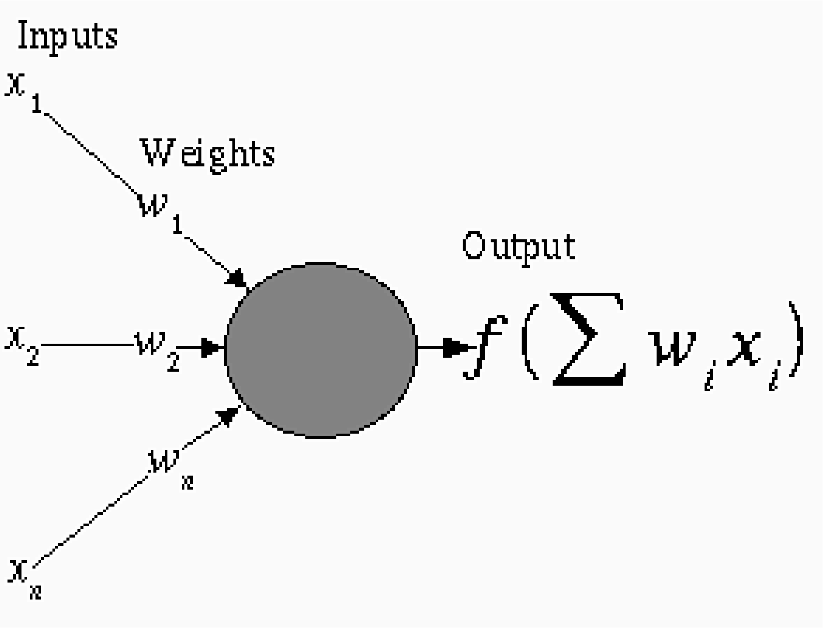
An Introduction To Neural Network, Backpropagation Algorithm

Basic Neuron Model In A Feedforward Network

- **Inputs x_i** arrive through pre-synaptic connections
- Synaptic efficacy is modeled using real **weights w_i**
- The response of the neuron is a **nonlinear function *f*** of its weighted inputs



Copyright G. A. Tagliarini, PhD

Inputs To Neurons

- Arise from other neurons or from outside the network
- Nodes whose inputs arise outside the network are called *input nodes* and simply copy values
- An input may *excite* or *inhibit* the response of the neuron to which it is applied, depending upon the weight of the connection

Copyright G. A. Tagliarini, PhD

Weights

- Represent synaptic efficacy and may be *excitatory* or *inhibitory*
- Normally, positive weights are considered as excitatory while negative weights are thought of as inhibitory
- ***Learning*** is the process of modifying the weights in order to produce a network that performs some function

Copyright G. A. Tagliarini, PhD

Output

- The response function is normally nonlinear
- Samples include
  - Sigmoid

$$f(x) = \frac{1}{1+e^{-\lambda x}}$$

  - Piecewise linear

$$f(x) = \begin{cases} x, if \ x \geq \theta \\ 0, if \ x < \theta \end{cases}$$

Copyright G. A. Tagliarini, PhD

Backpropagation Preparation

- **Training Set**
  A collection of input-output patterns that are used to train the network
- **Testing Set**
  A collection of input-output patterns that are used to assess network performance
- **Learning Rate-*η***
  A scalar parameter, analogous to step size in numerical integration, used to set the rate of adjustments

Copyright G. A. Tagliarini, PhD

Network Error

- Total-Sum-Squared-Error (TSSE)

$$TSSE = \frac{1}{2} \sum_{patterns} \sum_{outputs} (desired - actual)^2$$

- Root-Mean-Squared-Error (RMSE)

$$RMSE = \sqrt{\frac{2 * TSSE}{\# patterns * \# outputs}}$$

Copyright G. A. Tagliarini, PhD

A Pseudo-Code Algorithm

- Randomly choose the initial weights
- While error is too large
  - For each training pattern (presented in random order)
    - Apply the inputs to the network
    - Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer
    - Calculate the error at the outputs
    - Use the output error to compute error signals for pre-output layers
    - Use the error signals to compute weight adjustments
    - Apply the weight adjustments
  - Periodically evaluate the network performance

