INF 264 - Correction of exercice 5

1 Gradient descent and backpropagation in a simple neural network

1. The parameters are initialized as $\theta = (w_0, w_1, w_2, w_3, w_4) = (1, 1, 0, 1, 0)$. We start with $\mathbf{x_1} = (2, 1)$:
   We have

$$\begin{aligned} z(\mathbf{x_1}) &= w_0 \cdot \mathbf{x_1}[0] + w_1 \cdot \mathbf{x_1}[1] + w_2 \cdot 1 \\ &= 1 \cdot 2 + 1 \cdot 1 + 0 \cdot 1 \\ &= 3, \end{aligned}$$

$$\begin{aligned} h(\mathbf{x_1}) &= \mathrm{ReLU}\big(z(\mathbf{x_1})\big) \\ &= \max(0, 3) \\ &= 3 \end{aligned}$$

   and

$$\begin{aligned} \widehat{y}(\mathbf{x_1}) &= w_3 \cdot h(\mathbf{x_1}) + w_4 \cdot 1 \\ &= 1 \cdot 3 + 0 \cdot 1 \\ &= 3. \end{aligned}$$

   Similarly with $\mathbf{x_2} = (-3, 2)$:
   We have

$$\begin{aligned} z(\mathbf{x_2}) &= w_0 \cdot \mathbf{x_2}[0] + w_1 \cdot \mathbf{x_2}[1] + w_2 \cdot 1 \\ &= 1 \cdot (-3) + 1 \cdot 2 + 0 \cdot 1 \\ &= -1, \end{aligned}$$

$$\begin{aligned} h(\mathbf{x_2}) &= \mathrm{ReLU}\big(z(\mathbf{x_2})\big) \\ &= \max(0, -1) \\ &= 0 \end{aligned}$$

   and

$$\begin{aligned} \widehat{y}(\mathbf{x_2}) &= w_3 \cdot h(\mathbf{x_2}) + w_4 \cdot 1 \\ &= 1 \cdot 0 + 0 \cdot 1 \\ &= 0. \end{aligned}$$

2. The quadratic losses on every sample in the dataset $\mathcal{D}$ are

$$\begin{aligned} L(\mathbf{x_1}, y_1) &= \big(\widehat{y}(\mathbf{x_1}) - y_1\big)^2 \\ &= (3 - 1.3)^2 \\ &= 1.7^2 \\ &= 2.89 \end{aligned}$$

   and

$$\begin{aligned} L(\mathbf{x_2}, y_2) &= \big(\widehat{y}(\mathbf{x_2}) - y_2\big)^2 \\ &= (0 - 1.9)^2 \\ &= 1.9^2 \\ &= 3.61, \end{aligned}$$

hence the mean-squared loss of the network on the dataset $\mathcal{D}$ is

$$\begin{aligned}\mathcal{L}(\mathcal{D}) &= \frac{L(\mathbf{x_1}, y_1) + L(\mathbf{x_2}, y_2)}{2} \\ &= 0.5 \cdot (2.89 + 3.61) \\ &= 0.5 \cdot 6.5 \\ &= 3.25.\end{aligned}$$

3. We prove the five backpropagation equalities. For the three first equalities, we notice that L is a function of z and that z is a function of w_i, $i \in \{0, 1, 2\}$. We thus can apply the chain rule:

$$\begin{aligned}\forall i \in \{0,1,2\}, \quad \frac{\partial L(\mathbf{x}, y)}{\partial w_i} &= \frac{\partial L(\mathbf{x}, y)}{\partial z} \cdot \frac{\partial z(\mathbf{x})}{\partial w_i} \\ &= \delta_h(\mathbf{x}, y) \cdot \frac{\partial\big(w_0 \cdot \mathbf{x}[0] + w_1 \cdot \mathbf{x}[1] + w_2 \cdot 1\big)}{\partial w_i} \\ &= \begin{cases} \delta_h(\mathbf{x}, y) \cdot \mathbf{x}[0] & \text{if } i = 0 \\ \delta_h(\mathbf{x}, y) \cdot \mathbf{x}[1] & \text{if } i = 1 \\ \delta_h(\mathbf{x}, y) & \text{if } i = 2. \end{cases}\end{aligned}$$

