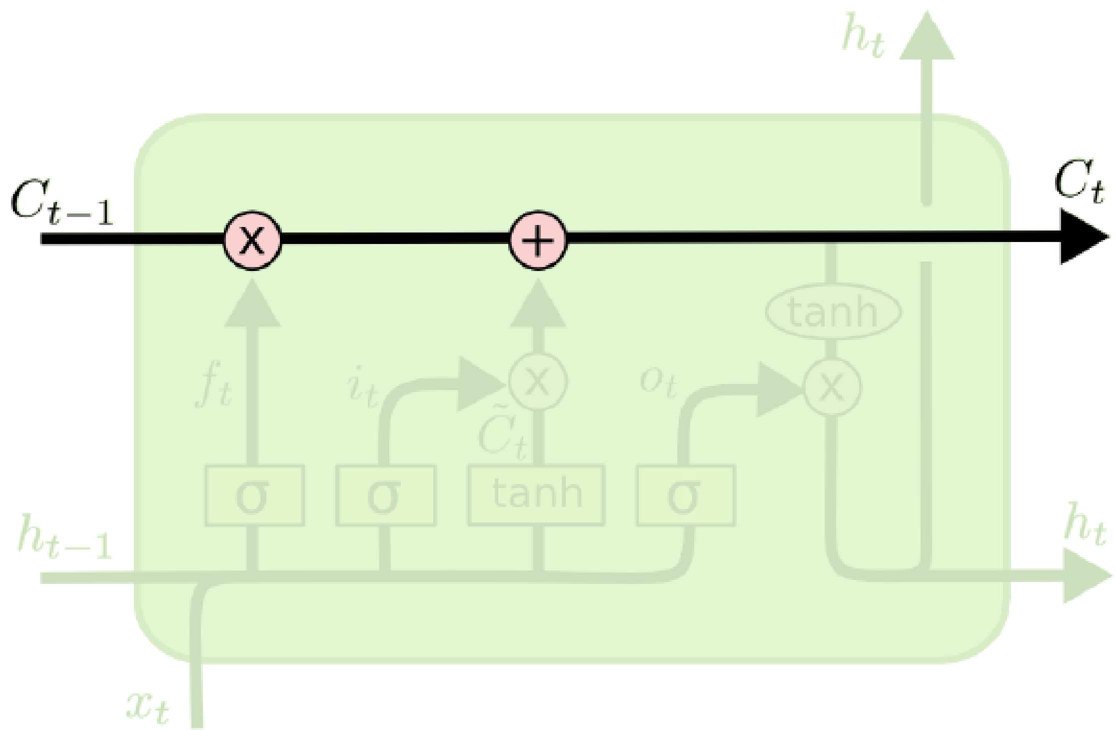
Memory unit

Memory unit uses LM input and entry to process them in SM

Connections between memory units are linear

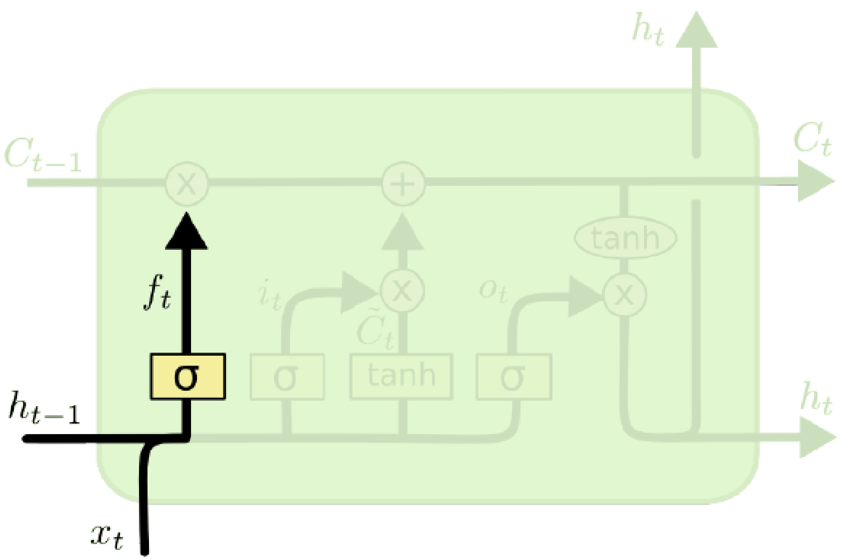
Conveyor belt

Stores long term memory



Forget layer

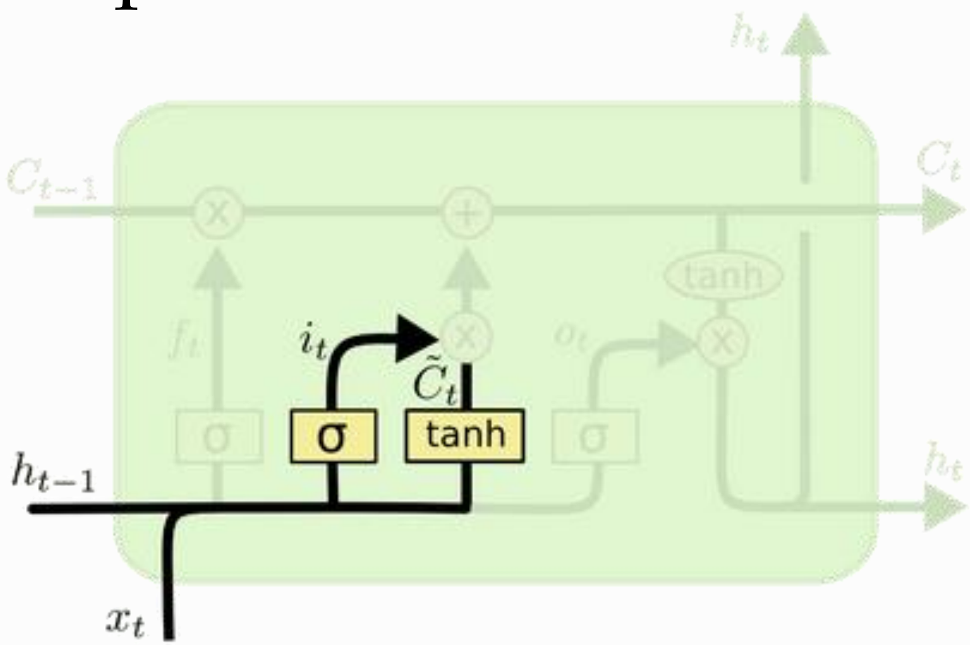
Forget layer multiplies some values in LM to erase information from it



