## Artificial Neural Networks (Gerstner). Solutions for week 6

### Recurrent neural networks

### Exercise 1. Unfolding in time

We consider a neural network with one recurrent hidden layer as shown in class. The output is (we have suppressed the threshold $\vartheta$)

$$\hat{y}_i(t) = g[\sum_j w_{ij}^{(2)} x_j^{(1)}(t)] \tag{1}$$

and neurons in the hidden layer have an activity

$$x_j^{(1)} = g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t) + \sum_i w_{ji}^{(rec)} x_i^{(1)}(t-1)] \tag{2}$$

a. Evaluate the output at time step $t = 4$ in terms of the weights and the inputs $x_k^{(0)}$ given at time steps $t = 1, 2, 3, 4$.

Hint: Insert the formula for $x_j^{(1)}$ into the formula for the output, and repeat the procedure recursively. Keep track of the time steps!

b. Construct an equivalent feedforward network.

**Solution:**

a.

$$\begin{aligned}
\hat{y}_i(t=4) =& g[\sum_j w_{ij}^{(2)} x_j^{(1)}(t=4)] && (3)\\
=& g[\sum_j w_{ij}^{(2)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=4) + \sum_i w_{ji}^{(rec)} x_i^{(1)}(t=3)]] && (4)\\
=& g[\sum_j w_{ij}^{(2)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=4) && (5)\\
&+ \sum_i w_{ji}^{(rec)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=3) + \sum_i w_{ji}^{(rec)} x_i^{(1)}(t=2)]]] && (6)\\
=& g[\sum_j w_{ij}^{(2)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=4) && (7)\\
&+ \sum_i w_{ji}^{(rec)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=3) && (8)\\
&+ \sum_i w_{ji}^{(rec)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=2) + \sum_i w_{ji}^{(rec)} x_i^{(1)}(t=1)]]]] && (9)\\
=& g[\sum_j w_{ij}^{(2)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=4) && (10)\\
&+ \sum_i w_{ji}^{(rec)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=3) && (11)\\
&+ \sum_i w_{ji}^{(rec)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=2) && (12)\\
&+ \sum_i w_{ji}^{(rec)} g[\sum_k w_{jk}^{(1)} x_k^{(0)}(t=1)]]]]] && (13)
\end{aligned}$$

b. Such a recurrent neural network can be unrolled into a feedforward equivalent with $t(=4)$ hidden layers and one output layer. The weights of the $t$ hidden layers are similar (or shared). The first

layer receives $x_k^{(0)}(t=1)$ as input and its output is fully connected to the upstream layer which receives $x_k^{(0)}(t=2)$ as external input. And so on, until the $t=4$ hidden layer which is fully connected to the output layer.

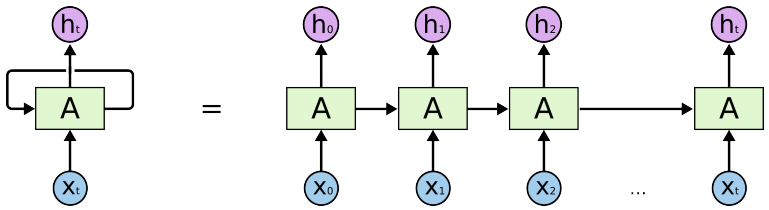


Figure 1: Unrolling through time of a RNN.

**Exercise 2. Vanishing or diverging gradients in recurrent neural networks.**

Consider a standard recurrent neural network with dynamics

$$h_t = \sigma(W^{\text{in}} x_t + W^{\text{rec}} h_{t-1} + b^{\text{h}}) \tag{14}$$
$$y_t = \sigma(W^{\text{out}} h_t + b^{\text{y}}) \tag{15}$$

and a recurrent neural network with LSTM-units

$$c_t = f_t c_{t-1} + i_t \sigma(W^{\text{in}} x_t + W^{\text{rec}} h_{t-1} + b^{\text{h}}) \tag{16}$$
$$h_t = o_t \sigma(c_t) \tag{17}$$
$$y_t = \sigma(W^{\text{out}} h_t + b^{\text{y}}), \tag{18}$$

where we omit the update equations for the forget $f_t$, input $i_t$ and output $o_t$ gates. For simplicity we assume all variables in the above equations are scalars and $h_0 = c_0 = 0$. We make the following additional assumptions:

| time step | input | input gate | forget gate | output gate |
|---|---|---|---|---|
| $t = 1, 2$ | $x_t = 1$ | $i_t = 1$ | $f_t = 0$ | $o_t = 0$ |
| $2 < t < T$ | $x_t = 0$ | $i_t = 0$ | $f_t = f$ | $o_t = 0$ |
| $t = T$ | $x_t = 0$ | $i_t = 0$ | $f_t = f$ | $o_t = 1$ |

a. Compute $\frac{dy_T}{dW^{\text{in}}}$ for the standard recurrent neural network.
Use the abbreviations $a_t^{\text{y}} = W^{\text{out}} h_t + b^{\text{y}}$ and $a_t^{\text{h}} = W^{\text{in}} x_t + W^{\text{rec}} h_{t-1} + b^{\text{h}}$.

