Benjamin Grewe, Matthew Cook, Giacomo Indiveri, Daniel Kiper, Wolfger von der Behrens, Valerio Mante
Lecture 13
Asst: Risi Nicoletta, Raphaela Kreiser, Ioana Calangiu {nicoletta, rakrei, calangiu}@ini.uzh.ch

Solution 13.1: Error Back Propagation

1. The error function for a single training sample is

$$E(S) = \frac{1}{2}\left(f(x_1 w_1 + x_2 w_2 + w_0) - d\right)^2 . \tag{1}$$

For general activation functions $f(a)$ there is no way to find an analytical solution (w_1, w_2, w_0) that minimizes the error. However, since we assume that $f(a)$ is differentiable, we can perform a gradient descent to find a local minimum of $E(S)$. For that, we regard the error function to be a function of the weights:

$$E(w_1, w_2, w_0) = \frac{1}{2}(f(x_1 w_1 + x_2 w_2 + w_0) - d)^2. \tag{2}$$

Making repeated use of the chain rule ($\frac{\partial x}{\partial y} = \frac{\partial x}{\partial z}\frac{\partial z}{\partial y}$), we can now compute the partial gradients with respect to any of the weights, *e.g.*,

$$\frac{\partial E(w_1, w_2, w_0)}{\partial w_1} = (f(x_1 w_1 + x_2 w_2 + w_0) - d) \cdot f'(x_1 w_1 + x_2 w_2 + w_0) \cdot x_1, \tag{3}$$

$$\frac{\partial E(w_1, w_2, w_0)}{\partial w_2} = (f(x_1 w_1 + x_2 w_2 + w_0) - d) \cdot f'(x_1 w_1 + x_2 w_2 + w_0) \cdot x_2, \tag{4}$$

and

$$\frac{\partial E(w_1, w_2, w_0)}{\partial w_0} = (f(x_1 w_1 + x_2 w_2 + w_0) - d) \cdot f'(x_1 w_1 + x_2 w_2 + w_0). \tag{5}$$

Note that for the derivation of these gradients we don't need to know $f(a)$ – it is enough if we know that it is differentiable.

If we had more than one training sample we would simply sum up all partial gradients over all training samples (the derivative of a sum is the sum of its derivatives).

2. The output of this simple network is

$$y(x_1, x_2, x_3) = f(x_1 w_{21} + f(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} + w_{20}) \tag{6}$$

and the corresponding error function

$$E(w_{11}, w_{12}, w_{10}, w_{21}, w_{22}, w_{20}) = \frac{1}{2}(f(x_1 w_{21} + f(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} + w_{20}) - d)^2. \tag{7}$$

The derivative to w_{22} is

$$\begin{aligned} \frac{\partial E}{\partial w_{22}} =& (f(x_1 w_{21} + f(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} + w_{20}) - d) \cdot & (8)\\ & f'(x_1 w_{21} + f(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} + w_{20}) \cdot & (9)\\ & f(x_2 w_{11} + x_3 w_{12} + w_{10}). & (10) \end{aligned}$$

The derivative to w_{11} is

$$\begin{aligned} \frac{\partial E}{\partial w_{11}} =& (f(x_1 w_{21} + f(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} + w_{20}) - d) \cdot & (11)\\ & f'(x_1 w_{21} + f(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} + w_{20}) \cdot & (12)\\ & f'(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} \cdot & (13)\\ & x_2 & (14) \end{aligned}$$

and to w_{12}

$$\begin{aligned} \frac{\partial E}{\partial w_{12}} =& (f(x_1 w_{21} + f(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} + w_{20}) - d) \cdot & (15)\\ & f'(x_1 w_{21} + f(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} + w_{20}) \cdot & (16)\\ & f'(x_2 w_{11} + x_3 w_{12} + w_{10}) w_{22} \cdot & (17)\\ & x_3. & (18) \end{aligned}$$

The first two lines are identical in each gradient and can be reused once computed. The last line of each gradient ((10), (14), and (18)) is the value of the current input that is multiplied by the respective weight. Between the gradients for w_{11} and w_{12} only the input part is changing.

In general, the non-input part of the gradients is the same for all weights of one unit and is called the *delta term* or *error term*. In our example, the delta term of the right unit consists of lines (8) and (9). The delta term of the left unit consists of lines (11), (12), and (13) or, equivalently, of lines (15), (16), and (17). The delta term of a lower[1] unit can be obtained from the delta terms of the higher units it projects to: the higher delta term is just multiplied by the weight connecting both units and by the derivative of the activation function of the lower unit. In our example, the delta term of the left unit is the delta term of the right unit (lines (8) and (9)) multiplied by $f'(x_2w_{11} + x_3w_{12} + w_{10})w_{22}$, *i.e.*, by the derivative of the activation function of the left unit and the weight connecting the units (see line (13) or (17)).

Thus, it is sufficient to compute the delta terms for each unit to get the derivatives by the weights. The fact that the delta terms represent the error contributed by each unit and that they are computable recursively from higher units gave this learning procedure the name *error back propagation*.

3. We can think of the delta terms as travelling back through the wires. Whenever they leave a neuron through an input, they are multiplied by the weight of the input. Whenever they enter a neuron through its output, they are multiplied by the derivative of the activation function of that input.

   Whenever the output of one neuron is used several times, all delta terms that meet at a junction are simply added.