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

Input layer

Input layer decides which values we'll update

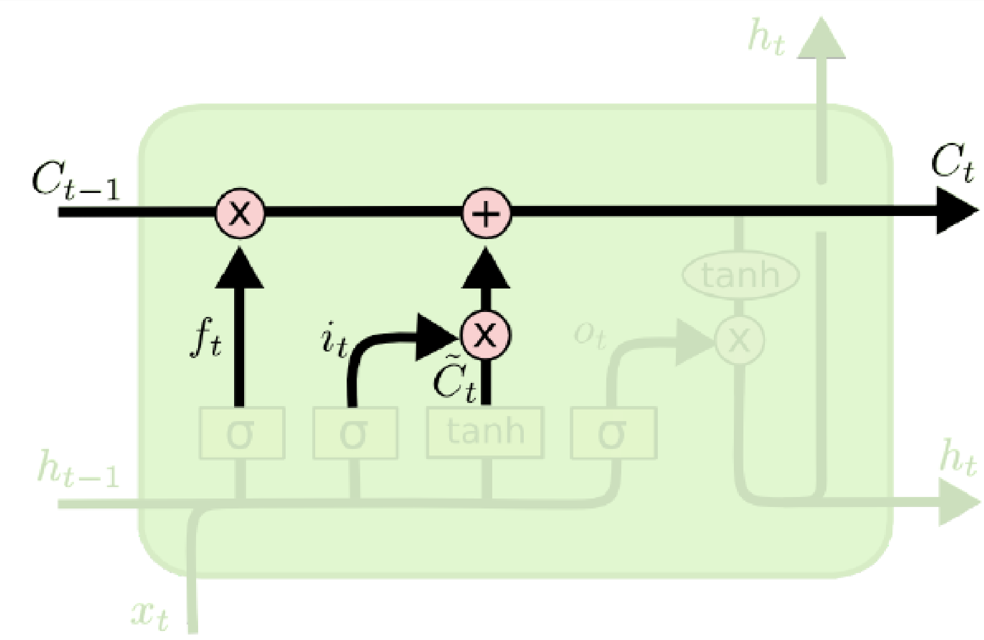


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

Long term memory updating

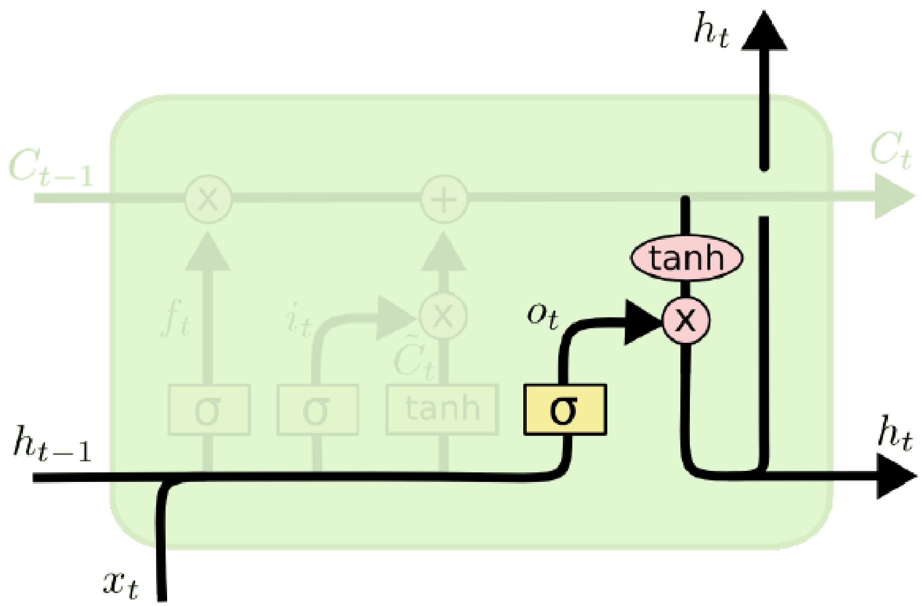
After evaluations, we update LTM



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

Hidden state updating

and hidden state

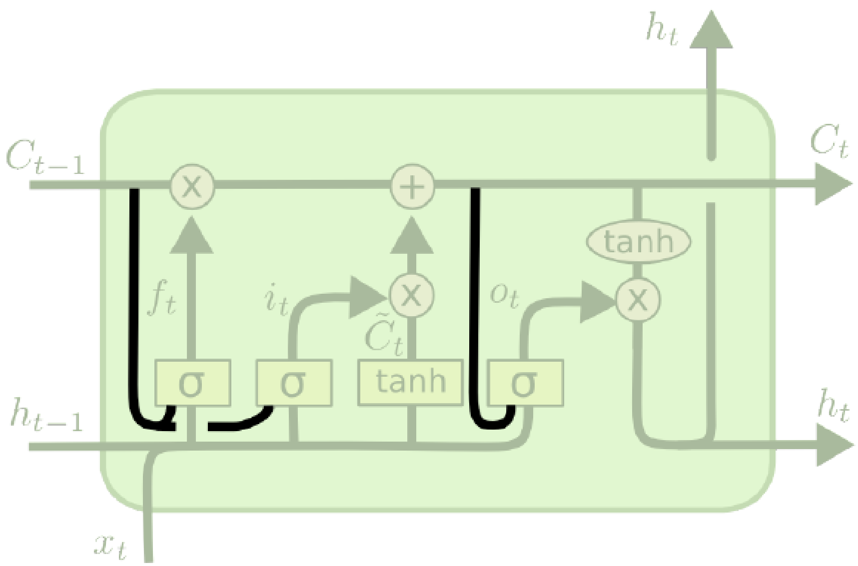


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t \text{ x } \tanh\left(C_t\right)$$

LSTM variation: peephole connections

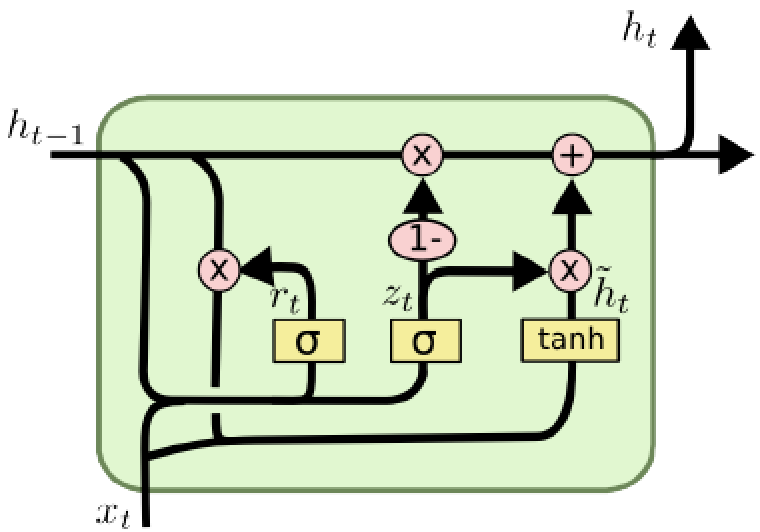
LTM vector can be used to update itself and the hidden state



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

Gated restricted unit

We can reduce the number of parameters in unit by rearranging operations and storing everything just with *h*



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

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

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

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

Memory unit analysis

Advantages

- Changes signal
- No vanishing gradient problem

Disadvantages

- Requires a lot of time to be trained
- Still forgets with exponentially growing chances of forgetting

Lecture plan

- Sequences and time series
- Recurrent neural networks
- RNNs with memory
- More connections
- Word representation