Similarly for the two last equalities, we notice that L is a function of $\widehat{y}$ and that $\widehat{y}$ is a function of w_i, $i \in \{3, 4\}$. We thus can apply the chain rule:

$$\begin{aligned}\forall i \in \{3,4\}, \quad \frac{\partial L(\mathbf{x}, y)}{\partial w_i} &= \frac{\partial L(\mathbf{x}, y)}{\partial \widehat{y}} \cdot \frac{\partial \widehat{y}(\mathbf{x})}{\partial w_i} \\ &= \delta_{\widehat{y}}(\mathbf{x}, y) \cdot \frac{\partial\big(w_3 \cdot h(\mathbf{x}) + w_4 \cdot 1\big)}{\partial w_i} \\ &= \begin{cases} \delta_{\widehat{y}}(\mathbf{x}, y) \cdot h(\mathbf{x}) & \text{if } i = 3 \\ \delta_{\widehat{y}}(\mathbf{x}, y) & \text{if } i = 4. \end{cases}\end{aligned}$$

4. We perform the backpropagation step of our network on the dataset $\mathcal{D}$. We start with $\mathbf{x_1} = (2, 1)$ and $y_1 = 1.3$:
We have

$$\begin{aligned}\delta_{\widehat{y}}(\mathbf{x_1}, y_1) &= \frac{\partial L(\mathbf{x_1}, y_1)}{\partial \widehat{y}} \\ &= 2 \cdot \big(\widehat{y}(\mathbf{x_1}) - y_1\big) \\ &= 2 \cdot (3 - 1.3) \\ &= 2 \cdot 1.7 \\ &= 3.4\end{aligned}$$

and

$$\begin{aligned}\delta_h(\mathbf{x_1}, y_1) &= \delta_{\widehat{y}}(\mathbf{x_1}, y_1) \cdot w_3 \cdot \mathrm{ReLU}'\big(z(\mathbf{x_1})\big) \\ &= 3.4 \cdot 1 \cdot \mathrm{ReLU}'(3) \\ &= 3.4 \cdot 1 \cdot 1 \\ &= 3.4,\end{aligned}$$

hence

$$\begin{aligned}\nabla_\theta L(\mathbf{x_1}, y_1) &= \Big(\tfrac{\partial L(\mathbf{x_1}, y_1)}{\partial w_0}, \tfrac{\partial L(\mathbf{x_1}, y_1)}{\partial w_1}, \tfrac{\partial L(\mathbf{x_1}, y_1)}{\partial w_2}, \tfrac{\partial L(\mathbf{x_1}, y_1)}{\partial w_3}, \tfrac{\partial L(\mathbf{x_1}, y_1)}{\partial w_4}\Big) \\ &= \big(\delta_h(\mathbf{x_1}, y_1) \cdot \mathbf{x_1}[0], \delta_h(\mathbf{x_1}, y_1) \cdot \mathbf{x_1}[1], \delta_h(\mathbf{x_1}, y_1), \delta_{\widehat{y}}(\mathbf{x_1}, y_1) \cdot h(\mathbf{x_1}), \delta_{\widehat{y}}(\mathbf{x_1}, y_1)\big) \\ &= (3.4 \cdot 2, 3.4 \cdot 1, 3.4, 3.4 \cdot 3, 3.4) \\ &= (6.8, 3.4, 3.4, 10.2, 3.4).\end{aligned}$$

Similarly with $\mathbf{x_2} = (-3, 2)$ and $y_2 = 1.9$:
We have

$$\begin{aligned}\delta_{\widehat{y}}(\mathbf{x_2}, y_2) &= \frac{\partial L(\mathbf{x_2}, y_2)}{\partial \widehat{y}} \\ &= 2 \cdot \big(\widehat{y}(\mathbf{x_2}) - y_2\big) \\ &= 2 \cdot (0 - 1.9) \\ &= 2 \cdot (-1.9) \\ &= -3.8\end{aligned}$$

and

$$\begin{aligned}\delta_h(\mathbf{x_2}, y_2) &= \delta_{\widehat{y}}(\mathbf{x_2}, y_2) \cdot w_3 \cdot \mathrm{ReLU}'\big(z(\mathbf{x_2})\big) \\ &= -3.8 \cdot 1 \cdot \mathrm{ReLU}'(-1) \\ &= -3.8 \cdot 1 \cdot 0 \\ &= 0,\end{aligned}$$

