

## Neural Networks

### Exercise 1 : Gradient Descent

(a) What are the differences between the perceptron training rule and the gradient descent method?

Answer

- The perceptron training rule updates the perceptron weights based on the error in the thresholded perceptron output, which can be 0, 1, or -1. Hence, only the sign of the error is used.
- Gradient descent updates the weights based on the delta rule, proportional to the error gradient.
- The perceptron training rule converges after a finite number of iterations, provided the training examples are linearly separable.
- The gradient descent converges asymptotically regardless of whether the training data is linearly separable.

(b) What are the requirements for gradient descent being successful as a learning algorithm?

Answer

- The hypotheses in the hypothesis space  $H$  are based on continuous parameters.
- The learning error can be differentiated with respect to hypothesis parameters.
- Ideally, the error function is convex, i.e. it has a single global minimum that gradient descent can converge to (if this is not given, gradient descent can still be used to find a locally optimal solution, but is not guaranteed to find a global optimum)

### Exercise 2 : Perceptron Learning

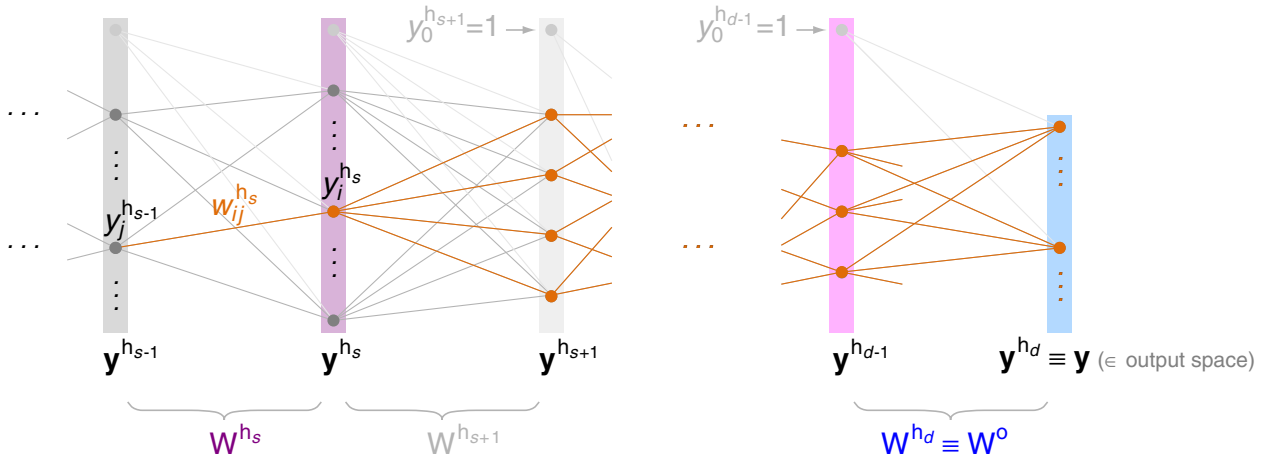
How can perceptrons be applied to solve a classification problem with more than two classes?

Answer

Instead of a thresholded class value, output one value for each class ("logits"). To process these values with a loss, make these outputs probabilities: Map them to the interval  $(0, 1)$  and normalize their sum ("softmax"). For the training, use a loss that not only considers the correctness of the most probably class but the whole distribution ("cross-entropy loss"). For the classification step (testing/inference): Find the argmax of the logits.

### Exercise 3 : From Chain Rule to Backpropagation

Consider the weight  $w_{ij}^{h_s}$  in the following graph of a neural network:



In order to update the weight, we want to compute the derivative  $\frac{\partial L(\mathbf{w})}{\partial w_{ij}^{h_s}}$ .

- Verify the correctness of the following chain rule:

$$\frac{\partial L}{\partial w_{ij}^{h_s}} = \frac{\partial L}{\partial y_i^{h_s}} \cdot \frac{\partial y_i^{h_s}}{\partial net_i^{h_s}} \cdot \frac{\partial net_i^{h_s}}{\partial w_{ij}^{h_s}} \quad (1)$$

with  $net_i^{h_s} := \sum_j w_{ij}^{h_s} \cdot y_j^{h_{s-1}}$  and  $y_i^{h_s} = \sigma(net_i^{h_s})$ .

- What is  $\frac{\partial y_i^{h_s}}{\partial net_i^{h_s}}$ ?

**Answer**

$\frac{\partial y_i^{h_s}}{\partial net_i^{h_s}} = \frac{\partial \sigma(net_i^{h_s})}{\partial net_i^{h_s}}$ , i.e., the derivative of the activation function at  $net_i^{h_s}$ .

- What is  $\frac{\partial net_i^{h_s}}{\partial w_{ij}^{h_s}}$ ?

**Answer**

$\frac{\partial net_i^{h_s}}{\partial w_{ij}^{h_s}} = y_j^{h_{s-1}}$

- $\frac{\partial L}{\partial y_i^{h_s}}$  can not be computed directly, but is based on results from a previous step. Identify terms from the network this value depends on.

**Answer**

Because all values  $y_l^{h_{s+1}}$  (within  $L$ ) depend on the value of  $y_i^{h_s}$ , we can apply the chain rule for multivariable functions:

$$\begin{aligned} \frac{\partial L}{\partial y_i^{h_s}} &= \sum_l \frac{\partial L}{\partial y_l^{h_{s+1}}} \cdot \frac{\partial y_l^{h_{s+1}}}{\partial net_l^{h_{s+1}}} \cdot \frac{\partial net_l^{h_{s+1}}}{\partial y_i^{h_s}} \\ &= \sum_l \frac{\partial L}{\partial y_l^{h_{s+1}}} \cdot \frac{\partial y_l^{h_{s+1}}}{\partial net_l^{h_{s+1}}} \cdot w_{li}^{h_{s+1}}, \end{aligned}$$

and these values are equivalent to the terms in Eq. 1, but in a previous step ("to the right" of the current layer in the network, highlighted in orange in the graph above).

- Which of those terms are computed in the forward propagation, which are computed in the backpropagation?

Answer

$\frac{\partial y_i^{h_s}}{\partial net_i^{h_s}}$  and  $\frac{\partial net_i^{h_s}}{\partial w_{ij}^{h_s}}$  can easily be computed in the forward propagation ("from left to right"). Afterwards, the remaining terms can be computed in the backpropagation ("from right to left").