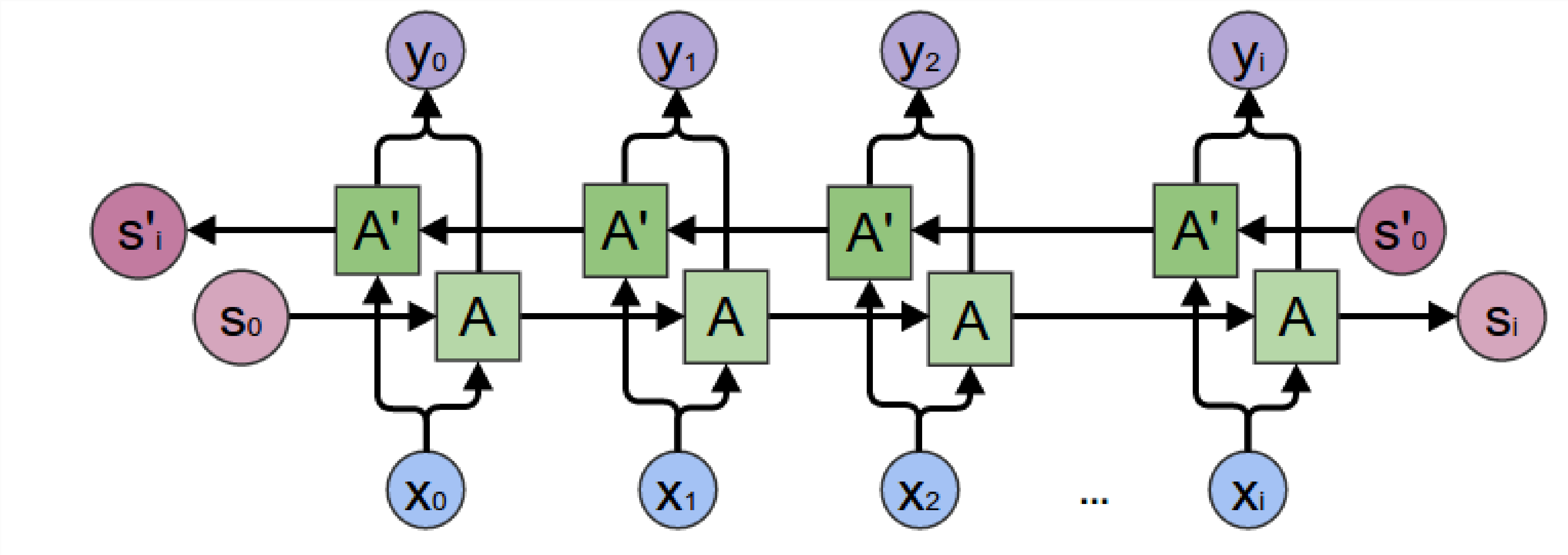
Main idea of adding reverse direction

Not only previous information is useful for understanding current signal

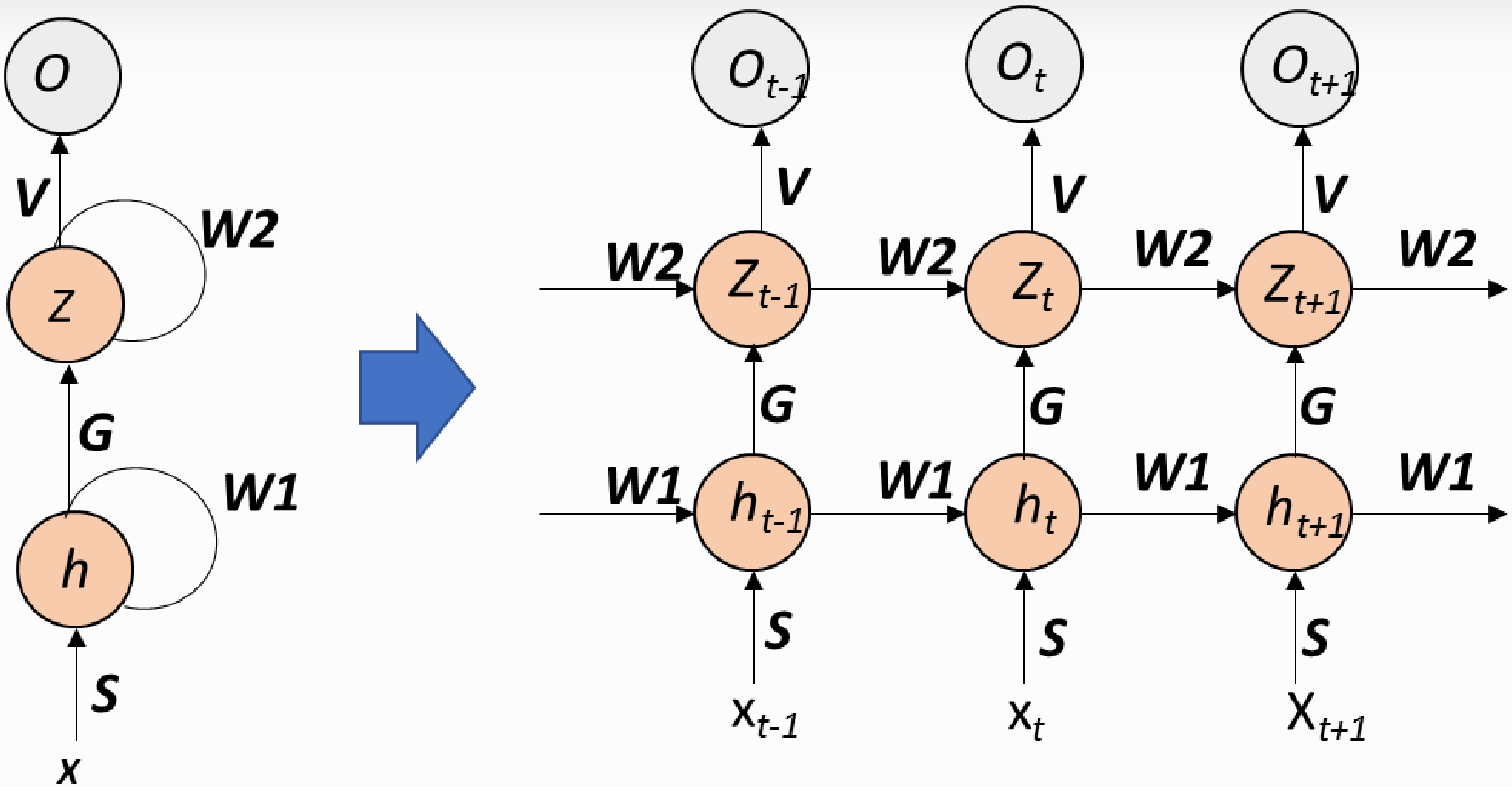
He said "Teddy bears are on sail!"