hence

$$\begin{aligned}\nabla_\theta L(\mathbf{x_2}, y_2) &= \Big(\tfrac{\partial L(\mathbf{x_2},y_2)}{\partial w_0}, \tfrac{\partial L(\mathbf{x_2},y_2)}{\partial w_1}, \tfrac{\partial L(\mathbf{x_2},y_2)}{\partial w_2}, \tfrac{\partial L(\mathbf{x_2},y_2)}{\partial w_3}, \tfrac{\partial L(\mathbf{x_2},y_2)}{\partial w_4}\Big) \\ &= \big(\delta_h(\mathbf{x_2}, y_2) \cdot \mathbf{x_2}[0], \delta_h(\mathbf{x_2}, y_2) \cdot \mathbf{x_2}[1], \delta_h(\mathbf{x_2}, y_2), \delta_{\widehat{y}}(\mathbf{x_2}, y_2) \cdot h(\mathbf{x_2}), \delta_{\widehat{y}}(\mathbf{x_2}, y_2)\big) \\ &= \big(0 \cdot (-3), 0 \cdot 2, 0, -3.8 \cdot 0, -3.8\big) \\ &= (0, 0, 0, 0, -3.8).\end{aligned}$$

The previous results combined give us the gradient of the network's loss function on the dataset $\mathcal{D}$:

$$\begin{aligned}\nabla_\theta \mathcal{L}(\mathcal{D}) &= \frac{\nabla_\theta L(\mathbf{x_1}, y_1) + \nabla_\theta L(\mathbf{x_2}, y_2)}{2} \\ &= 0.5 \cdot \big[(6.8, 3.4, 3.4, 10.2, 3.4) + (0, 0, 0, 0, -3.8)\big] \\ &= 0.5 \cdot (6.8, 3.4, 3.4, 10.2, -0.4) \\ &= (3.4, 1.7, 1.7, 5.1, -0.2).\end{aligned}$$

5. Assuming a learning rate $\eta = 0.01$, the gradient descent update gives:

$$\begin{aligned}\theta &\leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}(\mathcal{D}) \\ &\leftarrow (1, 1, 0, 1, 0) - 0.01 \cdot (3.4, 1.7, 1.7, 5.1, -0.2) \\ &\leftarrow (1, 1, 0, 1, 0) - (0.034, 0.017, 0.017, 0.051, -0.002) \\ &\leftarrow (0.966, 0.983, -0.017, 0.949, 0.002).\end{aligned}$$

6. The parameters were updated and are now $\theta = (w_0, w_1, w_2, w_3, w_4) = (0.966, 0.983, -0.017, 0.949, 0.002)$.
We start with $\mathbf{x_1} = (2, 1)$:
We have

$$\begin{aligned}z(\mathbf{x_1}) &= w_0 \cdot \mathbf{x_1}[0] + w_1 \cdot \mathbf{x_1}[1] + w_2 \cdot 1 \\ &= 0.966 \cdot 2 + 0.983 \cdot 1 - 0.017 \cdot 1 \\ &= 2.898,\end{aligned}$$

$$\begin{aligned}h(\mathbf{x_1}) &= \mathrm{ReLU}\big(z(\mathbf{x_1})\big) \\ &= \max(0, 2.898) \\ &= 2.898\end{aligned}$$

and

$$\begin{aligned}\widehat{y}(\mathbf{x_1}) &= w_3 \cdot h(\mathbf{x_1}) + w_4 \cdot 1\\ &= 0.949 \cdot 2.898 + 0.002 \cdot 1\\ &= 2.752.\end{aligned}$$

Similarly with $\mathbf{x_2} = (-3, 2)$:
We have

$$\begin{aligned}z(\mathbf{x_2}) &= w_0 \cdot \mathbf{x_2}[0] + w_1 \cdot \mathbf{x_2}[1] + w_2 \cdot 1\\ &= 0.966 \cdot (-3) + 0.983 \cdot 2 - 0.017 \cdot 1\\ &= -0.949,\end{aligned}$$

$$\begin{aligned}h(\mathbf{x_2}) &= \text{ReLU}\big(z(\mathbf{x_2})\big)\\ &= \max(0, -0.949)\\ &= 0\end{aligned}$$

and

$$\begin{aligned}\widehat{y}(\mathbf{x_2}) &= w_3 \cdot h(\mathbf{x_2}) + w_4 \cdot 1\\ &= 0.949 \cdot 0 + 0.002 \cdot 1\\ &= 0.002.\end{aligned}$$

