Recurrent Neural Networks

Advanced Machine Learning and Artificial Intelligence

Yuri Balasanov

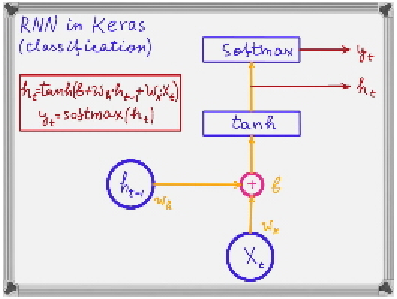
University of Chicago, MScA

Outline of the Session

- Recurrent neuron
- Unrolling recurrent neuron
- Memory cell, input-output relationships
- Motivation for LSTM
- LSTM structure

Text: Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Technologies to Build Intelligent Systems, Aurelien Geron, 2017
Text: Christopher Olah's blog and pictures in it:
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

- **Recurrent neural networks (RNN)** are a family of networks for representation learning about sequential data, like text or stream of sensor data
- If CNN applies convolution of the kernel across the the input feature map, RNN creates output depending not only the current input feature map, but also of the previous output called **hidden state**

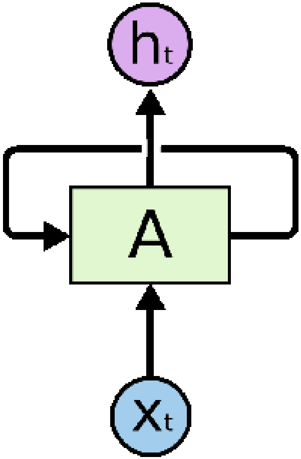


RNN in Keras
(classification)
$h_t = \tanh(b + W_h \cdot h_{t-1} + W_x \cdot X_t)$
$y_t = \text{softmax}(h_t)$
softmax
tanh
y_t
h_t
h_{t-1}
W_h
b
W_x
X_t

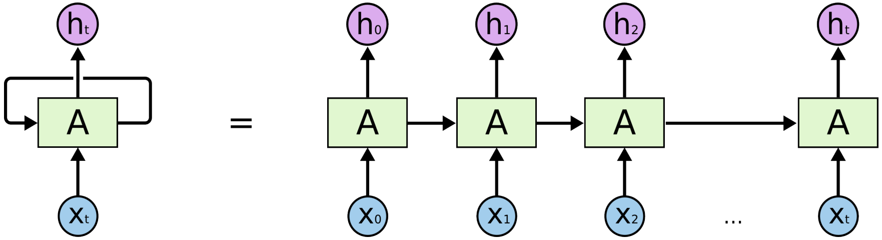
Recurrent Neuron

- Recurrent neuron looks like usual feedforward neuron, but it has recurrent collateral or feedback
- At each time step t (**frame**) recurrent neuron receives input x_t and its previous output h_t
- Such neuron has 2 sets of weights

$$h_t = \phi\left(w_x^T \cdot x_t + w_h^T \cdot h_t + b\right)$$



Unrolling Recurrent Neuron Through Time

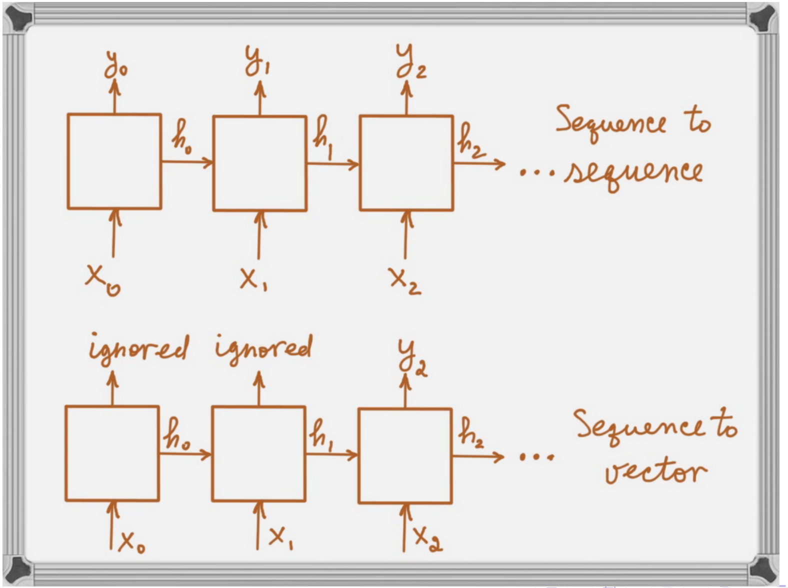


- Recurrent neuron is like multiple copies of the same neuron, which pass information to each other
- Such chain of copies transforms input ordered sequence into output ordered sequence

