Tutorial 05: Recurrent Neural Networks

This tutorial aims to introduce recurrent neural networks (RNNs) and their key advantages for sequential data.

1 Recurrent Neural Networks

The core idea behind RNNs is to consider temporal context for sequential data. Common examples of problems that include some kind of temporal information are e.g., tranlation and audio processing.

In this tutorial, we will specifically talk about RNNs that take a sequence of data as input and produce a sequence of predictions of equal length i.e., each instance of the input sequence can be matched to one instance of the output sequence. A simple (fully connected) form of such a network is depicted in Figure 1.

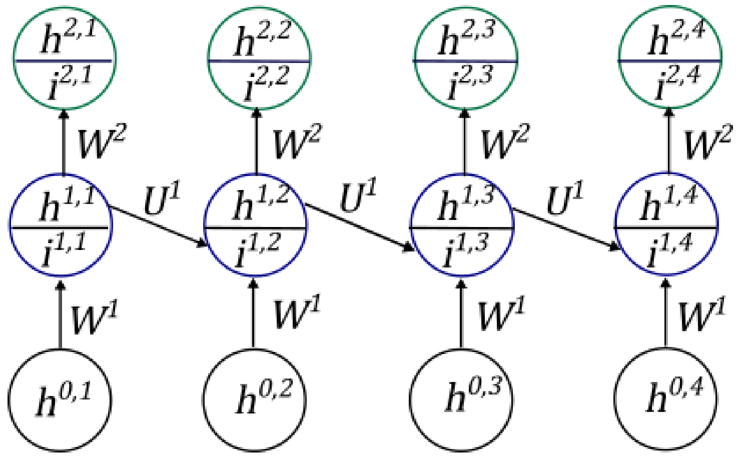

Figure 1: Visualisation of a recurrent neural network.

Like a fully connected network it has a number of layers including an input layer, an output layer and at least one hidden layer. Note, that each circle in Figure 1 does not represent a single neuron, but a whole layer. We will on this exercise sheet only discuss the concepts of the RNN for a single training example, which is in this case a single sequence of data points $\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^t \in \mathbb{R}^1 \times N_0$. Similar to our previous notation, we write $\boldsymbol{h}^{i,\tau} \in \mathbb{R}^1 \times N_i$ for the activation of the neurons in layer i (we may also say ith level) for the τth element of the sequence.

The forward propagation for the first element of the sequence works the same way as for a fully connected network.

$$\boldsymbol{h}^{i,1} = \mathrm{act}^i(\boldsymbol{h}^{i-1,1}\boldsymbol{W}^i + \boldsymbol{b}^i) = \mathrm{act}^i(\boldsymbol{i}^{i,1}), \tag{1}$$

as well as $\boldsymbol{h}^{0,\tau} = \boldsymbol{x}^\tau$. For other elements of the sequence however, any hidden layer is not only depending on the previous layer of the current element, but also on the hidden layer of the same level of the previous element:

$$\boldsymbol{h}^{i,\tau} = \mathrm{act}^i(\boldsymbol{h}^{i-1,\tau}\boldsymbol{W}^i + \boldsymbol{h}^{i,\tau-1}\boldsymbol{U}^i + \boldsymbol{b}^i) = \mathrm{act}^i(\boldsymbol{i}^{i,\tau}), \tag{2}$$

where $\boldsymbol{U}^i \in \mathbb{R}^{N_i \times N_i}$ is a matrix of "horizontal" weights, i.e., connecting neurons of the same hidden layer for two consecutive elements of the sequence. The output $\boldsymbol{h}^n$ of a network, as depicted in Figure 1, however can be determined only by knowing the state of the previous hidden layer for the current element of the sequence i.e., there exists no horizontal information flow in the output layer:

$$\boldsymbol{h}^{n,\tau} = \mathrm{act}^i(\boldsymbol{h}^{n-1,\tau}\boldsymbol{W}^n + \boldsymbol{b}^n) = \mathrm{act}^n(\boldsymbol{i}^{n,\tau}). \tag{3}$$

It is important to note that the weights connecting hidden layers of the same level, as well as those connecting layers of different levels are shared across all elements of the sequence, which is the reason why the matrices $\boldsymbol{W}^i$ and $\boldsymbol{U}^i$ in (2) have no mentioning of τ.