He said "Teddy Roosevelt was a great President!"

Bidirectional RNN



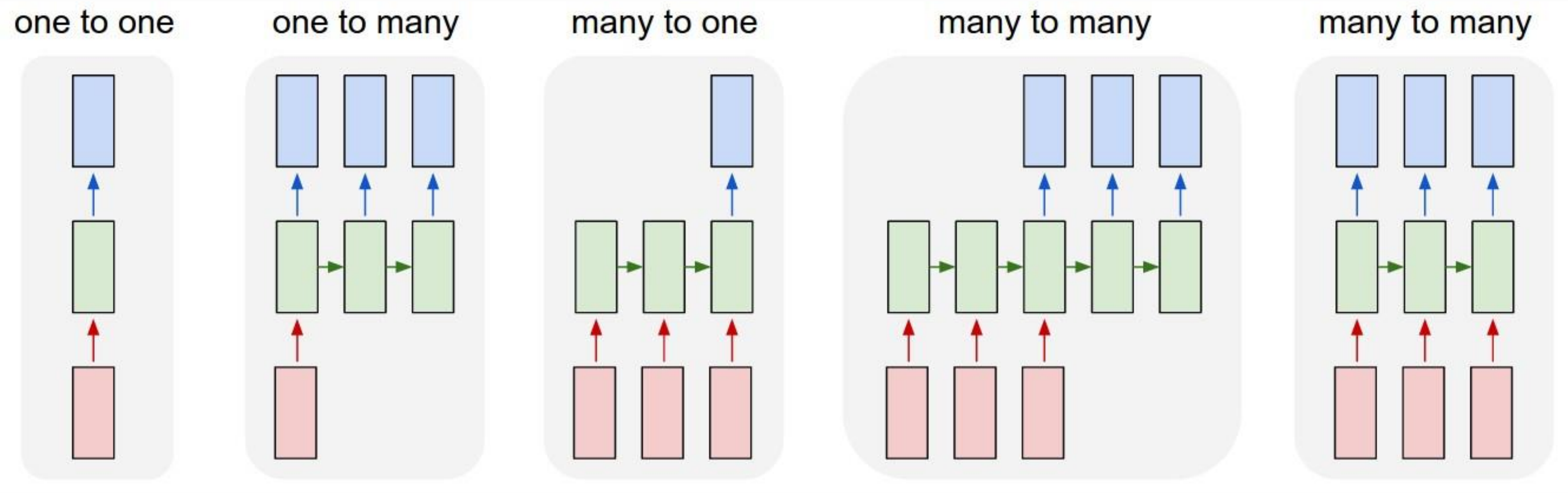
Deep RNN



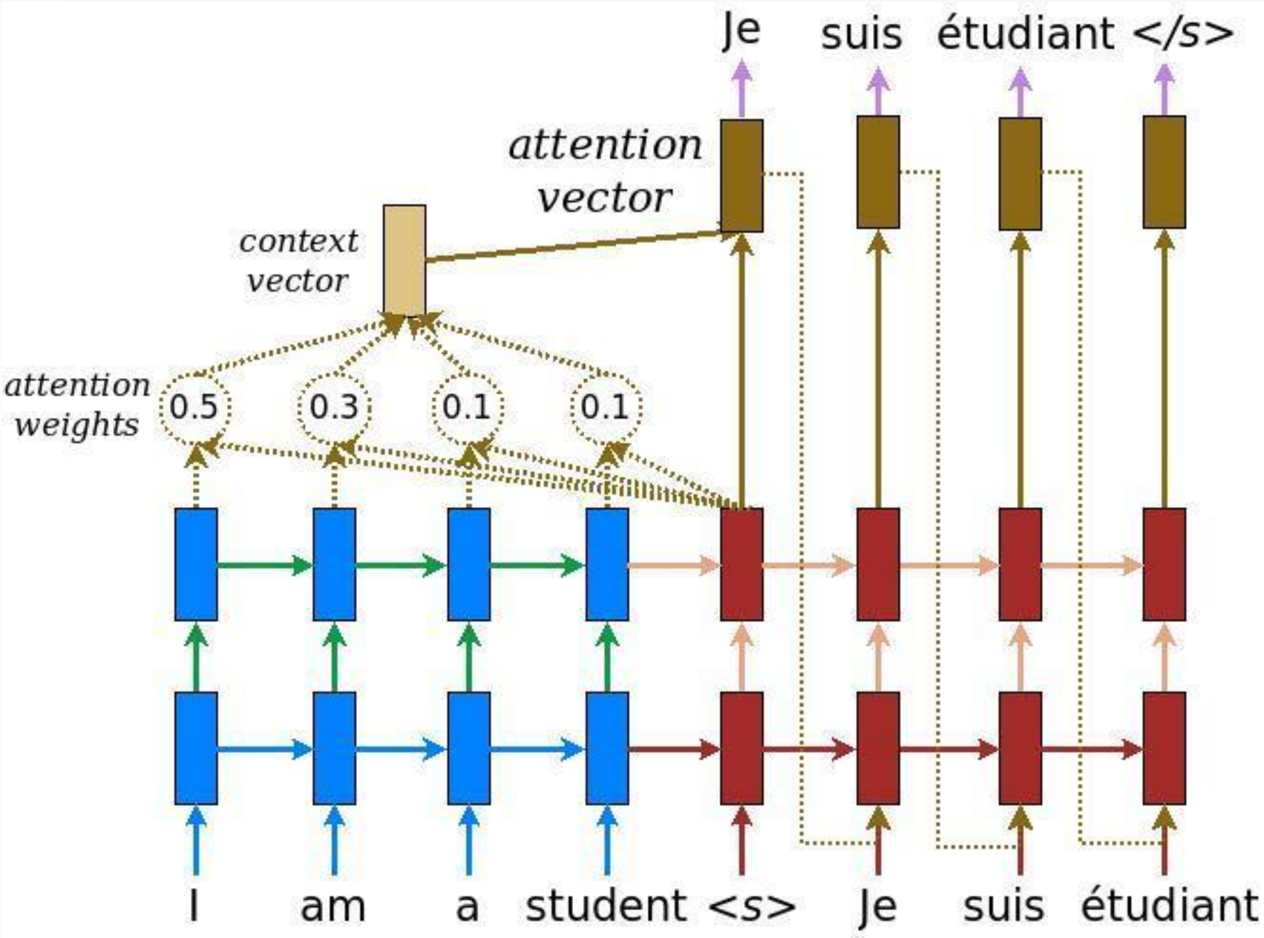
a) 2-layer Recurrent Neural Network (RNN)

b) Unfolded 2-layer Recurrent Neural Network (RNN)

