Recurrent Neural Networks (RNNs)

Zoltan Miklos

University of Rennes 1

2022

Sequential data

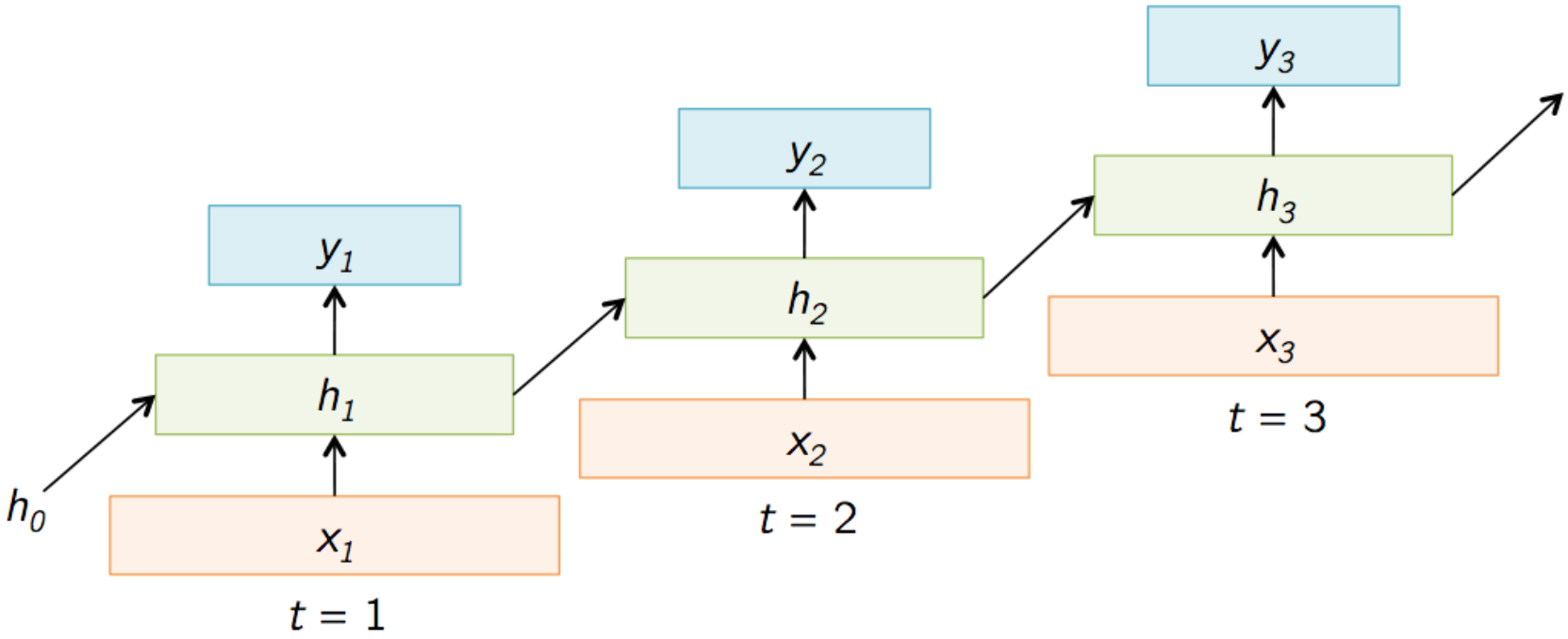
The specific structure of sequential data requires special methods

- Time series
- text in natural language
- bioinformatics (DNA, protein sequences, etc.)
- ...

Why not a feed-forward network for sequential data ?

- Input and output lengths can be different (for each training example)
- (more serious problem) if we learn some features at the beginning of the text (as we process the sequence of letters or words), we would like to use these features later
- if we have a long text (sequence), it is not easy to set a fixed length input (words far apart in the text might be strongly connected)

