TDT4171 - Assignment 7
Artificial Neural Network

Arthur Testard - id: 105022

1 Feedforward Neural Network

The purpose of this assignment is to implement the neural network given by Figure 22.3 of the textbook [RN10] or also given by Figure 1.

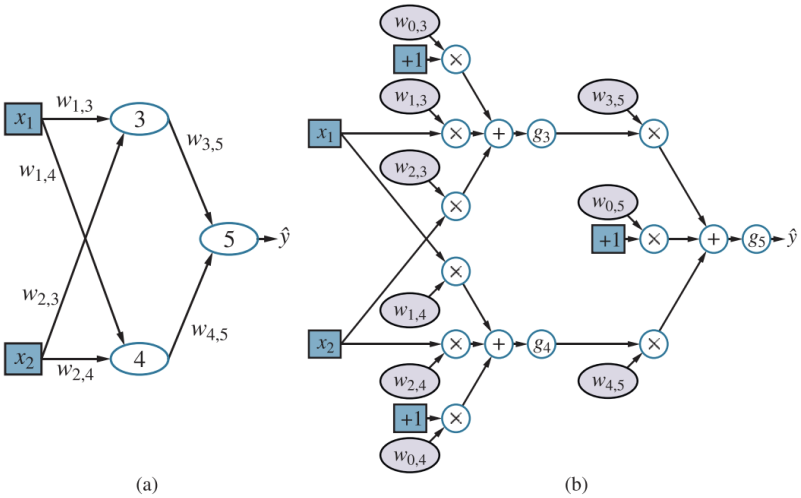


Figure 1: (a) A neural network with two inputs, one hidden layer of two units, and one output unit. Not shown are the dummy inputs and their associated weights. (b) The network in (a) unpacked into its full computation graph.

The implementation is given on `assignment_7.py`. It mainly concerns two functions: `forward_pass` and `backward_pass`. `forward_pass` is simply the computation of the forward process. Using the notations of the textbook it concerns computing for each nodes, 3, 4 and 5, the forward process:

$$y_3 = g_3\left(\sum_{i=0}^{2} w_{i,3}x_{i,3}\right) = sigmoid\left(\sum_{i=0}^{2} w_{i,3}x_{i,3}\right)$$
$$y_4 = g_4\left(\sum_{i=0}^{2} w_{i,4}x_{i,4}\right) = sigmoid\left(\sum_{i=0}^{2} w_{i,4}x_{i,4}\right)$$
$$y_5 = g_5\left(\sum_{i\in\{0,3,5\}} w_{i,3}x_{i,5}\right) = tanh\left(\sum_{i\in\{0,3,5\}} w_{i,5}x_{i,5}\right)$$

If we consider the convention:

$$\forall i \in \{3,4,5\},\ x_{0,i} = 1$$

The `backward_pass` is about the gradient descent, moreover, it's about computing the loss from a specific weight $w_{i,j}$. The loss is the mean squared error:

$$\mathcal{L} = \sum_{d\in\mathcal{D}} (t_d - o_d)^2$$

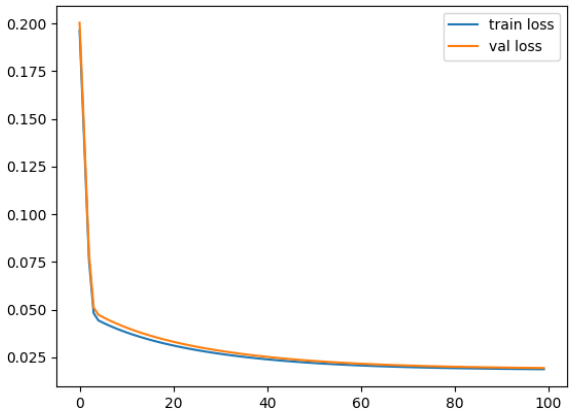
We can apply the chain rule to get the derivative of $\mathcal{L}$ for each $w_{i,j}$.

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_{i,5}} &= \frac{\partial \mathcal{L}}{\partial y_5} \times \frac{\partial y_5}{\partial w_{i,5}} \\
&= -(t - y_5) \times (1 - y_5^2) x_{i,5} \\
\frac{\partial \mathcal{L}}{\partial w_{i,4}} &= \frac{\partial \mathcal{L}}{\partial y_5} \times \frac{\partial y_5}{\partial x_5} \times \frac{\partial x_5}{\partial w_{i,4}} \\
&= -(t - y_5) \times (1 - y_5^2) w_5 \times y_4(1 - y_4) x_{i,4} \\
\frac{\partial \mathcal{L}}{\partial w_{i,3}} &= \frac{\partial \mathcal{L}}{\partial y_5} \times \frac{\partial y_5}{\partial x_5} \times \frac{\partial x_5}{\partial w_3} \\
&= -(t - y_5) \times (1 - y_5^2) w_5 \times y_3(1 - y_3) x_{i,3}
\end{aligned}
$$