Classification of RNNs



Attention mechanism



Attention weights

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

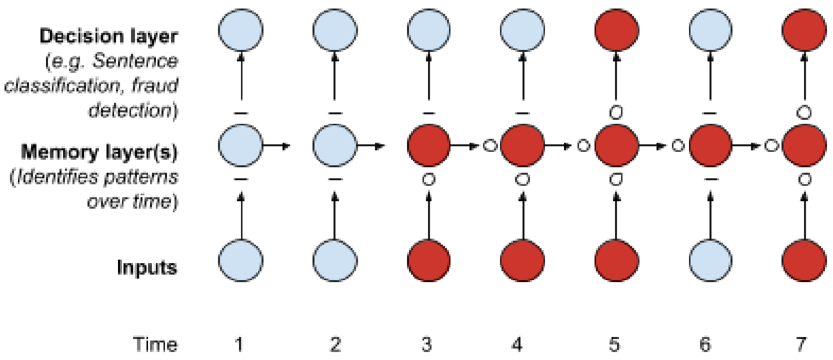
Context vector

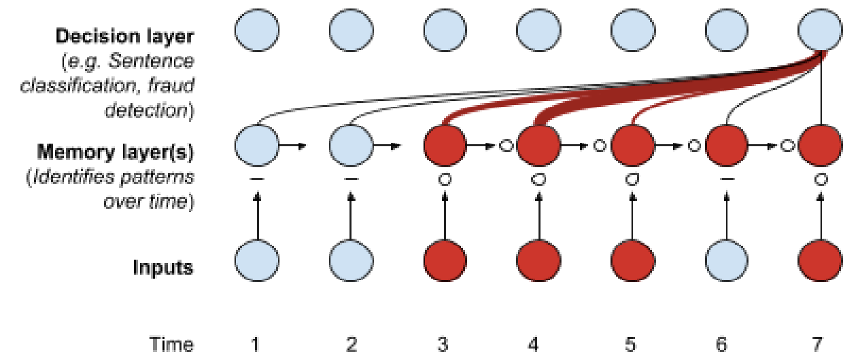
$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s$$

