Artificial Neural Network

Perceptron Introduction & Learning

Preliminaries

- Inputs - An input vector is the data given as one input to the algorithm. Written as x, with elements xi, where i runs from 1 to the number of input dimensions, m.
- Weights - wij - weighted connections between nodes i and j. For neural networks these weights are analogous to the synapses in the brain. They are arranged into a matrix W.
- Outputs - The output vector is y, with elements yj, where j runs from 1 to the number of output dimensions, n. We can write y(x,W) to remind ourselves that the output depends on the inputs to the algorithm and the current set of weights of the network.

Preliminaries

- Targets - The target vector t, with elements tj, where j runs from 1 to the number of output dimensions, n, are the extra data that we need for supervised learning, since they provide the ‘correct’ answers that the algorithm is learning about.

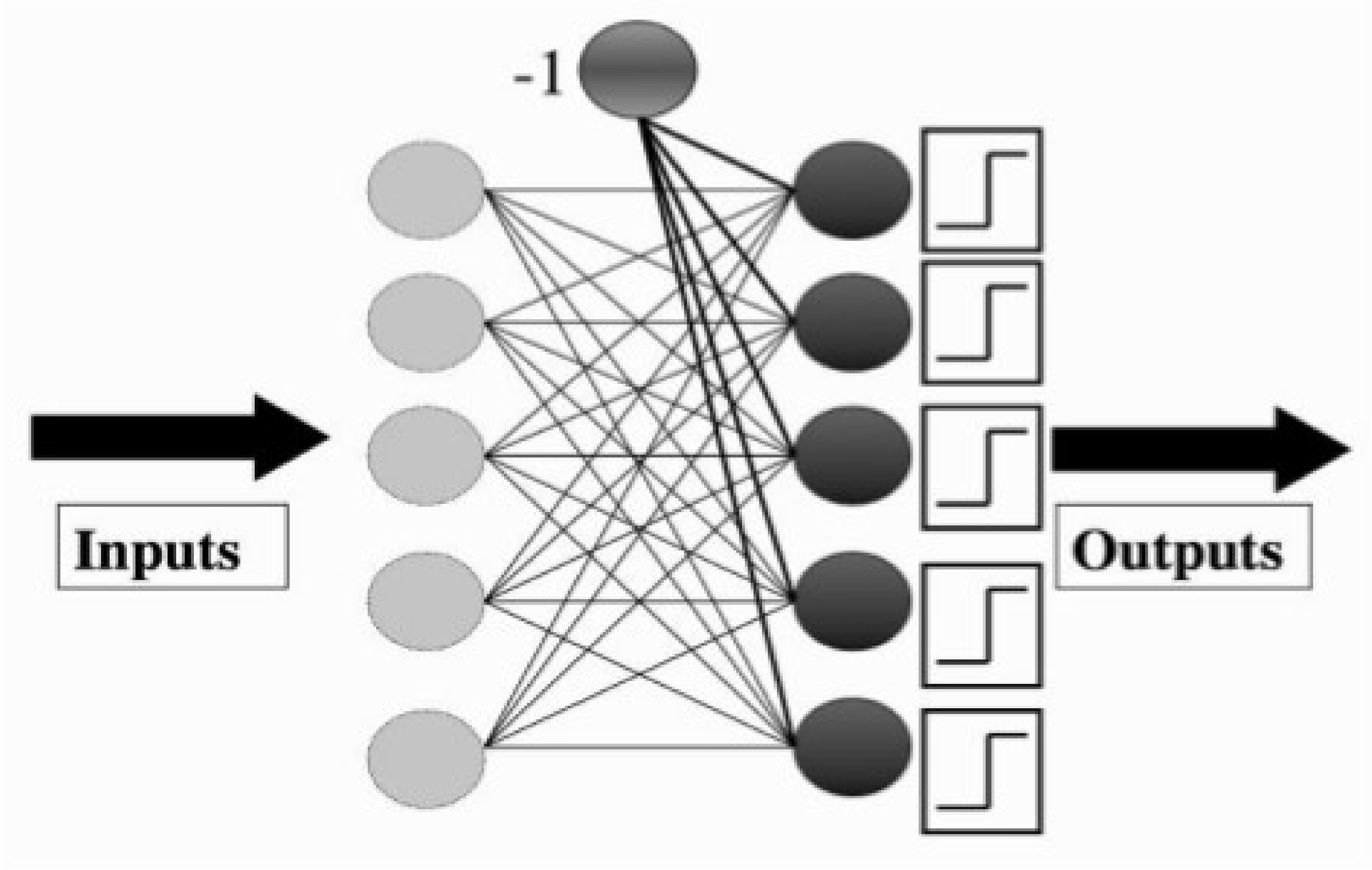
- Activation Function - For neural networks, g(·) is a mathematical function that describes the firing of the neuron as a response to the weighted inputs, such as the threshold function.

- Error E - a function that computes the inaccuracies of the network as a function of the outputs y and targets t.

Perceptron network

- The Perceptron is nothing more than a collection of McCulloch and Pitts neurons together with a set of inputs and some weights to fasten the inputs to the neurons.
- weight update: $\Delta wik = (tk - yk) * xi$, and the new value of the weight is the old value plus this value.

Perceptron network



URE 3.3 The Perceptron network again, showing the bias input.

