Aprendizagem 2022

Labs 5-6: Perceptron and Gradient Descent

Practical exercises

I. Perceptron

1. Considering the following linearly separable training data

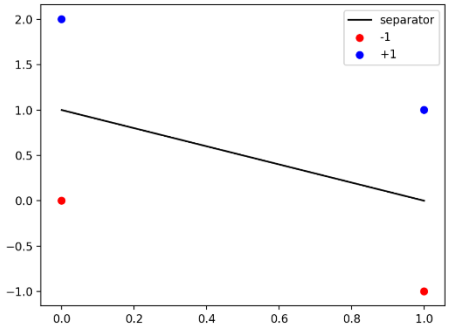
	y_1	y_2	y_3	output
x_1	0	0	0	−1
x_2	0	2	1	+1
x_3	1	1	1	+1
x_4	1	-1	0	−1

Given the perceptron learning algorithm with a learning rate of 1 for simplicity, sign activation, and all weights initialized to one (including the bias).

a) Considering y1 and y2, apply the algorithm until convergence.
Draw the separation hyperplane.

Considering the first observation, $o^1 = sign\,(\mathbf{w} \cdot \mathbf{x}) = +1$
as the perceptron made a mistake, we update the weights, $\mathbf{w}^{new} = [-1\ 1\ 1]^T$
The perceptron does not make mistakes for remaining observations.
If we do another epoch, weights do not change. Hence, we have convergence. □

$$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$$
$$x_2 = -x_1 + 1$$



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

Given with $\eta = 1$ and $\mathbf{w} = [1\ 1\ 1\ 1]^T$

$$o_1 = sign\,(\mathbf{w}^T\mathbf{x}) = +1$$
$$\mathbf{w}^{new} = (1\ 1\ 1\ 1)^T + 1(-1\ -1)(1\ 0\ 0\ 0)^T = [-1\ 1\ 1\ 1]^T$$

We can then repeat the same logic for the next data points with the updated $\mathbf{w}$.
$o_2 = +1$, as no mistake was made, we move to the next point.
No further mistakes are made for o_3 and o_4, concluding the first epoch.
Weights do not change for any observations after this epoch, thus the perceptron converged.

c) Identify the perceptron output for $\mathbf{x}_{new} = [0\ 0\ 1]^T$

$$o = sign\ (\mathbf{w}^T\mathbf{x}) = sign\ (-1 \cdot 1 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 1) = sign(0)$$

The point is on the boundary! Given the defined the sign function, $\mathbf{x}_{new}$ is labelled as +1.

d) What happens if we replace the sign function by the step function?

$$\Theta\ (x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Specifically, how would you change the learning rate to get the same results?

$$\mathbf{w} = \mathbf{w} + \eta\ (t - o)\ \mathbf{x}$$

The only term that depends on the perceptron output is the error $(y - o)$.

$$\delta_{sign} = \mathrm{t} - o_{sign} = \mathrm{t} - sign(\mathbf{w} \cdot \mathbf{x})$$
$$\Leftrightarrow \delta_{sign} = \begin{cases} +2 & t = +1, \mathbf{w} \cdot \mathbf{x} < 0 \\ -2 & t = -1, \mathbf{w} \cdot \mathbf{x} \geq 0 \end{cases}$$

$$\delta_{step} = \mathrm{t} - o_{step} = \mathrm{t} - \Theta(\mathbf{w} \cdot \mathbf{x})$$
$$\Leftrightarrow \delta_{step} = \begin{cases} +1 & t = +1, \mathbf{w} \cdot \mathbf{x} < 0 \\ -1 & t = -1, \mathbf{w} \cdot \mathbf{x} \geq 0 \end{cases}$$

There is a relation between the two error terms $\delta_{sign} = 2\delta_{step}$.

$$\mathbf{w} = \mathbf{w} + \eta_{sign}\delta_{sign}\ \mathbf{x} = \mathbf{w} + 2\ \eta_{sign}\delta_{step}\ \mathbf{x}$$

To get exactly the same update, we need to define a learning rate as $\eta_{step} = 2\eta_{sign}$,

