Universität des Saarlandes
Prof. Dr. Dietrich Klakow
Lehrstuhl für Signalverarbeitung
NNIA Winter Term 2019/2020

Exercise Sheet 10

Philip Georgis [s8phgeor], Pauline Sander [s8pasand], Vilém Zouhar [vizo00001]

(Solutions)

Deadline: 02.02.2021

Exercises

Exercise 10.1 - Architecture (0.5 + 1.5 = 2 points)

a) What is the benefit of an LSTM over RNN? Give a short explanation (2-4 sentences).

b) Draw an LSTM cell and provide the formulas used to calculate each element. Explain the function of each element.

Solution 10.1

a) LSTM contains explicit input, output and forget gates. It alleviates the vanishing gradient problem. Furthermore, an LSTM is better at handling long-distance dependencies as the memory state always contains some information on previous states.

b) Figure 1 shows an LSTM cell. It processes the input as follows:

1. $e_t = concat(x_t, c_{t-1})$ The LSTM cell takes an input x_t, the t^{th} element of the sequence, as well as the control state c_{t-1} of the previous LSTM cell (same as previous output) and concatenates them.
2. The LSTM cell also takes the memory state m_{t-1} of the previous LSTM cell.
3. $f_t = sigm(W_f \times e_t + b_f)$ contains values [0,1] and is based on e_t. The forget gate removes (reduces the intensity of) parts of the memory state through element wise multiplication with f_t: $g_t = f_t * m_{t-1}$
4. $h_t = sigm(W_h \times e_t + b_h)$ works in a way similar to g_t. $i_t = h_t * tanh(W_i \times e_t + b_i)$ is the information contained in e_t that should be added to the memory state.
5. $m_t = i_t + g_t$ is the new memory state.
6. Parts of this new memory become the new control state (removing information the same way as in step 4): $c_t = j_t * tanh(W_c \times e_t + b_c)$ with $j_t = sigm(W_j \times e_t + b_j)$
7. o_t is the output at position t, identical to c_t.
8. The new memory and control states m_t and c_t are passed on to the next cell.

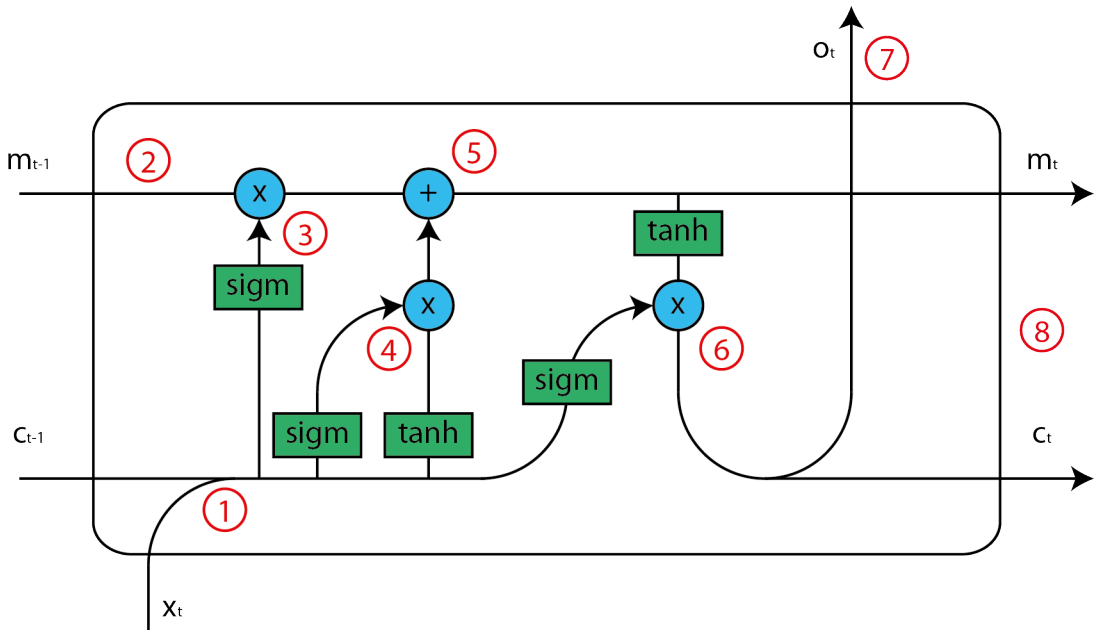

Figure 1: LSTM schema

Exercise 10.2 - Embeddings (1 + 0.5 + 0.5 = 2 points)