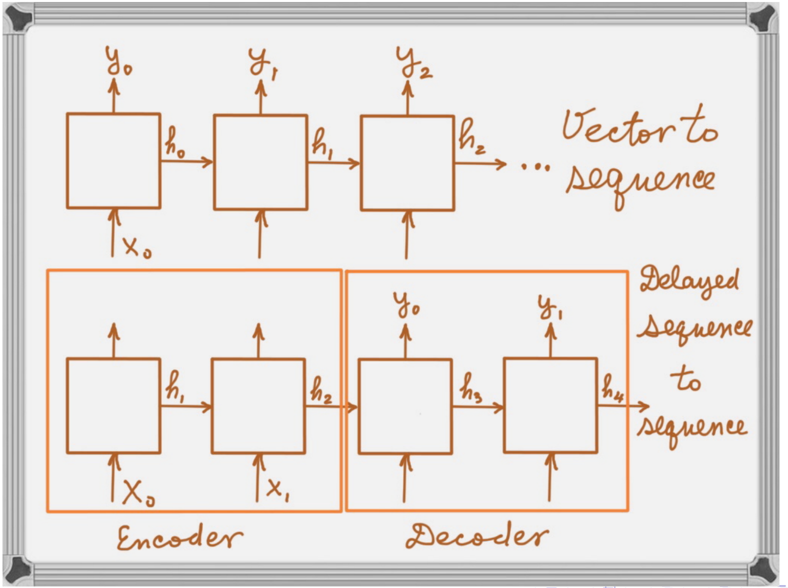
Memory Cell and Input-Output Relationships

- To enable a network remembering information over multiple steps there is a part of neural network called memory cell
- At each time t cell has a state $h_t = f\left(h_{t-1}, x_t\right)$
- State may or may not be different from output $y_t = y\left(h_{t-1}, x_t\right)$ of the cell
- An RNN can:
  1. Take a sequence of inputs and produce a sequence of outputs: useful for analysis of time series
  2. Take sequence of inputs and create a vector of outputs: input is a document, output is a vector of sentiment scores
  3. Take input as one vector and create output as a sequence of components: single input of an image results in a sequence of features
  4. Take a sequence and produce a vector (sequence to vector, or encoder) and follow it by a vector to sequence network (decoder): feed a sentense in one language, encode it into a vector representation, then decoder will convert this vector into a sequence of words in another language

Input-Output I



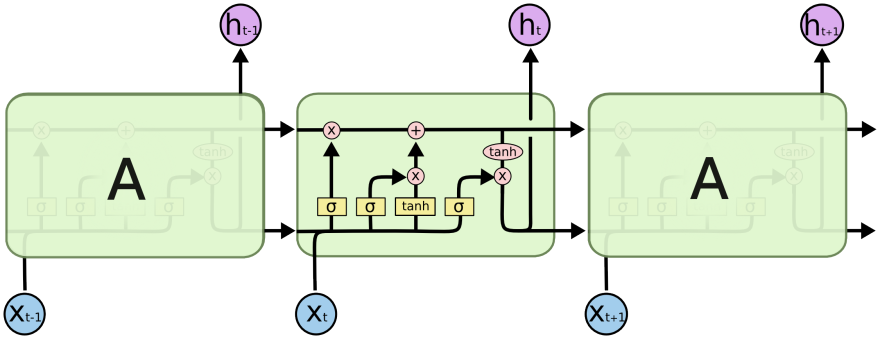
Input-Output II



Motivation for LSTM

- Recurrent neurons can be placed in layers capable to process sequences of large bandwidth and account for information in the previous step of ordered vectorized data
- In order to use information of several lags of inputs it is necessary to connect each neuron's output not only to its own input, but also to the inputs of previous layers
- Unfortunately, as the number of lags starts growing ability of recurrent network to learn goes down
- Dependencies lagged by multiple steps require **long term memory** in recurrent network. Training recurrent networks with long term memory runs into several fundamental problems
- Solution to such problems can be found in Long Short Term Memory networks (**LSTM**s)