Copyright G. A. Tagliarini, PhD

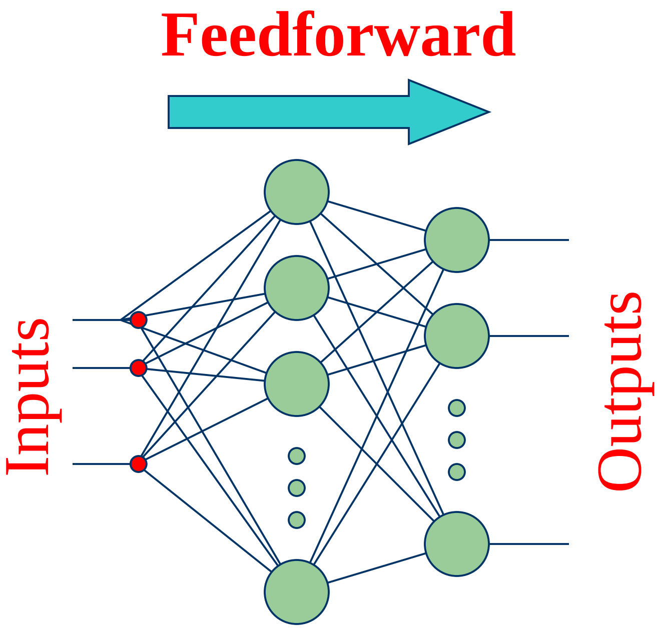
Possible Data Structures

- Two-dimensional arrays
  - Weights (at least for input-to-hidden layer and hidden-to-output layer connections)
  - Weight changes (Δ_{ij})
- One-dimensional arrays
  - Neuron layers
    - Cumulative current input
    - Current output
    - Error signal for each neuron
  - Bias weights

Copyright G. A. Tagliarini, PhD

Apply Inputs From A Pattern

- Apply the value of each input parameter to each input node
- Input nodes compute only the identity function



Copyright G. A. Tagliarini, PhD

Calculate Outputs For Each Neuron Based On The Pattern

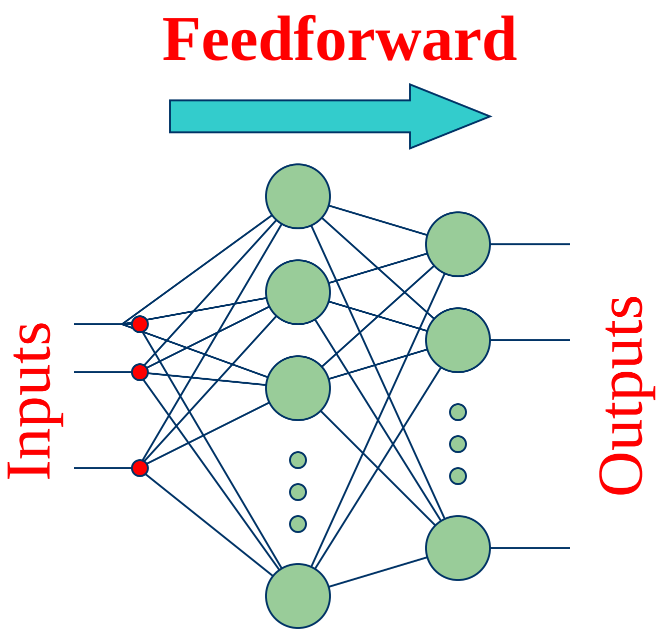
- The output from neuron j for pattern p is O_{pj} where

$$O_{pj}(net_j) = \frac{1}{1+e^{-\lambda net_j}}$$

and

$$net_j = bias * W_{bias} + \sum_k O_{pk} W_{kj}$$

k ranges over the input indices and W_{jk} is the weight on the connection from input k to neuron j



Copyright G. A. Tagliarini, PhD

Calculate The Error Signal For Each Output Neuron

- The output neuron error signal δ_{pj} is given by $\delta_{pj}=(T_{pj}-O_{pj})\, O_{pj}\,(1-O_{pj})$
- T_{pj} is the target value of output neuron j for pattern p
- O_{pj} is the actual output value of output neuron j for pattern p

Copyright G. A. Tagliarini, PhD

Calculate The Error Signal For Each Hidden Neuron

- The hidden neuron error signal δ_{pj} is given by

$$\delta_{pj} = O_{pj}(1 - O_{pj}) \sum_k \delta_{pk} W_{kj}$$

where δ_{pk} is the error signal of a post-synaptic neuron k and W_{kj} is the weight of the connection from hidden neuron j to the post-synaptic neuron k

