

Stavanger, December 17, 2024

Theoretical exercise 4

ELE520 Machine learning

Problem 1

Show that if the transfer function of the hidden units is linear, a two-layer network is equivalent to a one-layer network. (Hint: Express the forward computations and from input to output and see what happens when the nonlinear functions are replaced with linear functions like in the input layer. Use matrix notation with input vector \mathbf{x} , layer outputs \mathbf{y}^r and weight matrices $\Theta^r, r = 1, 2$.)

Problem 2

We want to design a two layer neural net to discriminate between the characters A, P, C, F which corresponds to the classes $\omega_1, \omega_2, \omega_3, \omega_4$ respectively. The characters are characterised by the number of corners, holes and symmetrical properties.

The training vectors we shall use is given as $\mathbf{x}_1 = (4 \ 1 \ 1)^T$, $\mathbf{x}_2 = (4 \ 1 \ 0)^T$, $\mathbf{x}_3 = (2 \ 0 \ 1)^T$ and $\mathbf{x}_4 = (2 \ 0 \ 0)^T$ where \mathbf{x}_i are the training vector for class ω_i .

The corresponding target vectors are $\mathbf{y}_1 = (1 \ 0 \ 0 \ 0)^T$, $\mathbf{y}_2 = (0 \ 1 \ 0 \ 0)^T$, $\mathbf{y}_3 = (0 \ 0 \ 1 \ 0)^T$ and $\mathbf{y}_4 = (0 \ 0 \ 0 \ 1)^T$.

The net is configured with number of inputs, hidden nodes and outputs according to $l = 3$, $n_H = 3$ and $M = 4$. In addition we will use a bias for each layer. The net is initialised with arbitrary values for the weights. An unlinear sigmmod is used which is of the type $f(z^r) = \frac{1}{1 + \exp^{-z^r}}$.

As you will see, this network introduces bias units. You will need to accomodate the learning rules to this.¹

- a) Draw the network when the arbitrary first layer weights is given by $\theta_1^1 = (0.0 \ -0.5 \ 0.5)^T$, $\theta_2^1 = (0.5 \ -0.5 \ 0.0)^T$, $\theta_3^1 = (-0.5 \ 0.0 \ 0.5)^T$ where θ_j^1 are the weights connected to the hidden node number j . The bias weights for the hidden nodes are 0.5, -0.5 and 0.5.

¹The learning rules presented in the textbook includes bias, but is a bit more involved as it is multilayered and uses indexing to a greater extent than what has been presented in the lectures.

Correspondingly, the second layer weights are given by $\theta_1^2 = (0.5 \ -0.5 \ 0.5)^T$, $\theta_2^2 = (0.0 \ -0.5 \ 0.5)^T$, $\theta_3^2 = (-0.5 \ 0.5 \ 0.0)^T$ and $\theta_4^2 = (0.5 \ 0.0 \ -0.5)^T$ where θ_k^2 are the weights connected to output node number k . The bias weights to the output nodes are -0.5 0.5, -0.5 and 0.5.

- b) Normalise the feature vectors in the training set according to $\mathbf{x} = \mathbf{x}/4$, so that $0 \leq x_i \leq 1$.
- c) Do one *forward computation* and compute $J(\theta)$ for the normalised training vector \mathbf{x}_1 .
- d) Do one *backward computation* with learning rate $\mu = 1$ based on the result from the previous subtask to update the weights in the net.