"Vanilla" RNN



http://slazebni.cs.illinois.edu/spring17/lec02_rnn.pdf

RNN Schema

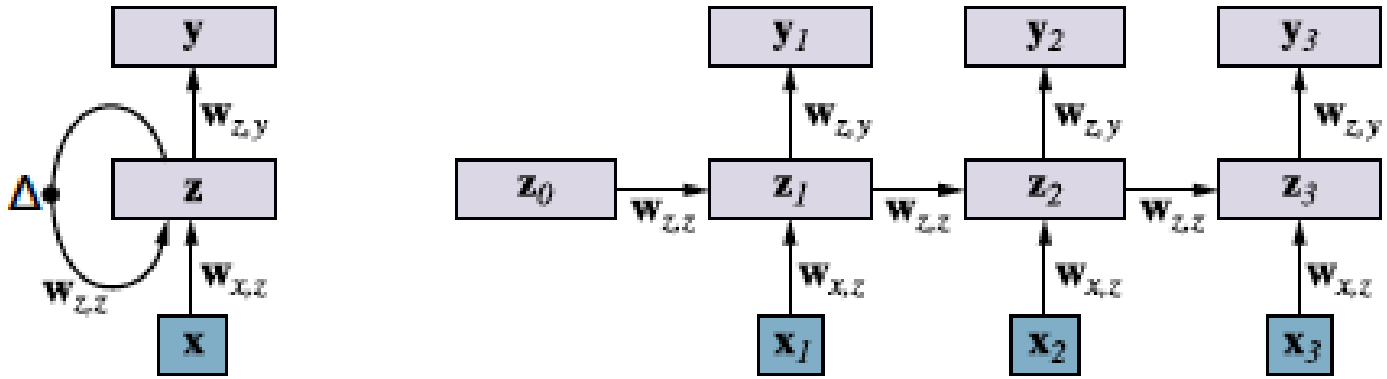


Figure 22.8 (a) Schematic diagram of a basic RNN where the hidden layer z has recurrent connections; the Δ symbol indicates a delay. (b) The same network unrolled over three time steps to create a feedforward network. Note that the weights are shared across all time steps.

Recurrent Neural Networks (RNN)

The simplest version of RNN ("vanilla" RNN) :

- When processing the input, at each time step $\langle t \rangle$, the network passes the activation function $h^{\langle t \rangle}$ to the next time step $\langle t+1 \rangle$
- In other words, at time $\langle t+1 \rangle$, it will consider the input word $x^{\langle t+1 \rangle}$ and the activation function $h^{\langle t \rangle}$ as an input
- the weights are shared over time

Markov assumption

- The hidden state at moment t is sufficient to describe all the information that we observed before this moment
- The hidden state at $t+1$ will only depend on the hidden state at t and the observed input
- this assumption may be invalid for a number of sequence related problems! Nevertheless this assumption simplifies the computations and the resulting networks are widely applicable
- (recall the naive Bayes assumption)

Vanishing gradients problem

The "vanishing gradient" is the following problem :

- Gradient descent methods are iterative techniques, that update the parameter values, proportional to the partial derivatives of the cost function
- These partial derivatives could be very small (very close to 0)
- Because of the chain rule (i.e. multiplication), some close to 0 derivatives lead to an estimation of close to 0 also for other parameters
- The gradient descent iteration could stop as the gradients do not change (they vanish)
- In a deep network, if we have a lot of small partial derivative, they will also prevent the training in the early layers

This problem arises for feed-forward networks as well but it is even more important for recurrent neural networks.

Basic RNN computation

Notation :

- $\mathbf{w}_{z,z}$ weights that describe how a state influences the next one
- $\mathbf{w}_{z,y}$ weights that describe how a state influences output at moment t
- $\mathbf{w}_{x,z}$ weights that describe how an input symbol influences the state
- $\mathbf{g}_z$ activation function at hidden layers, activation function at the output $\mathbf{g}_y$

RNN computation :

- $\mathbf{z}_t = f_{\mathbf{w}}(\mathbf{z}_{t-1}, \mathbf{x}_t) = \mathbf{g}_z(\mathbf{W}_{z,z}\mathbf{z}_{t-1} + \mathbf{W}_{x,z}\mathbf{x}_t)$
- $\hat{\mathbf{y}}_t = \mathbf{g}_y(\mathbf{W}_{z,y}\mathbf{z}_t)$