Copyright G. A. Tagliarini, PhD

Calculate And Apply Weight Adjustments

- Compute weight adjustments ΔW_{ji} at time t by

$$\Delta W_{ji}(t) = \eta\, \delta_{pj}\, O_{pi}$$

- Apply weight adjustments according to

$$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}(t)$$

- *Some add a momentum term* $\alpha * \Delta W_{ji}(t-1)$

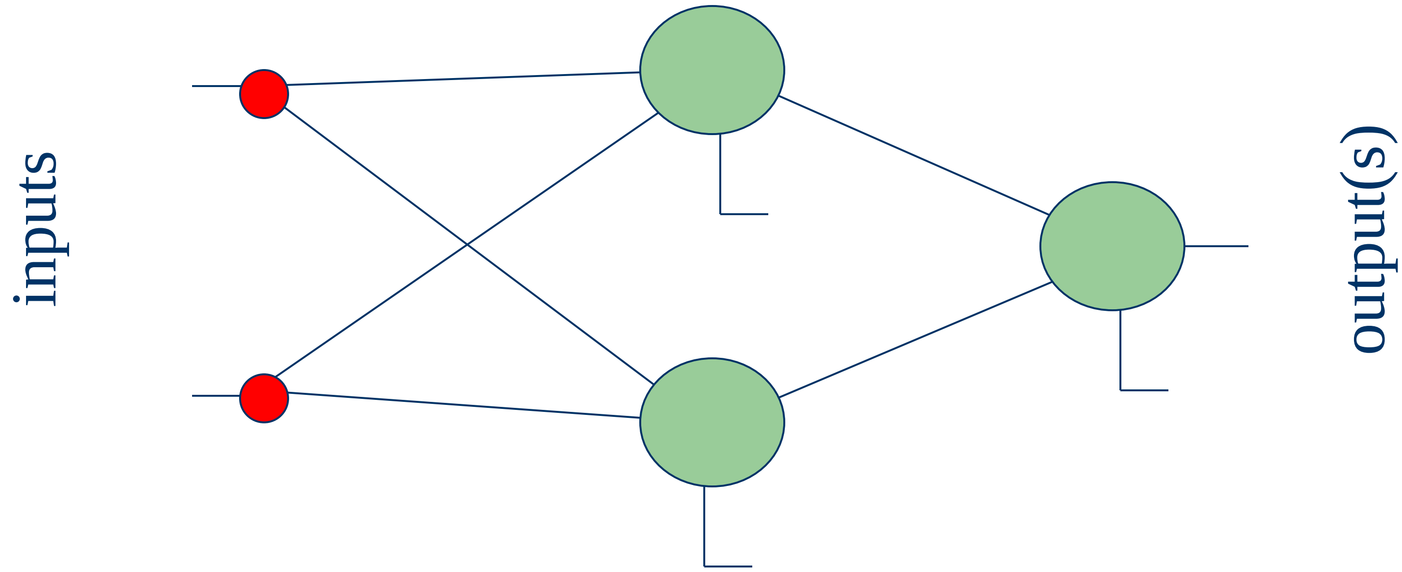
Copyright G. A. Tagliarini, PhD

An Example: Exclusive “OR”

- Training set
  - ((0.1, 0.1), 0.1)
  - ((0.1, 0.9), 0.9)
  - ((0.9, 0.1), 0.9)
  - ((0.9, 0.9), 0.1)
- Testing set
  - Use at least 121 pairs equally spaced on the unit square and plot the results
  - Omit the training set (if desired)

