1 RNN & LSTM

During this lab we will create models that use both architectures.

1.1 Input

One can deal with several types of models here.

- One to one - for each part of the input generate output.
  For example imagine an agent that plays a game and acts each time it receives a frame (input) from the game.
- One to many - one input and several outputs for this input.
  For example for word give me its dictionary description.
- Many to one - Many inputs and one output.
  For example classify review (many words) as either positive or negative (label).
- Many to many - many inputs many outputs.
  For example given a text translate it (note that here a delay between input and output may be helpfull).

Previously those architectures were used for dealing with text related tasks like language modeling.
In this case the input is just a sequence of tokens (represented by natural numbers) that we embed before passing them to LSTM/RNN modules.

In the laboratory scenario we deal of sequences that consist of 0 and 1 and for simplicity we treat each number as its own embedding.

2 RNN

2.1 Definition

Can be described by the following equation:

$$h_t = \tanh\left(W_h h_{t-1} + W_x x_t + b\right)$$

Where:

$$W_h \in \mathbb{R}^{\text{hidden\_dim},\text{hidden\_dim}}$$
$$W_x \in \mathbb{R}^{\text{hidden\_dim},\text{input\_dim}}$$
$$b \in \mathbb{R}^{\text{hidden\_dim}}$$

We usually start with

$$h_0 = 0 \in \mathbb{R}^{\text{hidden\_dim}}$$

or previous hidden state if we continue a spliced sequence.

We create outputs by processing h_t (depending of the use case we can process last one or also the intermediate ones).

2.2 Problem

Note the vanishing/exploding gradient problem:
For simplicity assume hidden_dim = 1 Lets calculate derivatives:

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \tanh\left(w_h h_{t-1} + W_x x_t + b\right)}{\partial\left(w_h h_{t-1} + W_x x_t + b\right)} \frac{\partial\left(w_h h_{t-1} + W_x x_t + b\right)}{\partial h_{t-1}}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \tanh\left(w_h h_{t-1} + W_x x_t + b\right)}{\partial\left(w_h h_{t-1} + W_x x_t + b\right)} w_h$$

The influence of W_h stacks up and either

- Explodes $w_h > 1$
- Vanishes $w_h < 1$
- stays $w_h = 1$

Now in the general case

$$\frac{\partial h_t}{\partial h_{t-1}} = W_h^T \frac{\partial \tanh\left(W_h h_{t-1} + W_x x_t + b\right)}{\partial\left(W_h h_{t-1} + W_x x_t + b\right)}$$

3 LSTM

Can be described by the following equations:

Input:

$$i_t = \sigma\left(W_{i,h}h_{t-1} + W_{i,x}x_t + b_i\right)$$

Forget:

$$f_t = \sigma\left(W_{f,h}h_{t-1} + W_{f,x}x_t + b_f\right)$$

$$g_t = \tanh\left(W_{g,h}h_{t-1} + W_{g,x}x_t + b_g\right)$$

Update cell state: decide what to forget (f_t) from cell state c_{t-1} and what to load (i_t) to c_t from g_t:

$$c_t = f_t * c_t + i_t * g_t$$

where $*$ is element-wise product

$$o_t = \sigma\left(W_{o,h}h_{t-1} + W_{o,x}x_t + b_o\right)$$

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

Briefly speaking we have two memories long-term (c_t) and short-term h_t. We first decide how to update long-term memory and then extract h_t from it.