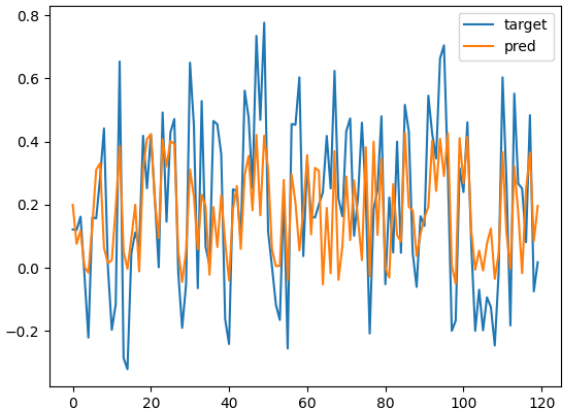
And then, we can update the parameters, for all i, j:

$$
w_{i,j} := w_{i,j} - \eta \frac{\partial \mathcal{L}}{\partial w_{i,j}}, \text{ with } \eta \text{ the learning rate}
$$

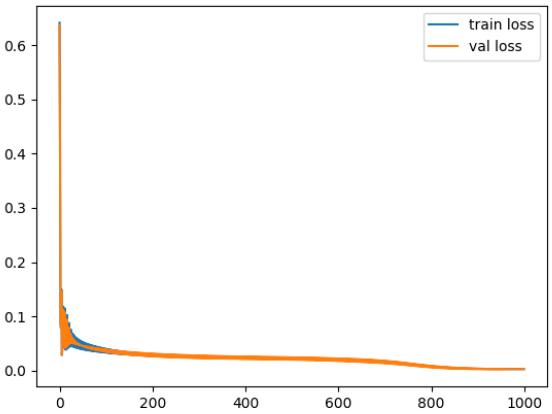
Now to train our model, we have to choose the way we initialize the parameters. I decided to use Xavier Glorot's uniform method for nodes 3 and 4 and I used the same value for node 5 because I was lazy. Xavier Glorot's uniform method consists in give for each parameters the a random value (uniformly) in the range $\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}\right)$. Now we have to choose the hyperparameters: the learning rate η and number of epochs on which we train for. I thus proposed the code given on `assignment_7.py` to get the best learning rate in the range given of learning rates for a given epoch. We can notice that for a different number of epoch, the learning rate which give the lowest value on the test set is not always the same. Figure 2 shows that point and resumes some cases of training.



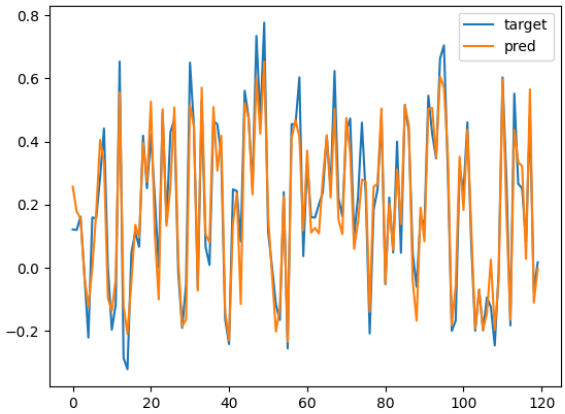
(a) Train and validation set loss on 100 epochs with $\eta = 0.7$, test loss ≈ 0.0193



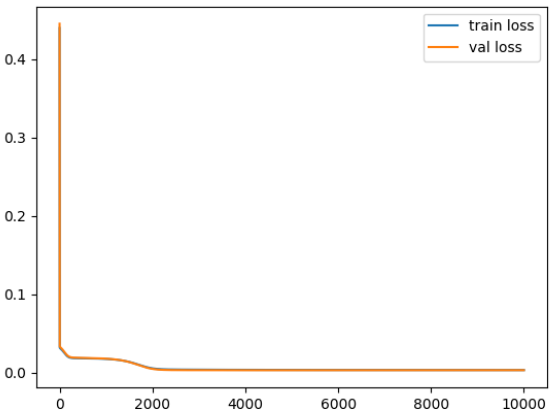
(b) Target and prediction plots for a training of 100 epochs on $\eta = 0.7$ on test set



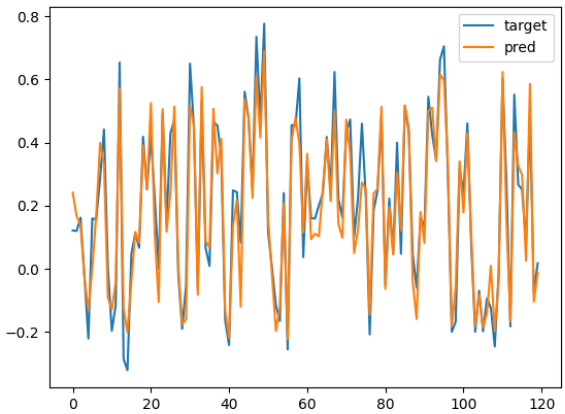
(c) Train and validation set loss on 1000 epochs with $\eta = 2.0$, test loss ≈ 0.0030



(d) Target and prediction plots for a training of 1000 epochs on $\eta = 2.0$ on test set



(e) Train and validation set loss on 10 000 epochs with $\eta = 0.5$, test loss ≈ 0.0029



(f) Target and prediction plots for a training of 10 000 epochs on $\eta = 0.5$ on test set

Figure 2: Three different training loss and predictions of models trained with different number of epochs with the best learning rate on each number of epoch

References

[RN10] Stuart J Russell and Peter Norvig. *Artificial intelligence a modern approach*. London, 2010.