Attention vector

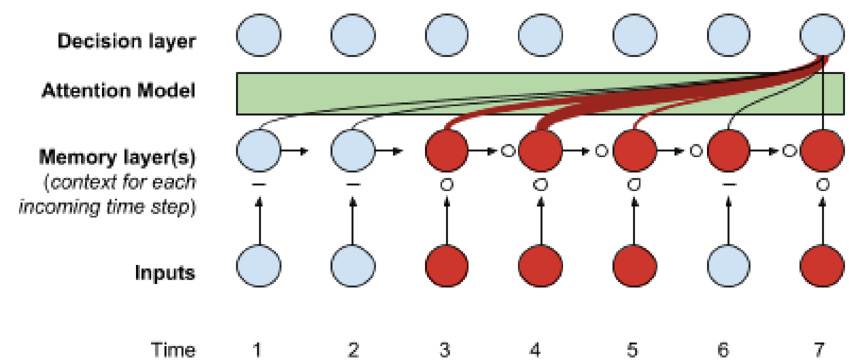
$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

Attention





Attention is a vertor of weights representing importance of all past inputs for current output



Attention discussed

- Attention vector is learned just as all other parameters
- Allows to proceed very long sequences
- Can be easily integrated in any deep neural network
- Has close relation with memory and memory networks

Lecture plan

- Sequences and time series
- Recurrent neural networks
- RNNs with memory
- More connections
- Word representation

The text representation question

Networks works with vectors.

How to vectorize text?

One-hot encoding

- Fix a vocabulary of size $|V|$
- Enumerate words with $i(w)$
- Each word w is represented as a vector $\left(0_1, \dots, 0_{i(w)-1}, 1_{i(w)}, 0_{i(w)+1}, \dots, 0_{|V|}\right)$

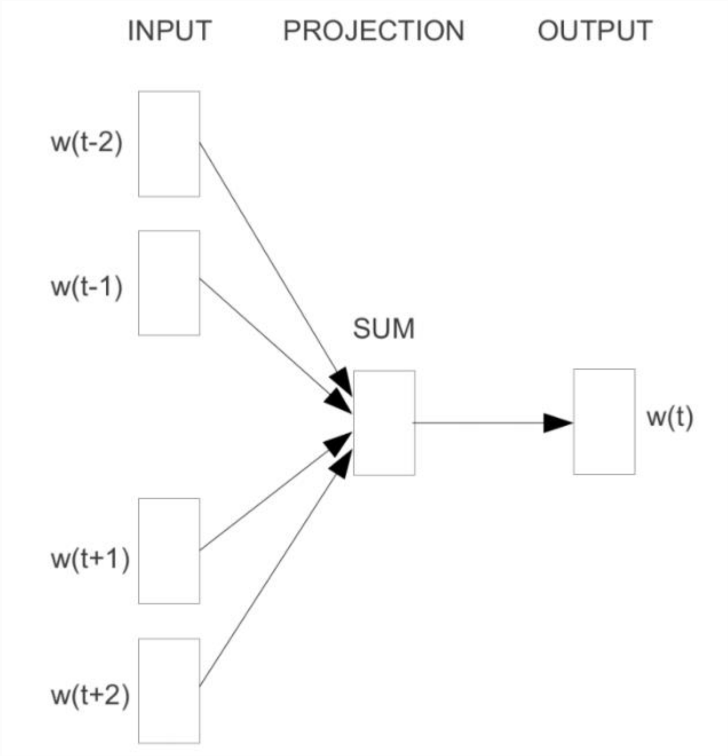
Main idea of Word2Vec

Distributional hypothesis (Harris, 1954): words that occur in the same contexts tend to have similar meanings