We consider a total loss function that is defined as the mean over all losses of the individual sequence outputs, i.e.,

$$L(\{\hat{\boldsymbol{y}}^{\tau'}\}, \{\boldsymbol{y}^{\tau'}\}) = \frac{1}{t}\sum_{\tau'=1}^{t} L(\hat{\boldsymbol{y}}^{\tau'}, \boldsymbol{y}^{\tau'}), \tag{4}$$

where $\{\hat{\boldsymbol{y}}^{\tau'}\}$ and $\{\boldsymbol{y}^{\tau'}\}$ represents the predictions and labels of all elements (all possible τ')of the sequence. The total loss therefore accounts for the prediction error of every element of the sequence equally.

2 Backpropagation Through Time

As for the feed forward NN, we want to apply the gradient descent algorithm to optimise the loss function of RNNs. However, there are a few subtle differences that need to be considered in order to calculate the gradient of an RNN.

2.1 Backpropagation for Vertical Weights ($\boldsymbol{W}$)

Let us first consider the loss for the last element of the sequence with respect to different network parameters. The derivative with respect to a weight of the output layer can be obtained accordingly to the case of a simple feed forward neural network, as the dependency (and therefore the use of the chain rule) ends at that layer

$$\frac{\mathrm{d}L(\hat{\boldsymbol{y}}^t,\boldsymbol{y}^t)}{\mathrm{d}w^n_{ij}}(\boldsymbol{\Theta},\{\boldsymbol{x}^t\}) = (\boldsymbol{\delta}^{n,t}\boldsymbol{h}^{n-1,t})^{\mathrm{T}}_{ij}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}) \tag{5}$$

with

$$\boldsymbol{\delta}^{n,t}_j = \sum_{k=1}^{N_n}\frac{\mathrm{d}L(\hat{\boldsymbol{y}}^t,\boldsymbol{y}^t)}{\mathrm{d}\hat{\boldsymbol{y}}^t_k}\frac{\mathrm{d}(\mathrm{act}^n(\boldsymbol{i}^{n,t})_k)}{\mathrm{d}\boldsymbol{i}^{n,t}_j}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}). \tag{6}$$

However, determining the gradient component according to the weights connecting layers of lower levels with each other, either vertically ($\boldsymbol{W}^i$) or horizontally ($\boldsymbol{U}^i$) is more difficult. The reason for this is that e.g., a change of weights in the input layer $\boldsymbol{W}^1$ does not only have a direct impact on the hidden state $\boldsymbol{H}^{1,t}$ via the input element $\boldsymbol{H}^{0,t}$, but also via all other input elements $\boldsymbol{H}^{0,i}$ by changing the hidden layers $\boldsymbol{H}^{1,i}$. The reason for this is that the changes in any $\boldsymbol{H}^{1,i}$ are propagated through the sequence to $\boldsymbol{H}^{1,t}$ via the weights $\boldsymbol{U}^1$. The resolution of these dependencies in the gradient is what we call backpropagation through time.

Consider now the following derivative of the loss function of the last element of the sequence

$$\frac{\mathrm{d}L(\hat{\boldsymbol{y}}^t,\boldsymbol{y}^t)}{\mathrm{d}w^{n-1}_{ij}} = \sum_{l=1}^{N_n}\boldsymbol{\delta}^{n,t}_l\frac{\mathrm{d}\boldsymbol{i}^{n,t}_l}{\mathrm{d}w^{n-1}_{ij}}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}) = \sum_{l=1}^{N_n}\boldsymbol{\delta}^{n-1,t}_l\frac{\mathrm{d}\boldsymbol{i}^{n-1,t}_l}{\mathrm{d}w^{n-1}_{ij}}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}), \tag{7}$$

with

$$\boldsymbol{\delta}^{n-1,t}_l(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}) = (\boldsymbol{W}^n\boldsymbol{\delta}^{n,t})_l\frac{\mathrm{d}\,\mathrm{act}^{n-1}(\boldsymbol{i}^{n-1,t})_l}{\mathrm{d}\boldsymbol{i}^{n-1,t}_l}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}). \tag{8}$$