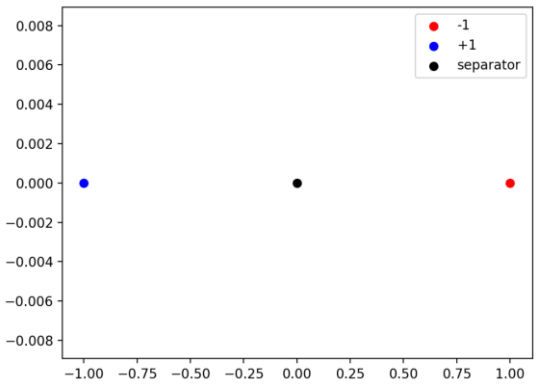
$$\mathbf{w} = \mathbf{w} + \eta_{step}\delta_{step}\ \mathbf{x} \quad \square$$

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

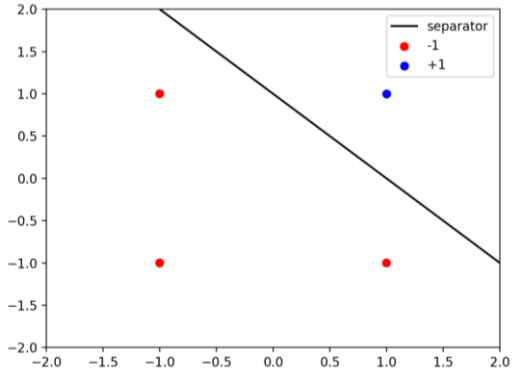
a) can learn the following logical functions: NOT, AND and OR

b) cannot learn the logical XOR function for two inputs

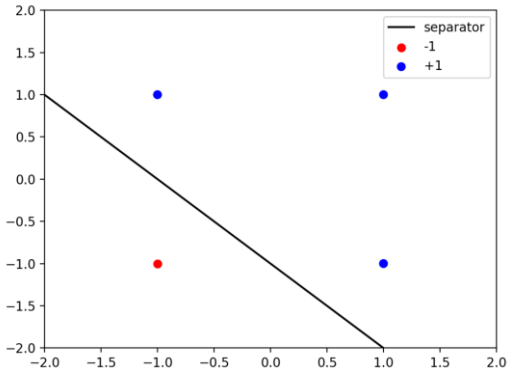
NOT: $\mathbf{w} = (0\ -1)^T$



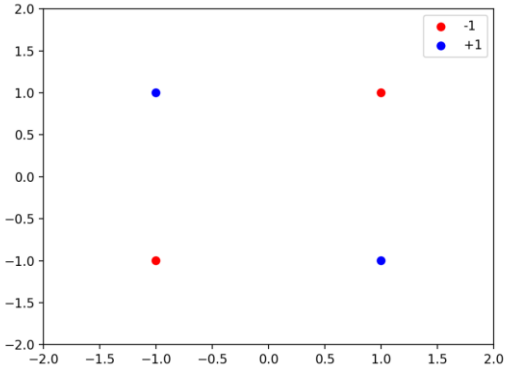
AND: $\mathbf{w} = (-1\ 1\ 1)^T$



OR: $\mathbf{w} = (1\ 1\ 1)^T$



XOR



II. Gradient descent learning

Considering the following training data

	y_1	y_2	outpu
x_1	1	1	1
x_2	2	1	1
x_3	1	3	0
x_4	3	3	0

3. Let us consider the following activation

$$\hat{z} = output(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{1 + \exp(-2\boldsymbol{w} \cdot \boldsymbol{x})}$$

and half sum of squared errors as the loss function

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{k=1}^{N} (z_k - \hat{z}_k)^2 \qquad where \;\; \hat{z}_k = output(\boldsymbol{x}_k, \boldsymbol{w})$$

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

To apply gradient descent, we want an update rule that moves a step of size η towards the opposite direction from the gradient of the error function with respect to the weights:

$$\mathbf{w}' = \mathbf{w} - \eta \frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}}$$

To find the learning rule, we must compute the gradient.
Before doing so, we should notice that the model is computing a sigmoid function, so:

$$output(\mathbf{x}, \mathbf{w}) \;=\; \frac{1}{1 + exp(-2\mathbf{w} \cdot \mathbf{x})} = \sigma(2\mathbf{w} \cdot \mathbf{x})$$

which has a well-known derivative that we will use later:

$$\frac{\partial \sigma(x)}{\partial x} = \frac{\partial \dfrac{1}{1 + exp(-x)}}{\partial x} = \sigma(x)(1 - \sigma(x))$$

