Probabilistic Unsupervised Learning
Exercise 6: Perceptrons

Submission deadline: Monday, January 31, 2022 at 10:00 a.m.

In the programming exercises, support is only
provided for Matlab or Python source code.

1. Consider a 2-layer Perceptron with inputs $s_i^{(1)}$, outputs $s_h^{(0)}$ and weight matrix W_{hi}. Applying gradient descent on the error function $E(W)$ from the lecture, derive the delta learning rule for the weights W_{hi} using the following activation functions:

   [2 *points*] Task A: linear

   $$s_h^{(0)} = g_h(\vec{s}^{(1)}, W) = \sum_i W_{hi} s_i^{(1)}, \tag{1}$$

   [3 *points*] Task B: hyperbolic tangent

   $$s_h^{(0)} = g_h(\vec{s}^{(1)}, W) = \frac{\exp\big(2\sum_i W_{hi} s_i^{(1)}\big) - 1}{\exp\big(2\sum_i W_{hi} s_i^{(1)}\big) + 1}. \tag{2}$$

   Hint: Use $\Delta W_{hi} = -\epsilon \frac{\partial E(W)}{\partial W_{hi}}$.
   Write down explicitly how to solve the derivatives.

2. Consider the 2-layer Perceptron with two input units and one output unit shown in Fig. (1):

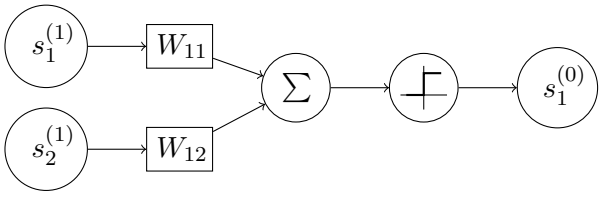

Figure 1: 2-layer Perceptron with Heaviside activation function

Use the following four data points $\vec{x}^{\,(1\ldots4)}$ as inputs $s_i^{(1)}$ and the corresponding binary labels $l^{\,(1\ldots4)}$ as target outputs:

$$\begin{aligned}\vec{x}^{\,(1)} &= (0,0), \quad l^{\,(1)} = 0\\ \vec{x}^{\,(2)} &= (1,0), \quad l^{\,(2)} = 1\\ \vec{x}^{\,(3)} &= (0,1), \quad l^{\,(3)} = 1\\ \vec{x}^{\,(4)} &= (1,1), \quad l^{\,(4)} = 1\end{aligned}$$

[2 *points*] Task A:
Plot the four data points in a 2-D scatter plot and color them by their label with a suitable color mapping of your choice. What logical function do they represent?

[1 *points*] Task B:
For the binary classification task, we now modify the linear activation function of Eq. (1) to the following Heaviside function (as also depicted in Fig. 1):

$$s_1^{(0)} = \begin{cases} 1, & \text{if } W_{11}s_1^{(1)} + W_{12}s_2^{(1)} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

What would be the problem of deriving update rules for this activation function as you did in Task 1?

[5 *points*] Task C:
Implement the Perceptron of Fig. 1 using the Heaviside function of Eq. (3) and the following delta update rule to update the weights after each presented data point:

$$\Delta W_{1i} = -\epsilon\big(s_1^{(0)} - t\big)s_i^{(1)}, \quad \text{with target output } t. \tag{4}$$

$$W_{1i} = W_{1i} + \Delta W_{1i} \tag{5}$$

Initialize all weights with $W_{1i} = 0$ and use a learning rate of $\epsilon = 1$. Update the weights w.r.t. the above given data points and labels in numerical order, i.e. from $\vec{x}^{\,(1)}$ to $\vec{x}^{\,(4)}$, and starting at $\vec{x}^{\,(1)}$ again. At each update step print the output, target, and the updated weights. Keep iterating until there are no more changes throughout a full iteration over all data points. (Hint: This should not take many iterations.)

[5 *points*] Task D:
Plot the resulting decision boundary into the 2-D scatter plot of Task A. Repeat Task C with different initializations of W and different learning rates ϵ. What do you observe and why?

[2 *points*] Task E:
Change the label of the last data point to $l^{\,(4)} = 0$ and repeat Tasks C. What do you observe and why?

[5 *bonus points*] Task F:
Switch the labels of the data points to

$$l^{\,(1)} = 1, \quad l^{\,(2)} = l^{\,(3)} = l^{\,(4)} = 0,$$

and apply the same Perceptron as before. Is the model still able to classify all points correctly? If not, why not and what could you do to correct this?