Exercise H62.1: Long short-term memory (LSTM) (homework, 10 points)

In this exercise we will use an LSTM for classifying a simple number series. The goal is to detect whether the sum of all the elements in the series is greater or equal to 100 (=class 1) or less (=class 0).

(a) Create the train and validation data as follows:

- Draw 10000 different series, each consisting of 30 integer numbers (i.e. digits) from 0 to 9, where each digit is uniformly distributed and independent from the others, that is, $\left(x_1^{(\alpha)}, \ldots, x_{30}^{(\alpha)}\right) \in \{0, 1, \ldots, 9\}^{30}$ with $x_t^{(\alpha)} \overset{\text{iid}}{\sim} \mathcal{U}_{\text{int}}(0, 9)$ for $t = 1, \ldots, 30,\ \alpha = 1, \ldots, 10000$.
- A series gets the label 1 if its sum is greater or equal to 100 and the label 0 otherwise.
- Use 8000 series as training set and 2000 series as test set.

(b) Build a recurrent network for number series classification as follows:

- The network is composed of 200 LSTM cells yielding an output vector $\underline{\mathbf{h}}^{(t)}$ in each time step t of the overall 30 time steps.
- On top of the LSTM layer the network there is a single linear output neuron (receiving input from the 200 LSTM cells) with logistic sigmoidal as non-linearity, i.e, $y(\underline{\mathbf{h}}^{(t)}) \in (0, 1)$. $y(\underline{\mathbf{h}}^{(30)})$ should be interpreted as the probability that the sum of the number series is greater or equal to 100.
  Hints for Keras users: Apply a `Dense` layer with one output neuron and sigmoid activation after the `LSTM` layer.

(c) Use *cross-entropy* between the labels of the training data set and the output y of the network *after the last time step* as loss function for the learning process *Hint:* You can use the *cross-entropy* for the binary case as loss function as the labels are either 1 or 0.

(d) Use classification accuracy as performance measure to compare validation and training performance for the trained model.

(e) As many frameworks differ in their implementation of LSTM and allow for different levels of tweaking the parameters you are free to choose parameters which seem reasonable to you (maybe the default values). Also decide on the architecture of the network on your own [1].

(f) For the training procedure iterate over the data 60 times in random fashion (i.e. use 60 epochs) and use a mini-batch size of 50.

(g) Train your model using the *Adam* algorithm. Use the following parameters: $\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 * 10^{-8}$.

Deliverables:

1. (8 points) Evaluate the final accuracy of the model on the validation data.

[1] You can look up a different implementation of LSTM than in the slides in the following article: http://tinyurl.com/q6dcybc. The implementation and notation used there seems to be more similar to most deep learning frameworks.

2. (2 points) Inspect the class distribution (no. of positive examples vs. no. of negative examples). Briefly disucss the validity of the accuracy measure in relation to that distribution and possible alternatives.

Total 10 points.