Having made this auxiliar computation, it is now easier to compute the derivative of the error function with respect to the parameter vector:

$$\frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial \frac{1}{2}\sum_{k=1}^{N}(z_\mathrm{k} - output(\mathbf{x}_k, \mathbf{w}))^2}{\partial \mathbf{w}} = \frac{\partial \frac{1}{2}\sum_{k=1}^{N}(z_\mathrm{k} - \sigma(2\mathbf{w} \cdot \mathbf{x}_k))^2}{\partial \mathbf{w}}$$
$$= -2\sum_{k=1}^{N}((z_\mathrm{k} - \sigma(2\mathbf{w} \cdot \mathbf{x}_k))(\sigma(2\mathbf{w} \cdot \mathbf{x}_k)(1 - \sigma(2\mathbf{w} \cdot \mathbf{x}_k)))\mathbf{x}_k)$$

So, we can write our update rule as follows:

$$\mathbf{w}' = \mathbf{w} - \eta \frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w} + 2\eta \sum_{k=1}^{N}((z_\mathrm{k} - \sigma(2\mathbf{w} \cdot \mathbf{x}_k))(\sigma(2\mathbf{w} \cdot \mathbf{x}_k)(1 - \sigma(2\mathbf{w} \cdot \mathbf{x}_k)))\mathbf{x}_k)$$

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

The original gradient descent does one update per epoch because its learning rule requires contributions from all data points to do one step. Let us compute it.

According to the problem statement, we start with weights $\mathbf{w} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and learning rate $\eta = 1.0$

$$\mathbf{w}' = \mathbf{w} + 2\eta \sum_{k=1}^{4} \left(\left(z_k - \sigma(2\mathbf{w}\cdot\mathbf{x_k})\right) \left(\sigma(2\mathbf{w}\cdot\mathbf{x_k})(1-\sigma(2\mathbf{w}\cdot\mathbf{x_k}))\right)\mathbf{x_k} \right)$$

$$= \begin{pmatrix}1\\1\\1\end{pmatrix} + 2\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\right)\left(\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\right)\right)\begin{pmatrix}1\\1\\1\end{pmatrix}$$

$$+ 2\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\2\\1\end{pmatrix}\right)\right)\left(\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\2\\1\end{pmatrix}\right)\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\2\\1\end{pmatrix}\right)\right)\right)\begin{pmatrix}1\\2\\1\end{pmatrix}$$

$$+2\left(0-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\3\end{pmatrix}\right)\right)\left(\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\3\end{pmatrix}\right)\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\3\end{pmatrix}\right)\right)\right)\begin{pmatrix}1\\1\\3\end{pmatrix}$$

$$+ 2\left(0-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\3\\3\end{pmatrix}\right)\right)\left(\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\3\\3\end{pmatrix}\right)\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\3\\3\end{pmatrix}\right)\right)\right)\begin{pmatrix}1\\3\\3\end{pmatrix} = \begin{pmatrix}0.99991997\\0.99991687\\0.99973507\end{pmatrix}$$

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

In stochastic gradient descent we make one update for each training example.
So, instead of summing across all data points we adapt the learning rule for one example only:

$$\mathbf{w}' = \mathbf{w} + 2\eta((z_k - \sigma(2\mathbf{w}\cdot\mathbf{x_k}))\ (\sigma(2\mathbf{w}\cdot\mathbf{x_k})(1-\sigma(2\mathbf{w}\cdot\mathbf{x_k})))\mathbf{x_k})$$

We can now do the updates. For the first observation:

$$\mathbf{w}' = \begin{pmatrix}1\\1\\1\end{pmatrix} + 2\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\right)\left(\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\left(1-\sigma\left(2\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\right)\right)\begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}1.0000122\\1.0000122\\1.0000122\end{pmatrix}$$

4. Let us consider the following function:

$$output(\boldsymbol{x},\boldsymbol{w}) = \frac{1}{1+\exp(-\mathbf{w}\cdot\mathbf{x})}$$

and the cross-entropy loss function

$$E(\boldsymbol{w}) = -log(p(\boldsymbol{z}|\boldsymbol{w})) = -\sum_{k=1}^{N}(z_k log(\hat{z}_k) + (1-z_k)log(1-\hat{z}_k))$$

a) Determine the gradient descent learning rule for this unit

To apply gradient descent, we want an update rule that moves a step of size η towards the opposite direction from the gradient of the error function with respect to the weights:

$$\mathbf{w}' = \mathbf{w} - \eta\frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}}$$

$$\frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial(-\sum_{k=1}^{N} z_k\ log\ output(\mathbf{x_k},\mathbf{w}) + (1-z_k)\log(1-output(\mathbf{x_k},\mathbf{w})))}{\partial \mathbf{w}} = -\sum_{k=1}^{N}\mathbf{x_k}(z_k - \sigma(\mathbf{w}\cdot\mathbf{x_k}))$$