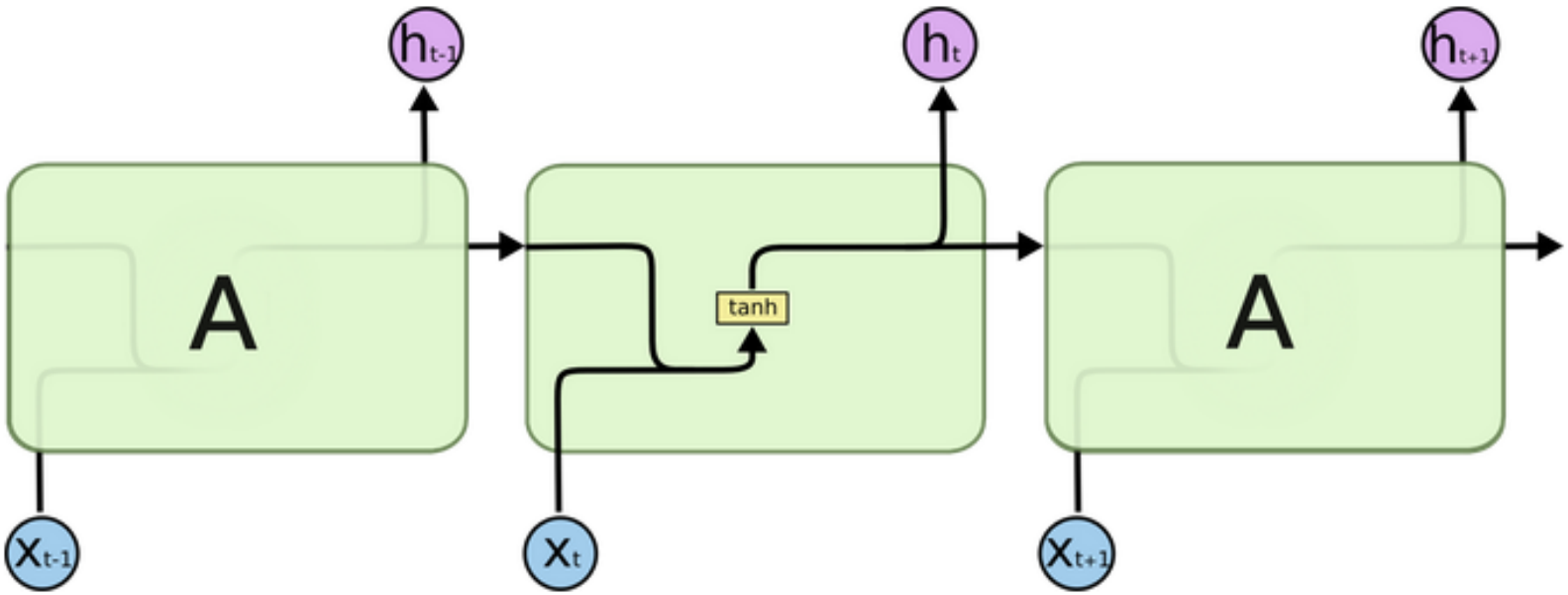
Training RNN

- How to learn the weights : gradient descent, with Backpropagation through time
- It we compute the gradient at t, it will depend on the gradient at $t-1$, i.e. the gradient computation is recursive
- This simple version of recurrent networks suffers from problems : vanishing gradient, or exploding gradient
- Need for a more sophisticated structure

Long Short Term Memory (LSTM)

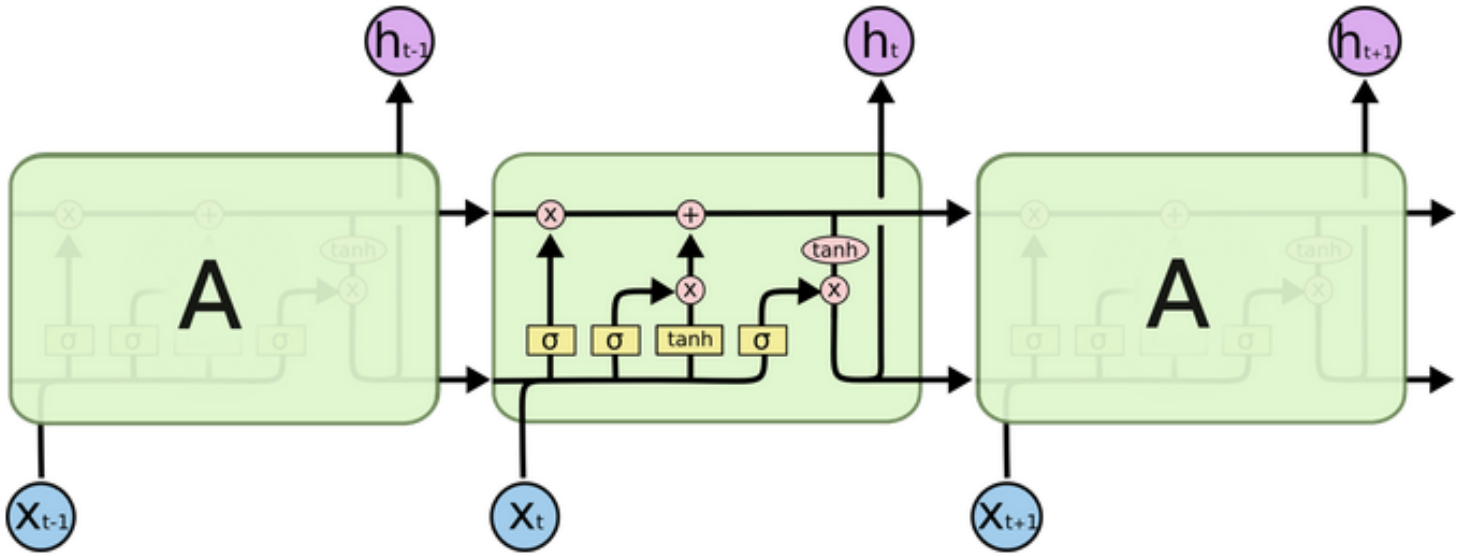
- A special type RNN, designed to avoid the vanishing gradient problem
- Capable of learning long-term dependencies
- Proposed by Hochreiter and Schmiedhuber (1997)
- Widely used in ingustry, for example Google : speach reconginition on the smatphone, Google translate, Apple : Siri, Amazon Alexa, etc.