b. Which value does $\frac{dy_T}{dW^{\text{in}}}$ approach with increasing $T$, if $|\sigma'(a_t^{\text{h}}) W^{\text{rec}}| > 1$ for all $t$? If $|\sigma'(a_t^{\text{h}}) W^{\text{rec}}| < 1$?

c. Compute $\frac{dy_T}{dW^{\text{in}}}$ for the LSTM network, using the above assumptions.

d. Which values does the result approach for different values of $|W^{\text{rec}}|$ and $f$?

**Solution:**

a.

$$\frac{dy_T}{dW^{\text{in}}} = \sigma'(a_T^{\text{y}}) W^{\text{out}} \frac{dh_T}{dW^{\text{in}}} \tag{19}$$
$$= \sigma'(a_T^{\text{y}}) W^{\text{out}} \sigma'(a_T^{\text{h}}) \left( x_T + W^{\text{rec}} \frac{dh_{T-1}}{dW^{\text{in}}} \right) \tag{20}$$
$$= \sigma'(a_T^{\text{y}}) W^{\text{out}} \sigma'(a_T^{\text{h}}) (W^{\text{rec}})^{T-2} \prod_{t=2}^{T-1} \sigma'(a_t^{\text{h}}) \left( x_2 + W^{\text{rec}} \sigma'(a_1^{\text{h}}) x_1 \right) \tag{21}$$

b. If $|W^{\text{rec}}| > 1$, $(W^{\text{rec}})^{T-2}$ diverges with increasing $T$ and thus $\frac{dy_T}{dW^{\text{in}}}$ diverges. If $|W^{\text{rec}}| < 1$, $\frac{dy_T}{dW^{\text{in}}}$ vanishes (converges to 0).

c.

$$\frac{dy_T}{dW^{\text{in}}} = \sigma'(a_T^{\text{y}})W^{\text{out}}o_T\sigma'(c_T)\frac{dc_T}{dW^{\text{in}}} \tag{22}$$

$$= \sigma'(a_T^{\text{y}})W^{\text{out}}o_T\sigma'(c_T)f^{T-2}\frac{dc_2}{dW^{\text{in}}} \tag{23}$$

$$= \sigma'(a_T^{\text{y}})W^{\text{out}}o_T\sigma'(c_T)f^{T-2}\sigma'(a_2^{\text{h}})x_2 \tag{24}$$

$$\tag{25}$$

d. The result is independent of $|W^{\text{rec}}|$, vanishes for $f < 1$ and is independent of $T$ for $f = 1$.

**Exercise 3. Geometrical Theory of Threshold Neural Networks**

In this exercise we consider networks consisting of step-function neurons: $g(a) = 0.5[1 + sgn(a)]$ and associate each neuron with a hyperplane. The total input is $a = \sum_k w_k x_k - \theta$. We start in input dimension $d = 2$ and go later to $d = 10$.

a. Draw 5 non-parallel straight lines in generic position on a white sheet of paper in front of you. Generic means: Your configuation should not be a special case. More precisely, each line should cross within the region on the paper the four other lines and no three lines can go through the same point.

b. Label the lines from 1 to 5. Associate to each line one hidden neuron in the first hidden layer that implements the corresponding hyperplane (=line in 2d).

c. Label the distinct regions A, B, C, ... starting at the top, running clockwise around in the outer circle of regions and then continue with labeling the inner regions

d. How many distinct regions are there?

e. Mark your regions A, D, G in grey shading

f. Construct a neural network that assigns input patterns from regions A,D,G to the same class C and all other regions to non-C.

Hint: You will need a second hidden layer. Set all weights to either +1 or -1. Start by constructing a neuron in the second hidden layer that only responds to input in region A.

g. Now let us assume that the input dimension is $d = 10$ and you use $n = 20$ neurons in the first hidden layer. Are the numbers for $d = 10$ and $n = 20$ much smaller or larger than those occurring in a standard neural network for e.g. image classification? Or are they in the the same order of magnitude?

h. With the set-up of (g), how many distinct regions are there?