The derivative in (7) has to be broken down into two parts according to (2):

$$\frac{\mathrm{d}\boldsymbol{i}^{n-1,t}_l}{\mathrm{d}w^{n-1}_{ij}}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}) = \boldsymbol{h}^{n-2,t}_i\delta_{jl} + \frac{\mathrm{d}(\boldsymbol{h}^{n-1,t-1}\boldsymbol{U}^{n-1})_l}{\mathrm{d}w^{n-1}_{ij}}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}) \tag{9}$$

with the Kronecker delta $\delta_{jl} = 1$ if $j = l$ and $\delta_{jl} = 0$ otherwise. For the remaining derivative in (9) we now have to apply the chain rule over and over again, whichis very similar to the case of the feed-forward NN, where the error $\boldsymbol{\delta}$ is backpropagated. The backpropagation through time of the error can be formulated according to the backpropagation of a feed forward NN:

$$\boldsymbol{\delta}^{n-1,\tau}_l((\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\})) = (\boldsymbol{U}^{n-1}\boldsymbol{\delta}^{n-1,\tau+1})_l\frac{\mathrm{d}\,\mathrm{act}^{n-1}(\boldsymbol{i}^{n-1,\tau})_l}{\mathrm{d}\boldsymbol{i}^{n-1,\tau}_l}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}). \tag{10}$$

At every step of the backpropagation, i.e., at every element of the sequence, one term is added, leading to the final form of the derivative:

$$\frac{\mathrm{d}L(\hat{\boldsymbol{y}}^t,\boldsymbol{y}^t)}{\mathrm{d}w^{n-1}_{ij}} = \sum_{\tau=1}^{t}(\boldsymbol{\delta}^{n-1,\tau}\boldsymbol{h}^{n-2,\tau})^{\mathrm{T}}_{ij}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}). \tag{11}$$

Derivatives of the loss of any previous prediction $\hat{\boldsymbol{y}}^{\tau}$ of course work the same way, only that the backpropagation through time starts at that particular element of the sequence. Given that the total loss in our case is simply given by the sum of the loss of each individual prediction, the total gradient's component for a particular weight is also given by the sum over the according component of the gradients for each individual prediction. Components of the gradient for weights of the second to last layer are then given by

$$\frac{\mathrm{d}L(\{\hat{\boldsymbol{y}}^{\tau'}\},\{\boldsymbol{y}^{\tau'}\})}{\mathrm{d}w_{ij}^{n-1}} = \sum_{s=1}^{t}\sum_{\tau=1}^{s}(\boldsymbol{\delta}^{n-1,\tau,s}\boldsymbol{h}^{n-2,\tau})_{ij}^{\mathrm{T}}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}), \tag{12}$$

where $\boldsymbol{\delta}^{n-1,\tau,s}$ means that we consider only the sequence of length $t = s$ in (8), i.e., it is the error of the sth element that is backpropagated in (10).

