Note: intermediate steps were calculated using `numpy` and, in general, will not be shown.

1. Considering the following linearly separable training data:

	y_1	y_2	y_3	z
x_1	0	0	0	-1
x_2	0	2	1	1
x_3	1	1	1	1
x_4	1	-1	0	-1

Given the perceptron learning algorithm with a learning rate $\eta = 1$, sign activation and all weights initialized to one (including the bias):

(a) Considering y_1 and y_2, apply the algorithm until convergence. Draw the separation hyperplane.

(b) Considering all input variables, apply one epoch of the algorithm. Do weights change for an additional epoch?

(c) Identify the perceptron output for $x_{new} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$.

(d) What happens if we replace the sign function with the step function? Specifically, how would you change η to ensure the same results?

(a)

(b)

(c)

(d)

2. Show graphically, instantiating the parameters, that a perceptron:

(a) Can learn the NOT, AND and OR logical functions.

(b) Can't learn the XOR logical function (for two inputs).

(a)

(b)

3. Let us consider the following activation function:

$$\hat{z}(x, w) = \frac{1}{1 + e^{-2wx}}$$

Consider also the half sum of squared errors as the loss function:

$$E(w) = 1/2 \sum_{i=1}^{N} (z_i - \hat{z}(x_i, w))^2$$

(a) Determine the gradient descent learning rule for this unit.

(b) Compute the first gradient descent update, assuming an initialization of all ones.

(c) Compute the first stochastic gradient descent update assuming an initialization of all ones.

(a)

(b)

(c)

4. Let us consider the following activation function:

$$\hat{z}(x, w) = \frac{1}{1 + e^{-wx}}$$

Here, we'll be using the cross-entropy loss function:

$$E(w) = -\sum_{i=1}^{N} z_i \log \hat{z}(x_i, w) + (1 - z_i) \log(1 - \hat{z}(x_i, w))$$

(a) Determine the gradient descent learning rule for this unit.

(b) Compute the first gradient descent update, assuming an initialization of all ones.

(c) Compute the first stochastic gradient descent update assuming an initialization of all ones.

(a)

(b)

(c)

5. Consider now the activation function described in the previous exercise, paired with the half sum of squared errors loss function.

(a) Determine the gradient descent learning rule for this unit.

(b) Compute the stochastic gradient descent update for input $x_{new} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, $z_{new} = 0$, with initial weights $w = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$ and learning rate $\eta = 2$.

(a)

(b)

6. Consider the sum squared and cross-entropy loss functions. Any stands out? What changes when one changes the loss function?