Standard RNN



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

LSTM

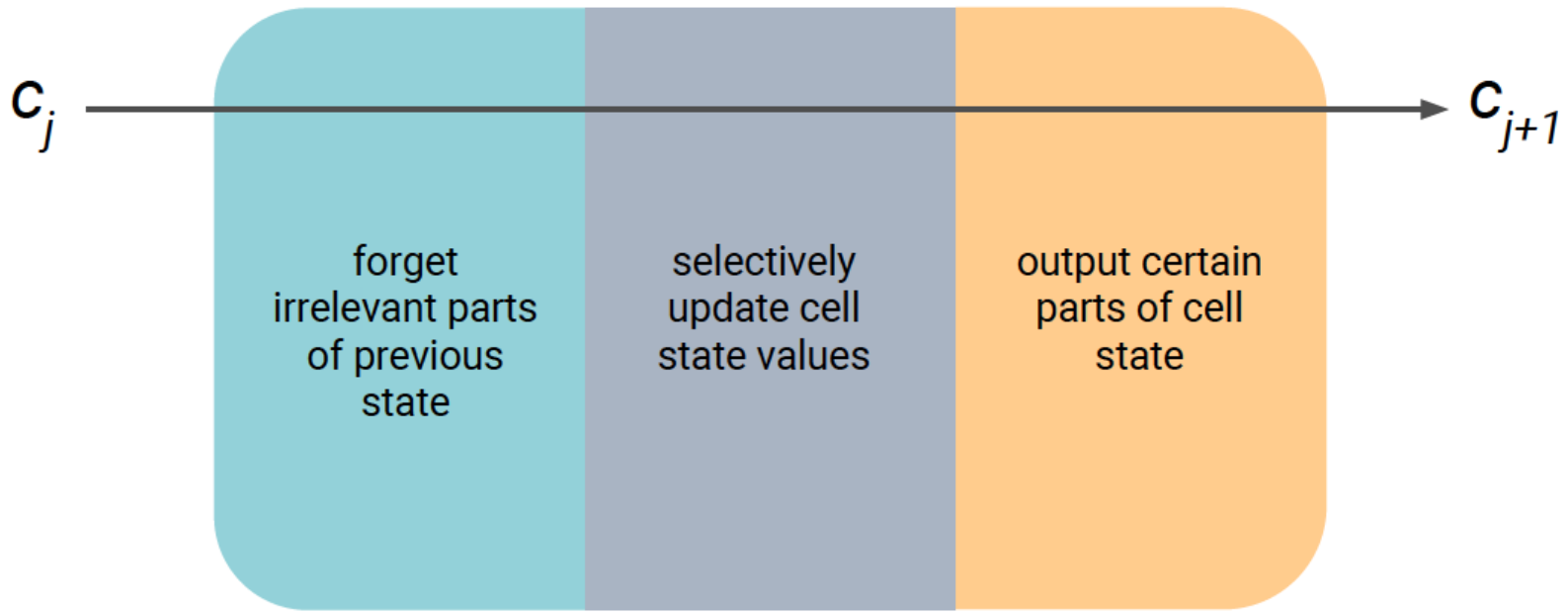


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)

- A state of the cell. LSTM regulates which information should be added/removed from the state, using gates.
- Input gate : forget irrelevant parts of the previous state
- Output gate : output certain parts of cell state
- Forget gate : selectively update cell state values

LSTM



http://introductiontodeeplearning.com

LSTM update equations

- $\mathbf{f}_t = \sigma(\mathbf{W}_{x,f}\mathbf{x}_t + \mathbf{W}_{z,f}\mathbf{z}_{t-1})$ (forget)
- $\mathbf{i}_t = \sigma(\mathbf{W}_{x,i}\mathbf{x}_t + \mathbf{W}_{z,i}\mathbf{z}_{t-1})$ (input)
- $\mathbf{o}_t = \sigma(\mathbf{W}_{x,o}\mathbf{x}_t + \mathbf{W}_{z,o}\mathbf{z}_{t-1})$ (output)
- $\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{i}_t \odot tanh(\mathbf{W}_{x,c}\mathbf{x}_t + \mathbf{W}_{z,c}\mathbf{z}_{t-1})$
- $\mathbf{z}_t = tanh(\mathbf{c}_t) \odot \mathbf{o}_t$

where $\odot$ denotes pointwise multiplication of vectors

Gated Recurrent Unit (GRU)

- An alternative to LSTM, it uses a specific gate technique
- Can also eliminate the vanishing gradients problem
- Efficient technique for sequence to sequence learning