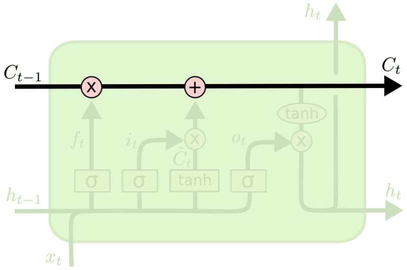
LTSM Structure I



- Inside LTSM cell there are several interacting layers containing activation functions (sigmoid σ or tanh), element-by element operation (encircled "$+$" or "$\times$"), vector transformations (arrows), concatenations (merging arrows) and copying (arrowed forks)

LSTM Structure II

Cell state

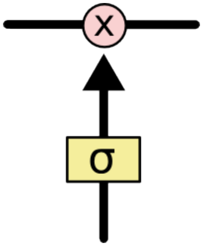


- The cell state runs through the entire LTSM with some possibilities of sequential transformations
- Transformations may be removing or adding some information to the state

LTSM Structure III

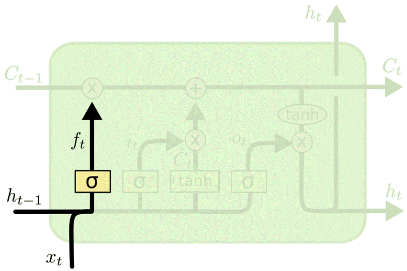
Gates

- Transformations of the state information are done through the gates
- Sigmoid multiplied by the data vector decides what part gets through
- There are 3 gates: input, forget and output



LSTM Structure IV

Forget gate layer

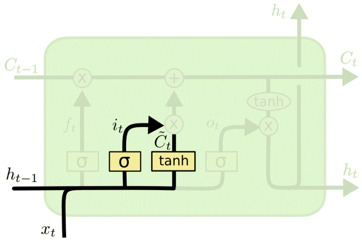


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

- First gate: **Forget gate layer**: returns an number between 0 and 1 for each value in the state vector
- If new information x_t contradicts the old information (new name of a subject) the old information needs to be forgotten

LSTM Structure V

Input gate layer

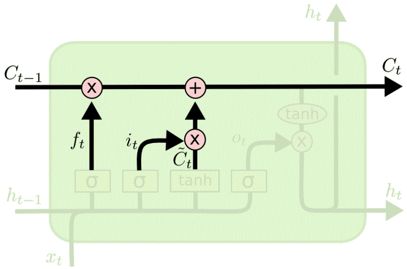


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- New information is added to the state through **Input gate layer**: a two-step procedure of deciding which new information should stay in the state
- Sigmoid layer decides which values to update, tanh activator creates a vector of new values

LSTM Structure VI

Updating state

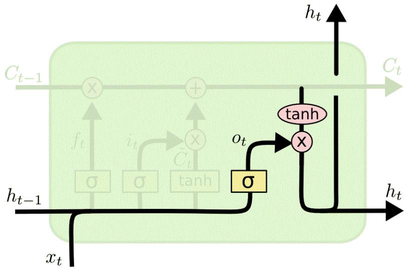


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Updating the old state information C_{t-1} to C_t is done by multiplying the old state by f_t (forgetting), then adding the result to $i_t \times C_t$, after that the result is sent straight out through the **Output gates layer**

LSTM Structure VII

Output gate layer



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

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

- Output is a filtered cell state:
  1. First, sigmoid layer decides what components of the cell state to output
  2. Then, cell state is passed through tanh to make the values between -1 and 1, and multiplied by the output of the sigmoid gate telling which parts to let out