Hint: use approximate formula from class. You can approximate $2^{10}$ by 1000.

i. We now construct a second hidden layer. How many neurons will you need in the second hidden layer if each of these neurons respond to exactly one region?

Is the size of this second hidden layer smaller or larger compared to those in a standard artificial neural network (ANN), e.g. for image classification?

What can you conclude?

Hint: Think of flexibility, regularization, generalization, and the no-free-lunch theorem and write a conclusion in four sentences along the following scheme

Even though the second layer in a standard ANN rather large, the size of this second layer is ...

Since the no-free lunch theorem states that ...

Moreover, good generalization requires that ...

Taking these points together, it follows that ...

j. Do you agree with the following statement: 'It is possible to solve each classification task with just two hidden layers, but the number of neurons in the second hidden layer will have to grow exponentially'.

**Solution:**

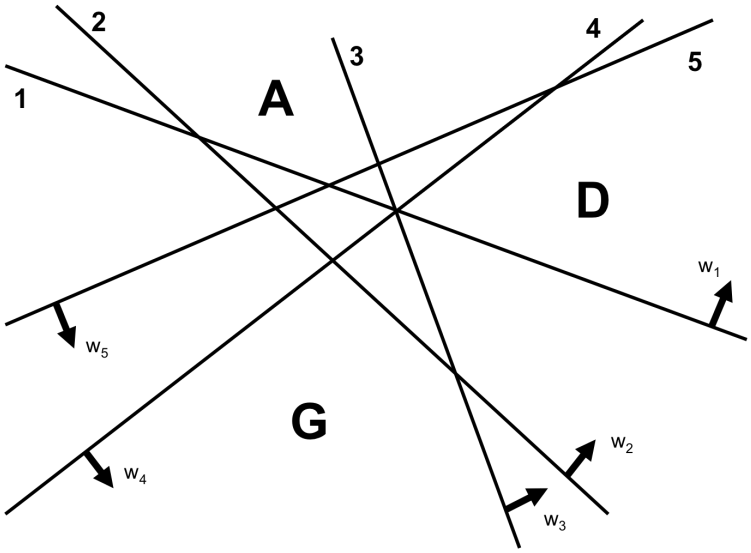


Figure 2: Sketch for Exercise 3a, b, c, and e.

d. 16

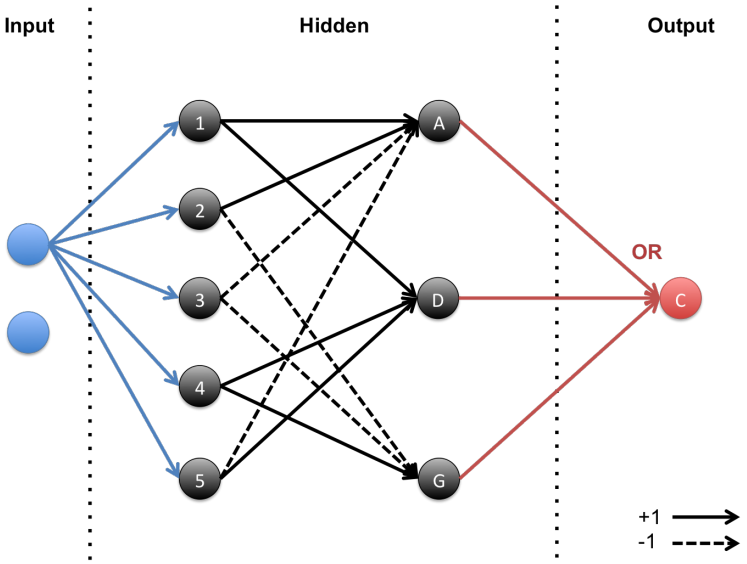


Figure 3: Solution for Exercise 3f.

g. Much smaller. In a typical ANN the input image is $d = 16 \times 16$ and there was up to $n = 1000$ hidden units.

h. The approximate formula is $number \sim n^d = 2^{10}10^{10} = 10^{13}$.

i. We would need $10^{13}$ neurons in the second hidden layer. Even though the second layer in a standard ANN is rather large, the size of this second layer is much larger. Since the no-free lunch theorem states that *any two optimization algorithms are equivalent when their performance is averaged across all possible problems*, the general idea that an ANN should be able to solve any (and every) problem is wrong. Moreover, good generalization requires that data points falling into previously unexplored regions are correctly classified. Taking these points together, it follows that the number of units should be smaller than the number of possible regions in order for the classification ANN to generalize.

j. As illustrated in this exercise, you should agree with this statement.