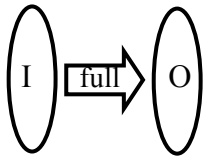
CMSC 727 - Assignment 2: More Basic Supervised Learning – Spring 2017

Do the following problems by hand (using a calculator is OK). Turn in a hardcopy only of your answers.

1. Consider a single-layer, fully-connected feedforward neural network like that pictured at the right, where each input unit in I is connected to each and every output node in O, and w_{ji} designates the weight on the connection to output node j from input node i. The real-valued input and activation level of each output node j are given by $in_{Oj} = \sum_{l \epsilon I} w_{jl} a_{Il} + \sum_{l,k \epsilon I} w_{jlk} a_{Il} a_{Ik}$ where w_{jl} and w_{jlk} are adaptable weights, and $a_{Oj} = (1 + e^{-s \cdot in_{Oj}})^{-1}$ respectively, where s > 0 is a fixed global parameter. Weights w_{jlk} are said to be "higher-order weights" because they modulate products of activation levels.



a. Define an "error measure" for the p^{th} input-output pair by $E^p = \sum_{k \in O} (a_{Ok}^2 - 2a_{Ok}c_{Ok} - (1 - c_{Ok})(1 + c_{Ok}))$ and the superscript p is mostly omitted for simplicity. Here c_{Ok} is the correct/target output value for output node k when the p^{th} input pattern is present. An iterative gradient descent procedure is to be used during network training to minimize E^p. Use the gradient descent method with the given E^p to derive and state separate learning rules ($\Delta w_{jl} = \ldots$ and $\Delta w_{jlk} = \ldots$) for how weights w_{jl} and w_{jlk} should change during training after the network sees the p^{th} training sample. Weight changes Δw_{ji} and Δw_{jlk} should be expressed solely in terms of activation levels such as a_{Oj} , a_{Ii} , output node input values such as in_{Oj}, target values c_{Oj}, and/or weights such as w_{ji} and w_{jlk}, and constant parameters.

b. Could a network defined as in this problem, having two input nodes, a single bias node, no hidden nodes, and one output node, learn to perform an exclusive-OR function of the inputs? Assume input node activation values are 0 or 1 only, and that output node values are counted as correct if they are within ±0.1 of the correct value. Justify your answer.

2. Consider a single-layer, fully-connected, feedforward-only neural network (like that pictured in the preceding problem), with 2 input nodes in I and 2 output nodes in O, but now the output node activations are linear functions of the inputs: $\vec{a}_O = W \vec{a}_I$, making this a linear associative memory. We want to train this network to produce these two associations:

$$\vec{a}_I^1 = \begin{bmatrix} .6 \\ .8 \end{bmatrix} \text{ gives } \vec{a}_O^1 = \begin{bmatrix} 100 \\ -100 \end{bmatrix} \qquad \text{and} \qquad \vec{a}_I^2 = \begin{bmatrix} -.8 \\ .6 \end{bmatrix} \text{ gives } \vec{a}_O^2 = \begin{bmatrix} 100 \\ -100 \end{bmatrix}$$

where $\vec{a}_I^p$ and $\vec{a}_O^p$ are the pth training sample's input and output patterns, respectively.

a. Using vector and matrix notation, write the formula(s) used in general for one-step Hebbian learning of weight matrix W in linear associative memories like this.

b. Suppose that $W = 0$ initially and a learning rate $\eta = 1.0$ is used. Show the resultant weight matrix W following one-step Hebbian learning in storing the two associations given above in this network, and how your computed it.

c. With the trained weights W learned in (b), has the trained network correctly learned the two given associations? If you say, yes, what two properties of the input patterns made this possible? If no, what property of the output patterns prevented this?

d. Suppose that this same network was trained on the same two input-output association pairs using the delta rule (derived using gradient descent) instead of Hebbian learning. Would it correctly learn the two associations? No need to do any calculations here: Just answer yes or no, and explain how you know from the information given above.