Main idea: characterize words with its context by learning such representations

Continuous bag of words

Predict word given its context

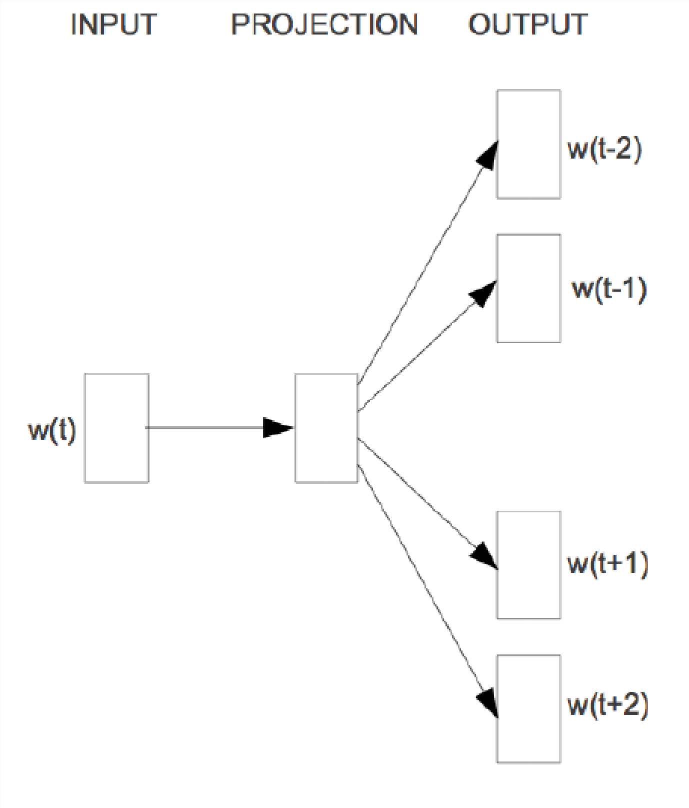


Loss function is

$-\log \Pr(w_i | \text{context}(w_i))$

Skip-gram

Predict context given word

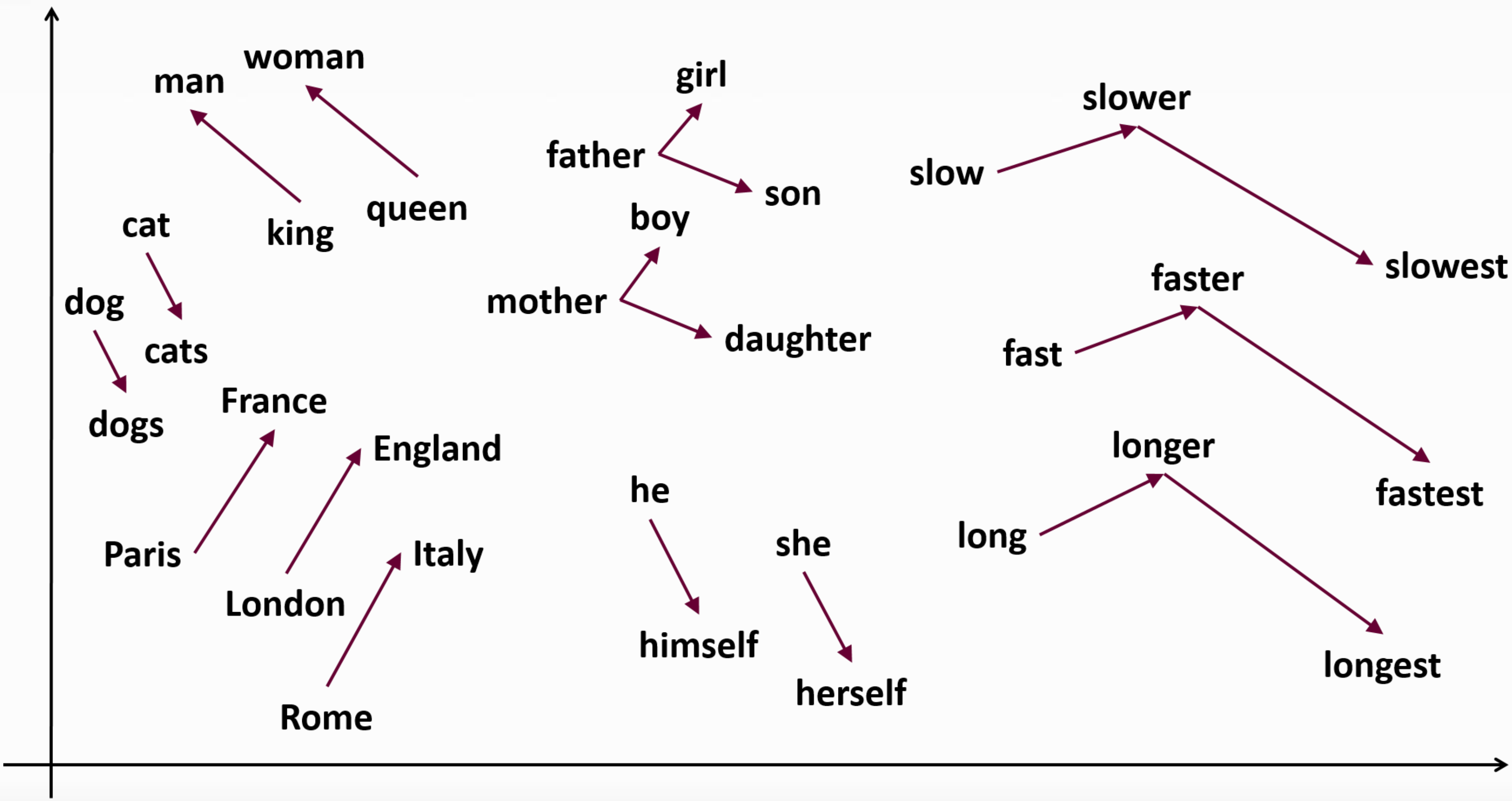


Loss function is
$-\log \Pr(\text{context}(w_i)|w_i)$

How to train?

- Show pairs of word in context, say $(w_i | \text{context}_j(w_i))$
- Subsample from the previous set deleting frequent words more frequently
- Negative sampling for creating negative samples

Word2Vec properties



State-of-the-art embeddings

- BERT
- ELMO
- FastText
- Skip-Thoughts
- Quick-Thoughts
- InferSent (for machine translation)
- RusVectōrēs (for Russian)