The quadratic losses on every sample in the dataset $\mathcal{D}$ now are

$$\begin{aligned}L(\mathbf{x_1}, y_1) &= \big(\widehat{y}(\mathbf{x_1}) - y_1\big)^2\\ &= (2.752 - 1.3)^2\\ &= 1.452^2\\ &= 2.108\end{aligned}$$

and

$$\begin{aligned}L(\mathbf{x_2}, y_2) &= \big(\widehat{y}(\mathbf{x_2}) - y_2\big)^2\\ &= (0.002 - 1.9)^2\\ &= 1.898^2\\ &= 3.602,\end{aligned}$$

hence the mean-squared loss of the network on the dataset $\mathcal{D}$ reduced to

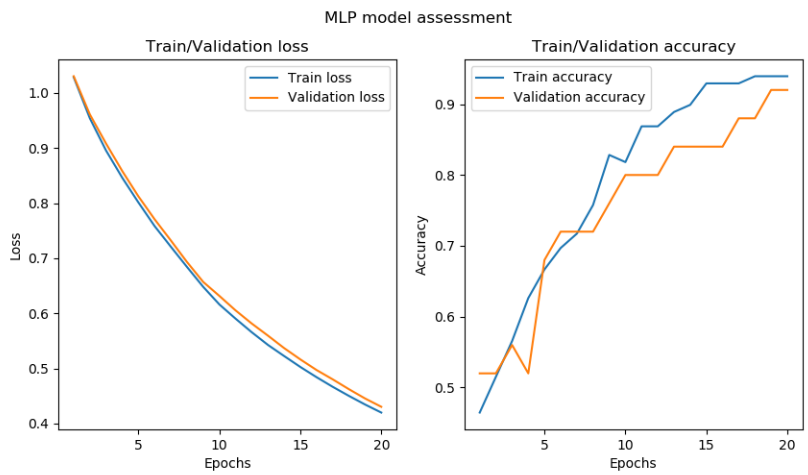
$$\begin{aligned}\mathcal{L}(\mathcal{D}) &= \frac{L(\mathbf{x_1}, y_1) + L(\mathbf{x_2}, y_2)}{2}\\ &= 0.5 \cdot (2.108 + 3.602)\\ &= 0.5 \cdot 5.71\\ &= 2.855.\end{aligned}$$

7. The learning rate hyper-parameter tunes the strength of the gradient descent updates. The larger the learning rate, the higher the amplitude of the gradient correction term $\eta \cdot \nabla_\theta \mathcal{L}$. This means that when the learning rate is small, the parameters will require many updates to change significantly. On the contrary, with a high learning rate each update will change the parameters considerably.

   In practice, a learning rate policy is usually implemented such that the learning rate decreases the more the neural network gets trained. The reason for this is that when training a neural network, we seek to modify this network's parameters so as to minimize the loss function of the network; at the beginning of the optimization process, we are far from a good local extremum, thus we can apply strong gradient corrections to the parameters at each update in order to quickly converge to a good local extremum. As we get closer and closer to such a good local extremum, we make sure to decrease the learning rate so that the parameters have a smaller and smaller amplitude of oscillation around the local extremum: we hope that the network's parameters will be trapped in a tight neighborhood of the good local extremum.

2 Neural networks in Keras

5. Results of our MLP model's training:



Our MLP model displays more than 90% train and validation accuracy after 20 epochs (no underfitting). The test accuracy is slightly lower with [], which may indicate that our model overfits a little bit.

6. We perform model selection of our MLP on the following hyper-parameters:

- Activations used in every hidden layer: 'activation type' $\in$ {'sigmoid', 'relu'}
- Decay factor of the learning rate every n epochs: 'decay factor' $\in$ {0.75, 0.25}
- Momentum value in SGD with momentum: 'momentum' $\in$ {0.1, 0.9}
- Use the Nesterov momentum instead of the classical momentum in SGD: 'Nesterov' $\in$ {False, True}
- Number of hidden dense layers in the MLP: 'network depth' $\in$ {1, 2}.

Every model was trained for 5 epochs, with a batch size of 16 samples, a base learning rate of 0.01 and a number of epochs between each learning rate decay equal to 2.

The best MLP model was obtained with the hyper-parameters instance {'activation type'='relu', 'decay factor'= 0.75, 'momentum'= 0.9, 'Nesterov'=True, 'network depth'= 1}, and this model achieved a test accuracy of 0.777 (after only 5 epochs). We can notice that our models with 2 hidden layers and with a smaller decay factor tended to underfit more.