So, we can write our update rule as follows:

$$\mathbf{w}' = \mathbf{w} - \eta\frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w} + \eta\sum_{k=1}^{N}\mathbf{x_k}(z_k - \sigma(\mathbf{w}\cdot\mathbf{x_k}))$$

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

According to the problem statement, we start with weights $\mathbf{w} = \begin{pmatrix}1\\1\\1\end{pmatrix}$ and learning rate $\eta = 1.0$

$$\mathbf{w}' = \mathbf{w} + \eta \sum_{k=1}^{N} \mathbf{x_k}(z_k - \sigma(\mathbf{w}\cdot\mathbf{x_k})) = \begin{pmatrix}1\\1\\1\end{pmatrix} + \begin{pmatrix}1\\1\\1\end{pmatrix}\left(1 - \sigma\left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\right)$$

$$+(2)\left(1 - \sigma\left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\2\\1\end{pmatrix}\right)\right) + \begin{pmatrix}1\\1\\3\end{pmatrix}\left(0 - \sigma\left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\3\end{pmatrix}\right)\right) + \begin{pmatrix}1\\3\\3\end{pmatrix}\left(0 - \sigma\left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\3\\3\end{pmatrix}\right)\right) = \begin{pmatrix}-0.9269\\-2.9072\\-4.9118\end{pmatrix}$$

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

$$\mathbf{w}' = \mathbf{w} + \eta\mathbf{x_k}(z_k - \sigma(\mathbf{w}\cdot\mathbf{x_k}))$$

$$\mathbf{w}' = \mathbf{w} + \eta\mathbf{x_k}(z_k - \sigma(\mathbf{w}\cdot\mathbf{x_k})) = \begin{pmatrix}1\\1\\1\end{pmatrix} + \begin{pmatrix}1\\1\\1\end{pmatrix}\left(1 - \sigma\left(\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\right) = \begin{pmatrix}1\\1\\1\end{pmatrix} + \begin{pmatrix}1\\1\\1\end{pmatrix}(1 - \sigma(3)) = \begin{pmatrix}1.0474\\1.0474\\1.0474\end{pmatrix}$$

5. Let us consider the following function:

$$output(\mathbf{x}, \mathbf{w}) = exp((\mathbf{w}\cdot\mathbf{x})^2)$$

and half sum of squared errors as the loss function

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

$$\mathbf{w}' = \mathbf{w} - \eta\frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}}$$

$$\frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial \frac{1}{2}\sum_{k=1}^{N}(z_k - output(\mathbf{x_k}, \mathbf{w}))^2}{\partial \mathbf{w}} = -2\sum_{k=1}^{N}\left((z_k - \exp((\mathbf{w}\cdot\mathbf{x_k})^2))\exp((\mathbf{w}\cdot\mathbf{x_k})^2)(\mathbf{w}\cdot\mathbf{x_k})\mathbf{x_k}\right)$$

We can write our update rule as follows:

$$\mathbf{w}' = \mathbf{w} - \eta\frac{\partial \mathrm{E}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w} + 2\eta\sum_{k=1}^{N}\left((z_k - \exp((\mathbf{w}\cdot\mathbf{x_k})^2))exp((\mathbf{w}\cdot\mathbf{x_k})^2)(\mathbf{w}\cdot\mathbf{x_k})\mathbf{x_k}\right)$$

b) Compute the stochastic gradient descent update for input $\mathbf{x}_{new} = [1\ 1]^T$, $z_{new} = 0$ initialized with $\mathbf{w} = [0\ 1\ 0]^T$ and learning rate η=2

$$\mathbf{w}' = \mathbf{w} + 2\eta(z_k - \exp((\mathbf{w}\cdot\mathbf{x_k})^2))\exp((\mathbf{w}\cdot\mathbf{x_k})^2)(\mathbf{w}\cdot\mathbf{x_k})\mathbf{x_k}$$

For the first observation:

$$\mathbf{w}' = \begin{pmatrix}0\\1\\0\end{pmatrix} + 2(2)\left(0 - exp\left(\left(\begin{pmatrix}0\\1\\0\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)^2\right)\right)exp\left(\left(\begin{pmatrix}0\\1\\0\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)^2\right)\left(\begin{pmatrix}0\\1\\0\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}\right)\begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}-4e^2\\1 - 4e^2\\-4e^2\end{pmatrix}$$

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

Hints: analyse the properties of 3.a and 4.a update rules.