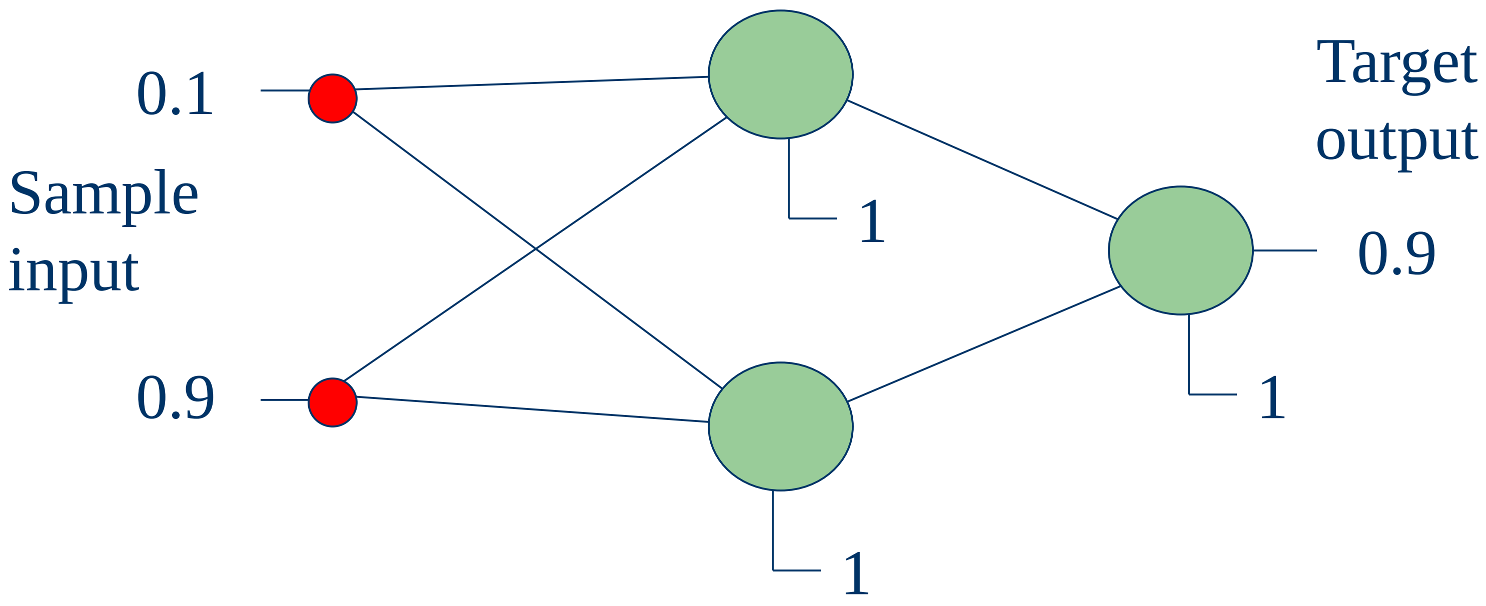
Copyright G. A. Tagliarini, PhD

An Example (continued): Network Architecture



Copyright G. A. Tagliarini, PhD

An Example (continued): Network Architecture

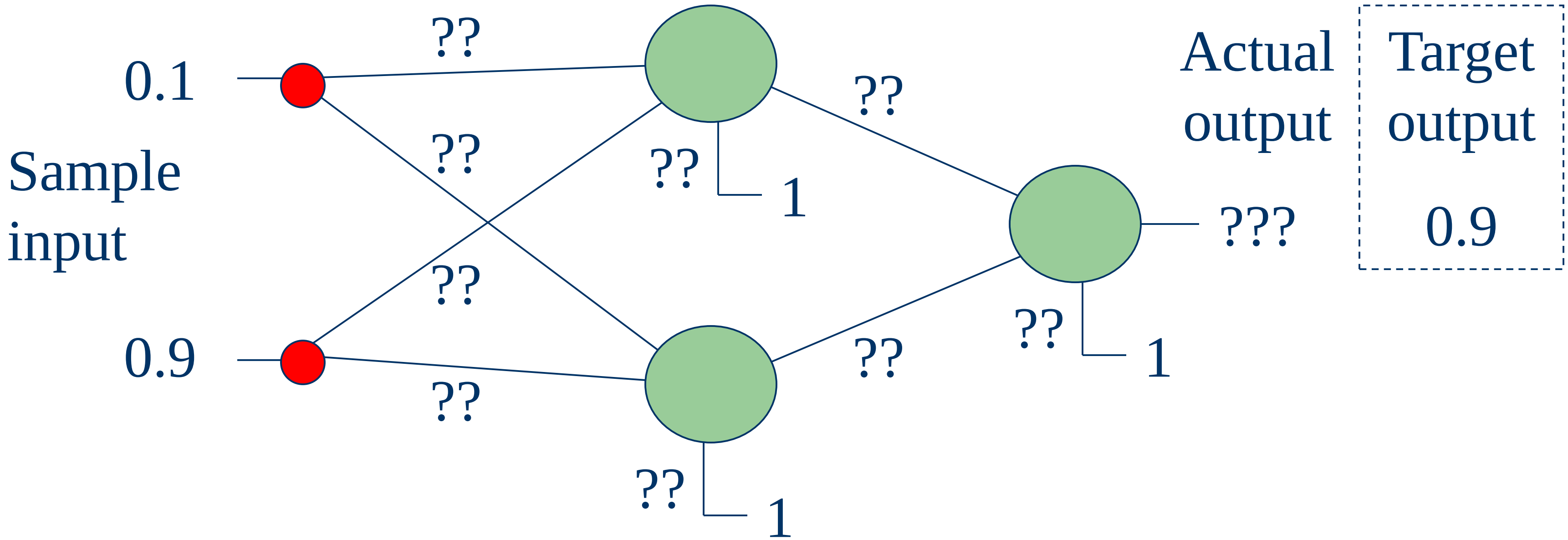


Feedforward Network Training by Backpropagation: Process Summary

- Select an architecture
- Randomly initialize weights
- While error is too large
  - Select training pattern and feedforward to find actual network output
  - Calculate errors and backpropagate error signals
  - Adjust weights
- Evaluate performance using the test set

Copyright G. A. Tagliarini, PhD

An Example (continued): Network Architecture



Copyright G. A. Tagliarini, PhD