# 1 Long Short Term Memory

Simple Recurrent Networks (SRNs) can learn medium-range dependencies but have difficulty learning long range dependencies. Long Short Term Memory (LSTM) is able to learn long range dependencies using a combination of forget, input and output gates (Hochreiter and Schmidhuber, 1998).

Figure 1: LSTM

The LSTM maintains a context layer which is distinct from the hidden layer but contains the same number of units. The full workings of the LSTM at each timestep are described by these equations:

**Gates**

$$f_t = \sigma(U_f h_{t-1} + W_{fX_t} + b_f)$$
$$i_t = \sigma(U_i h_{t-1} W_{iX_t} + b_i)$$
$$o_i = \sigma(U_o h_{t-1} + W_{oX_t} + b_o)$$

**Candidate activation**

$$g_t = \tanh(U_g h_{t-1} + W_{gX_t})$$

**State**

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

**Outputs**:

$$h_t = o_t \odot \tanh c_t$$

First, the **forget** gate ($f$) is used to determine, for each context unit, a ratio between 0 and 1 by which the value of this context unit will be multiplied. If the ratio is close to zero, the previous value of the corresponding context unit will be largely forgotten; if it is close to 1, the previous value will be largely preserved.

Next, **update** values ($g$) between $-1$ and $+1$ are computed using tanh, and the input gate ($i$) is used to determine ratios by which these update values will be multiplied before being added to the current context values.

Finally, the **output** gate ($o$) is computed and used to determine the ratios by which tanh of the context unit values will be multiplied in order to produce the next hidden unit values.

In this way, the context units are able to specialise, with some of them changing their values frequently while others preserve their state for many timesteps, until particular circumstances cause the gates to be opened and allow the value of those units to change.

## 1.1 Embedded Reber Grammar

The ability of different sequence processing algorithms to learn long range dependencies can be explored using the Reber Grammar and Embedded Reber Grammar.



Figure 2: reber grammar

The Reber Grammar (RG) is defined by the finite state machine shown on the left. When there is a choice between two transitions, they are understood to be chosen with equal probability. The Embedded Reber Grammar (ERG) is shown on the right, where each box marked **REBER GRAMMAR** contains an identical copy of the finite state machine on the left. The difficulty in learning the ERG is that the network must remember which transition (T or P) occurred after the initial B, and retain this information while it is processing the transitions associated with the RG in one of the two identical boxes, in order to correctly predict the T or P occurring before the final E.

In the exercises for this week, you will be demonstrating that the SRN is able to learn the RG but struggles to learn the ERG, whereas the LSTM can also learn the ERG. We can imagine that one of the context units is somehow assigned the task of retaining the knowledge of the initial T or P, and that this knowledge is preserved by appropriately high and low values for the forget gate and the input and output gate, respectively.

## 1.2 Gated Recurrent Unit



Figure 3: gated recurrent unit

The Gated Recurrent Unit (GRU) is similar to LSTM but has only two gates instead of three. Its update equations are as follows:

**Gates**

$$z_t = \sigma(U_z h_{t-1} + W_z X_t + b_z)$$
$$r_t = \sigma(U_g(r_t \odot h_{t-1}) + W_g X_t + b_g)$$

**Candidate Activation**

**Output**

## 1.3 Optional Video

## 1.4 further reading

Fahlman, S. E. (1991). The recurrent cascade-correlation architecture (Technical Report CMU-CS-91-100). Carnegie-Mellon University, Department of Computer Science.

Hochreiter, S., and Schmidhuber, J., 1997. Long short-term memory. Neural Computation, 9(8), 1735-1780.

Two excellent web resources for LSTM: - Understanding LSTM Networks (Colah, 2015. Github) - LSTM (Long Short Term Memory (Huerta, n.d. christianherta.de) - LaTeX-TutorialUse of Reber Grammar