Artificial Intelligence

Tutorial week 2 – Neural networks

COMP9414

1 Theoretical Background

A feedforward network, with n inputs, m outputs and a hidden layer with p elements, might be mathematically modelled as:

$$Y(t) = Wo^T f_{NL}(Wi^T X(t)) \tag{1}$$

Error of the network:

$$E(t) = Y(t) - T(t) \tag{2}$$

With: $T(t)$ Targets.

The backpropagation algorithm allows iteratively updating the weights of the network, backpropagating the mean squared error, i.e.:

$$mse = \frac{1}{s}\sum_{k=1}^{m}\sum_{t=1}^{s} e_k^2(t) \tag{3}$$

With: s total number of samples.
e_k k-th component of the vector $E(t)$.

This value (mse) is the objective function to minimise. Then, the gradient is computed respect to each of the network parameters, obtaining:

$$\Delta Wo = -\alpha * \frac{2}{s} * E(t) * Z^T(t) \tag{4}$$

$$\Delta Wi^T = -\alpha * \frac{2}{s} * E^T(t) * Wo_j * \frac{\partial f_{NL}}{\partial Zin} * X^T(t) \tag{5}$$

With: α learning rate.

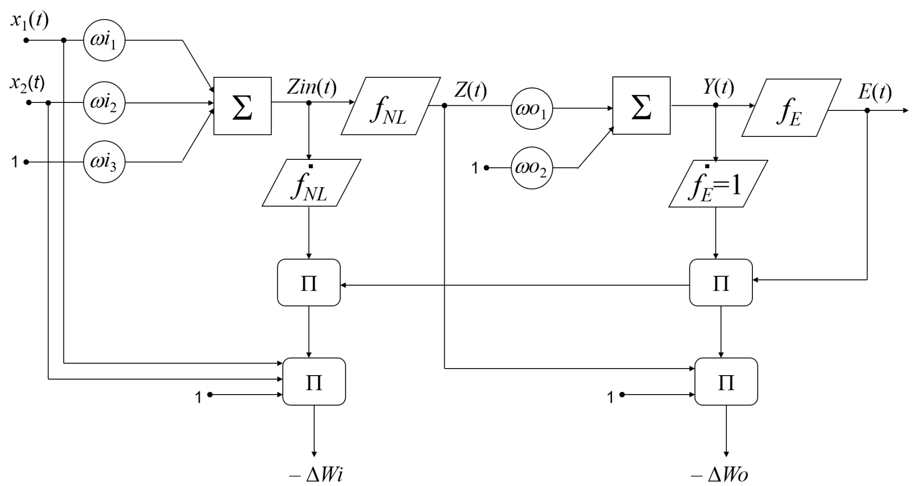

Figure 1: Multi-layer perceptron of architecture $2 \times 1 \times 1$. Delta values computed with the backpropagation algorithm.

2 Setup

(a) Using Python, create a feed-forward neural network with one hidden layer (one unit in each layer – see Fig. 1).

(b) Input and target vectors:
Input=[-2 -1 0 1; -1 0 1 2];
Target=[-1.5 -1 1 1.5]

(c) Initialise the network parameters (weights) as follows: $wi_1 = 0.0651$, $wi_2 = -0.6970$, $wo = -0.1342$, $bi = 0$, $bo = -0.5481$.

(d) Compute the delta values for each of the network parameters considering Eq. (4) and Eq. (5) and as shown in Fig. 1. As a non-linear function, use the hyperbolic tangent:

$$f_{NL}(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{6}$$

(e) To train the network, use a learning rate $\alpha = 0.01$ during 500 epochs.

3 Experiments

(a) Plot the input and target vectors. Show both variables in the same figure.

(b) Propagate the input through the network to obtain the output $Y(t)$. Plot the network output and the target vector in the same figure to compare them.

(c) Train the network. Once the network has been trained, plot the mean squared error for each epoch. Use a semi-logarithmic scale at the y-axis.

(d) Once again plot the outputs and the target vector using the network trained instead to compare the results.

(e) Introduce variations to the learning rate and number of training epochs. Observe and comment the obtained results. Some suggested tests are:

- Keep the number of epochs fixed at 500, then vary the learning rate to $\alpha = 0.1$ and $\alpha = 0.001$.
- Keep the learning rate fixed at $\alpha = 0.01$, then vary the number of epochs to 50 and 1000.

(f) Vary the initial value of the weights to the following set: $wi_1 = 0$, $wi_2 = 0$, $wo = 1$, $bi = 1$, $bo = 1$. The training is highly dependent of the initial solution, why?

(g) Currently, there are libraries that simplify the process of creating and training neural networks. TensorFlow and Keras are among the most popular ones. Can you propose a simplified version of the previous experiment using TF and Keras? Consider using Stochastic Gradient Descent (SGD) as the optimizer. What is the difference with Gradient Descent?