To perform the following exercise you have to read an article about static and contextualized embeddings. Answer the following questions based on the article:

a) Give short explanations of static and contextualized word embeddings. Provide examples.

b) What are the advantages of contextualized (dynamic) word embeddings over static ones?

c) What is transfer learning? On which task is a model (e.g. BERT) pre-trained? (**Hint:** read the article till the end) Can we pre-train without any task? Why / why not?

Solution 10.2

a) Static embeddings (e.g. word2vec) produce one vector given a single word or subword unit. This is an issue for homographs: *man* is going to have the same word embedding for both the noun and the verb.

Contextualized word embeddings (e.g. BERT) produce the vector representation given a single word and also the current context. The word *man* would have different embeddings in the sentences ***Man**, is it cold today?* and *Go **man** the post!* and *He is a **man** of few words.*

b) The issue with homographs is not the primary driver for contextualized word embeddings. The primary advantage of contextualized/dynamic word embeddings is that they better enable us to capture more meaning of the word. A disadvantage would be that the resulting model is much bigger.

c) In transfer learning the model is trained on one task, but then the task changes. The model can be slightly modified to suit the new task, e.g. by substituting the top layer so that the shape fits, etc. BERT is trained on masked language modeling

and next sentence prediction (technically *Does the second sentence follow from the first one?*).

For the pre-training we do still need some rudimentary task to be defined, so that the model can be optimized with respect to some defined loss.

Exercise 10.3 - Exam Preparation (6 points)

Start preparing for the exam:
Go over all the chapters and create the structure of the course. You can do it in form of a mind map, bullet points structure etc., whatever format works better for you.

Solution 10.3

Numbers corresponds to chapters according to the slides.

Ch. 1 Organization.

Ch. 2 Showcasing NN input architectures (image classification, ASR, MT); language model definition.

Ch. 3 Basic linear algebra concepts; Eigendecomposition; SVD; PCA

Ch. 4 The general task of ML; capacity; cross-validation; MLE; SVM (briefly); linear regression

Ch. 5 Feed-forward NN; gradient-based optimization (esp. gradient descent); learning rate; Jacobian ($\rightarrow \mathbb{R}^n$) & Hessian ($\rightarrow \mathbb{R}$, second order) matrix; Newton method (second order optimization); Taylor series; activation functions (ReLU, TanH, sigmoid); loss functions (MSE, cross-entropy)

Ch. 6 Chain-rule; computational graphs; multilayer perceptron; universal approximation

Ch. 7 Regularization (esp. norm penalties); data augmentation; multitask learning; early stopping; parameter sharing/tying; bagging/ensemble methods; dropout; adversarial training

Ch. 8 Optimization algorithms; stochastic/batch/minibatch GD; ill-conditioning; plateaus; local minima; cliffs and vanishing/exploding gradients; SGD, SGD with momentum, AdaGrad, RMSProp, Adam (+bias correction); parameter initialization; pretraining

Ch. 9 Mathematical convolution; 2D convolution; convolutional layer (receptive field, parameter sharing, multi-channel input); pooling (max, avg); hyperparameters (kernel size, padding and stride); structured output

Ch. 10 RNNs and their computation graphs; RNN training/inference; sequence modelling and connected tasks; biRNN; seq2seq (encoder-decoder); recursive NNs; long-term dependencies; LSTM; explicit memory; Transformer (RNN-less encoder, decoder still sequential); attention