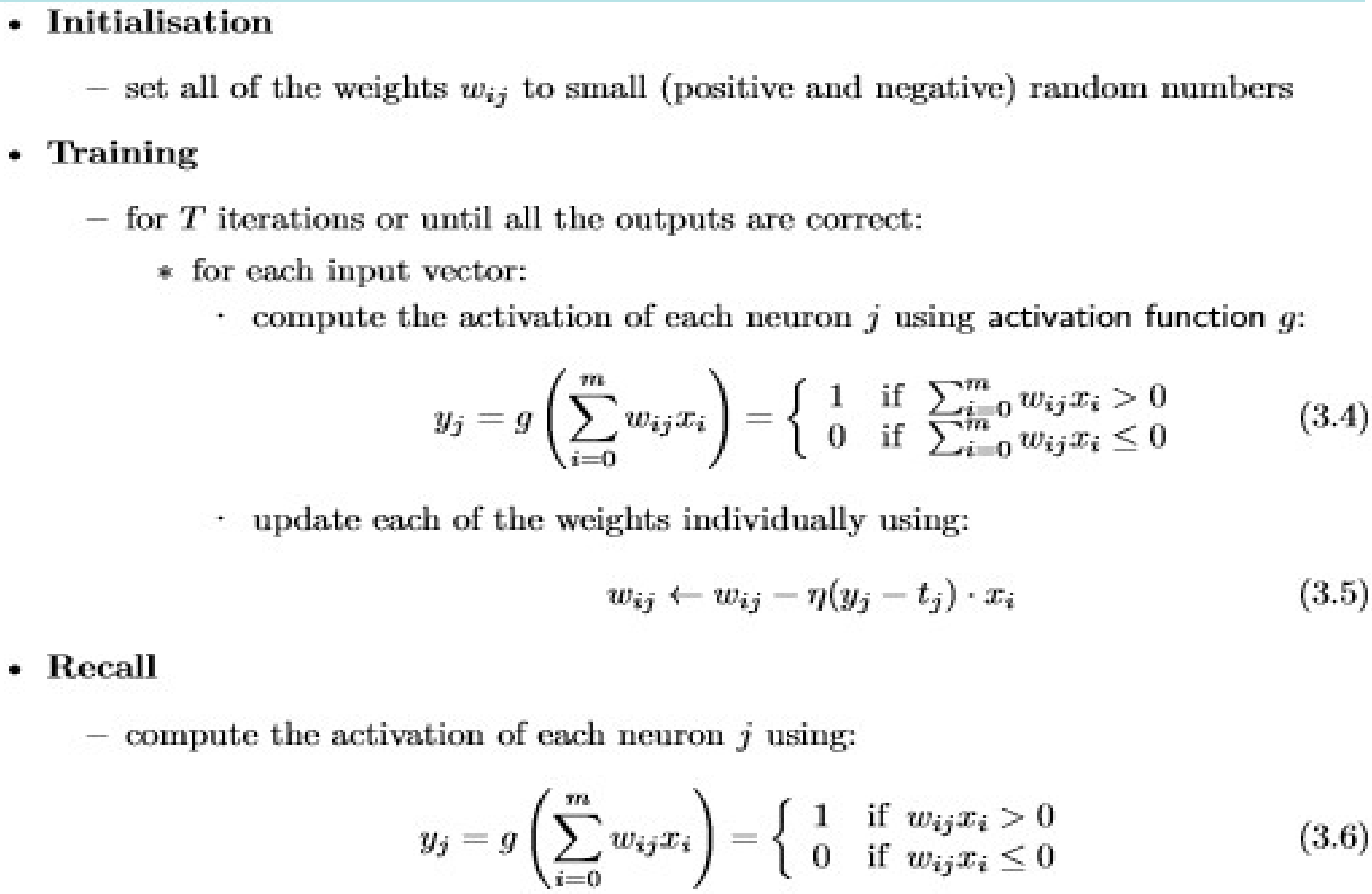
Learning rate

- The value of the learning rate decides how fast the network learns. It’s quite important, so it gets a little subsection of its own (next), but first let’s write down the final rule for updating a weight wij: $wij \leftarrow wij + \eta(tj - yj) \cdot xi$.
- moderate learning rate, typically $0.1 < \eta < 0.4$
- learning rate is a crucial parameter for convergence

Bias input

- an extra input weight to the neuron, with the value of the input to that weight always being fixed (usually the value of -± is chosen
- the value of the weight will change to make the neuron fire—or not fire, whichever is correct—when an input of all zeros is given, since the input on that weight is always -1, even when all the other inputs are zero.
- This input is called a bias node, and its weights are usually given a 0 subscript, so that the weight connecting it to the jth neuron is w0j.

Perceptron – learning algorithm



- **Initialisation**
  - set all of the weights w_{ij} to small (positive and negative) random numbers
- **Training**
  - for T iterations or until all the outputs are correct:
    - for each input vector:
      - compute the activation of each neuron j using activation function g:

$$y_j = g\left(\sum_{i=0}^{m} w_{ij}x_i\right) = \begin{cases} 1 & \text{if } \sum_{i=0}^{m} w_{ij}x_i > 0 \\ 0 & \text{if } \sum_{i=0}^{m} w_{ij}x_i \leq 0 \end{cases} \tag{3.4}$$

      - update each of the weights individually using:

$$w_{ij} \leftarrow w_{ij} - \eta(y_j - t_j) \cdot x_i \tag{3.5}$$

- **Recall**
  - compute the activation of each neuron j using:

$$y_j = g\left(\sum_{i=0}^{m} w_{ij}x_i\right) = \begin{cases} 1 & \text{if } w_{ij}x_i > 0 \\ 0 & \text{if } w_{ij}x_i \leq 0 \end{cases} \tag{3.6}$$

Complexity

- The recall phase loops over the neurons, and within that loops over the inputs, so its complexity is O(mn).
- The training part does this same thing, but does it for T iterations, so costs O(Tmn).