Benjamin Grewe, Matthew Cook, Giacomo Indiveri, Daniel Kiper, Wolfger von der Behrens, Valerio Mante
Lecture 12
Asst: Lucas Pompe, Eduarda Streit Morsch, Reinhard Loidl{lucas, eduarda, rloidl}@ini.uzh.ch

Exercise 12.1: Feed-forward networks and Backpropagation

Feed-forward neural networks consist of several layers of neurons. Let's denote the inputs to the first layer as $\mathbf{x}$ and the output(s) of the last layer as $\mathbf{y}$. Training such a network means finding a set of weights of the neurons, such that for a given input $\mathbf{x}$ the network answers with the desired output $\mathbf{y}$. However, we don't know exactly the functional dependency between $\mathbf{x}$ and $\mathbf{y}$ (if we knew, we wouldn't need to build a neural network to learn it). Mostly, we are only given some samples

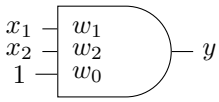
$$S = \{(\mathbf{x}, \mathbf{d})^1, \ldots, (\mathbf{x}, \mathbf{d})^l\} \tag{1}$$

of inputs ($\mathbf{x}^i$) and the corresponding desired output ($\mathbf{d}^i$). Training the network now means that we want it to get these samples right, or at least *as right as possible*. For that, we define an error function

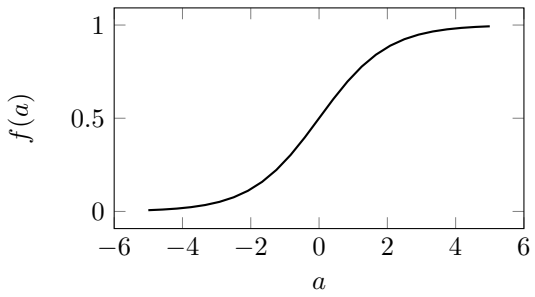
$$E(S) = \sum_i \frac{1}{2} \|\mathbf{y}(\mathbf{x}^i) - \mathbf{d}^i\|^2 \tag{2}$$

that tells us by how much we are off the desired output. Here $\mathbf{y}(\mathbf{x})$ denotes the output of the network given the input $\mathbf{x}$. The goal of the training is to tune the weights of the neurons such that the error function is minimized.

1. Let's consider the most simple neural network, consisting of only one unit with two inputs:



   The weighted sum $a = x_1 w_1 + x_2 w_2 + w_0$ of the inputs is called the activation of that neuron. The output is a non-linear transformation $y = f(a)$ of the activation, where f, the activation function, is usually a sigmoid:



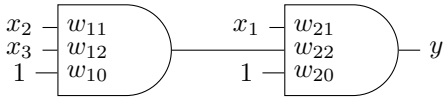
   (a) Given the above activation function and a single training sample $((x_1, x_2), (d))$, what is the error function?

   (b) By looking at the error function, do you think that there is a closed form to compute the minimum for general activation functions? If yes, what is it? If no, what could we do instead to minimize the error?

   (c) What is the derivative of the error function with respect to w_1 and w_2? You can assume that $f(a)$ is differentiable everywhere and write $f'(a)$ for $\frac{\partial f(a)}{\partial a}$. (Hint: You have to use the chain-rule.)

   (d) What would change if we had more than just one training sample?

2. Now we increase the complexity of our network by replacing the former input x_2 with the output of another neuron having the same activation function $f(a)$:



   (a) What is the output $y(\mathbf{x})$ of the network?

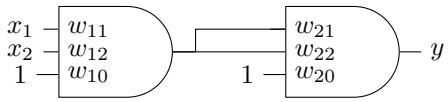
(b) Given a single training sample $((x_1, x_2, x_3), (d))$, what is the error function?

(c) What is the derivative of the error function with respect to w_{22}?

(d) What is the derivative of the error function with respect to w_{11} and w_{12}?

(e) Look at your derivatives for w_{11} and w_{12}. Which terms of the computation of the derivative for w_{22} can be reused?

3. Finally, we want to see the changes if we reuse the output of one neuron. For that, we change our network as follows:



Using the same samples and ignoring the input values for x_1, answer the following questions:

(a) Given a sample $((x_1, x_2), (d))$, what is the error function?

(b) What is the derivative of the error function with respect to w_{11} and w_{12}? How are the terms of the computation of the gradients of w_{21} and w_{22} combined now?