Given the linearity in (12) we can introduce a new combined error term at every element $\tau \leq t$ of the sequence

$$\tilde{\boldsymbol{\delta}}_{l}^{n-1,\tau} = (\boldsymbol{W}^{n}\boldsymbol{\delta}^{n,\tau} + \boldsymbol{U}^{n-1}\tilde{\boldsymbol{\delta}}^{n-1,\tau+1})\frac{\mathrm{d}\,\mathrm{act}^{n-1}(\boldsymbol{i}^{n-1,\tau})_{l}}{\mathrm{d}\boldsymbol{i}_{l}^{n-1,\tau}}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}) \tag{13}$$

and $\tilde{\boldsymbol{\delta}}^{n-1,t+1} = 0$. Picturing this from a graphical point of view, we obtain the combined error at every element of the sequence by adding the horizontally backpropagated error obtained from the following sequence element and the vertically backpropagated error obtained from the error of this element at the next layer. Using the combined error the gradient component can then be expressed as

$$\frac{\mathrm{d}L(\{\hat{\boldsymbol{y}}^{\tau'}\},\{\boldsymbol{y}^{\tau'}\})}{\mathrm{d}w_{ij}^{n-1}} = \sum_{\tau=1}^{t}(\tilde{\boldsymbol{\delta}}^{n-1,\tau}\boldsymbol{h}^{n-2,\tau})_{ij}^{\mathrm{T}}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}). \tag{14}$$

2.2 Backpropagation for Horizontal Weights ($\boldsymbol{U}$)

The gradient component corresponding to the horizontal connections u_{ij}^{n-1} follows quite similar reasoning. Equation (7) still holds when replacing the derivatives with respect to w_{ij}^{n-1} with those with respect to u_{ij}^{n-1}, as previous steps only consider uses of the chain rule and were not specific to w_{ij}^{n-1}. However the remaining derivative of $\boldsymbol{i}^{n-1,t}$ needs to be evaluated seperately. Considering the definition of the input as given in (2), it seems clear that neither the first term including the result of the vertically lower layer, nor the bias depend on the horizontal weight $u_{i}^{n-1}j$. The remainaing derivative can therefore be expressed as

$$\frac{\mathrm{d}\boldsymbol{i}_{l}^{n-1,t}}{\mathrm{d}u_{ij}^{n-1}} = \frac{(\mathrm{d}\boldsymbol{h}^{n-1,t-1}\boldsymbol{U}^{n-1})_{l}}{\mathrm{d}u_{ij}^{n-1}}. \tag{15}$$

As both terms in the nominator depend on the horizontal weight, we need to apply the product rule. E.g., by using the component-wise definition of the matrix multiplication, we can rewrite the derivative as

$$\frac{\mathrm{d}\boldsymbol{i}_l^{n-1,t}}{\mathrm{d}u_{ij}^{n-1}} = \boldsymbol{h}^{n-1,t-1}\delta_{jl} + \left(\frac{\mathrm{d}\boldsymbol{h}^{n-1,t-1}}{\mathrm{d}u_{ij}^{n-1}}\boldsymbol{U}^{n-1}\right)_l, \tag{16}$$

which looks quite similar to (9).

The gradient components accoriding to the horizontal weights can therefore be expressed in a similar manner using the backpropagation of the combined error as

$$\frac{\mathrm{d}L(\{\hat{\boldsymbol{y}}^{\tau'}\},\{\boldsymbol{y}^{\tau'}\})}{\mathrm{d}u_{ij}^{n-1}} = \sum_{\tau=2}^{t}(\tilde{\boldsymbol{\delta}}^{n-1,\tau}\boldsymbol{h}^{n-1,\tau-1})_{ij}^{\mathrm{T}}(\boldsymbol{\Theta},\{\boldsymbol{x}^{\tau'}\}). \tag{17}$$

Note that the sum in (17) has one element less than the sum in (14), because the $n-1$th hidden layer of the first element of the sequence $\boldsymbol{h}^{n-1,0}$ does not depend on $\boldsymbol{U}^{n-1}$ anymore, but still depends on $\boldsymbol{W}^{n-1}$. Gradient components for the bias follow similarly by adjusting the derivatives, but are not needed for this week's exercise sheet.

For the gradient components according to weights of lower layers, the backpropagation can be done in a very similar manner. However, it is important to consider all influencing paths (in the graphical representation) when calculating the combined errors. As the network on the exercise sheet only has one hidden layer, we set aside the corresponding formulas at this point.