Solution 13.2: Hopfield Networks

1. The matrix is symmetric with zero entries in the diagonal.

2. For the patterns to be stable the neurons are not supposed to change when updated. This implies that when one of the patterns is applied to the network, the activation of every neuron is positive if its state is supposed to be 1, and negative otherwise.

   From this requirement we can work out constraints on the weights w_{ij} connecting the neurons i and j (remember that $w_{ij} = w_{ji}$). For the pattern $(-1, 1, 1 - 1)$, these are:

$$\begin{aligned} w_{21} + w_{31} - w_{41} &< 0 \\ -w_{12} + w_{32} - w_{42} &\geq 0 \\ -w_{13} + w_{23} - w_{43} &\geq 0 \\ -w_{14} + w_{24} + w_{34} &< 0 \end{aligned}$$

   For the pattern $(-1, 1, -1, 1)$, these are:

$$\begin{aligned} w_{21} - w_{31} + w_{41} &< 0 \\ -w_{12} - w_{32} + w_{42} &\geq 0 \\ -w_{13} + w_{23} + w_{43} &< 0 \\ -w_{14} + w_{24} - w_{34} &\geq 0 \end{aligned}$$

   Sometimes, the network has to find in a learning task a weight matrix $\boldsymbol{W}$, such that all the above mentioned constraints are fulfilled. Note that this is not always possible.

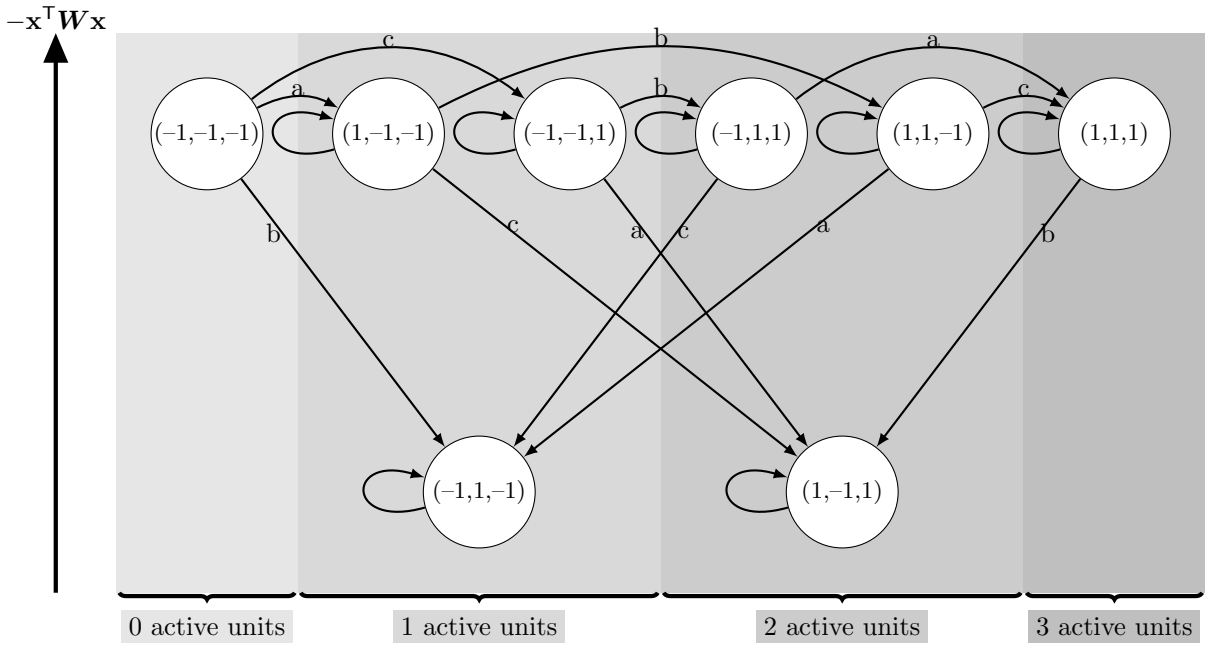
   Here, it is possible and a concrete example is:

$$\boldsymbol{W} = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{pmatrix} = \begin{pmatrix} 0 & -10 & 6 & 8 \\ -10 & 0 & 7 & 9 \\ 6 & 7 & 0 & -11 \\ 8 & 9 & -11 & 0 \end{pmatrix}$$

3. Each neuron can be in one of two states: $2^3 = 8$.

4.-5. In the following figure you see all possible states the network can be in and the transitions when updating the first, second, or third neuron (a, b, and c). The height of a state corresponds to its 'energy' $-\mathbf{x}^\mathsf{T}\boldsymbol{W}\mathbf{x}$. From this figure we can see that $(1, -1, 1)$ and $(-1, 1, -1)$ are stable states of the network (fundamental memories).

[1] Here, *lower* means closer to the input. In our example, the left unit is lower than the right unit.



The states are ordered in a way such that towards the right the number of active unit increases. It is no coincidence that the arrows:

- never go up in energy
- if energy stays the same, arrows never go from the right to the left (in this case, the number of active units never decreases).

The last point is a consequence of the threshold condition that if the weighted sum equals zero, the unit stays/gets active. This property can be used to show that we always reach a stable state. However, we do not go into more details for this fundamental property of Hopfield networks.