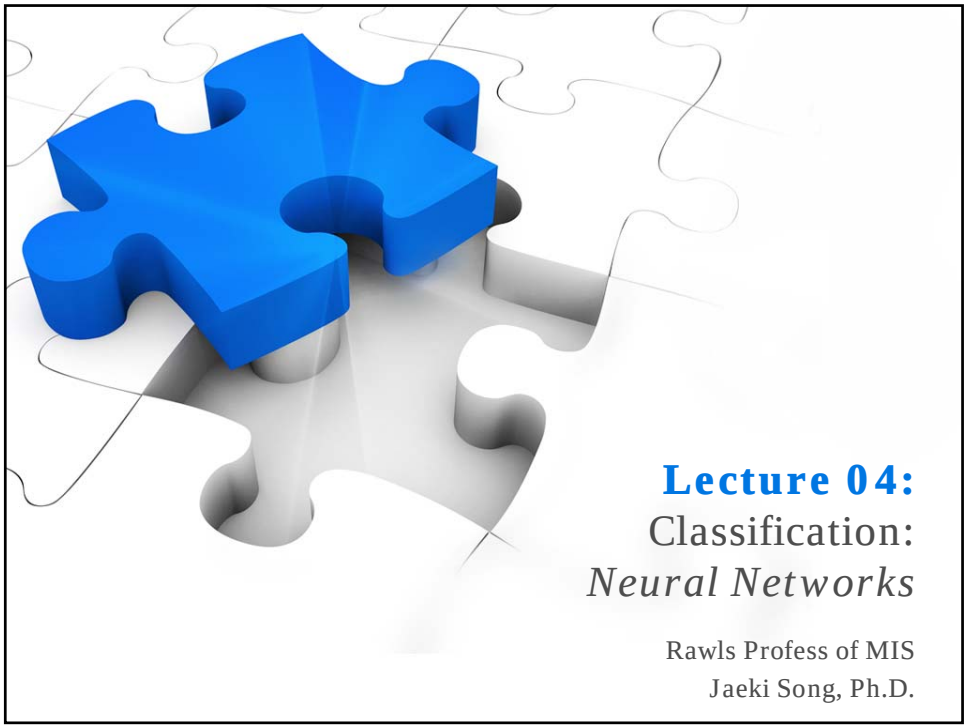
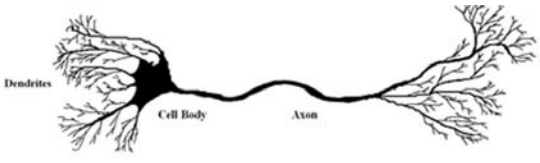


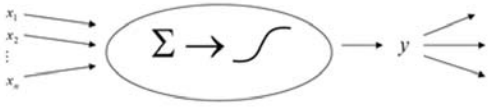
Lecture 04:
Classification:
Neural Networks

Rawls Profess of MIS
Jaeki Song, Ph.D.

Overview

- *Artificial neural networks* (hereafter, ***neural networks***) represent an attempt at a very basic level to imitate the type of nonlinear learning that occurs in the networks of neurons found in nature





Encoding: Input and Output

- Continuous variables:
  - min-max normalization $X^* = \frac{X - \min(X)}{\text{range}(X)} = \frac{X - \min(X)}{\max(X) - \min(X)}$
- Categorical variables:
  - Indicator (flag) variables:
    - Ie: females represented as “female”=1 and “male”=0
  - Ordered classes:
    - Ie: If $0 \le output < 0.25$, classify *first-grade reading level*
    - If $0.25 \le output < 0.50$, classify *second-grade reading level*

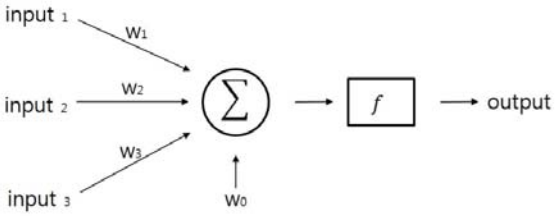
Encoding: Input and Output

- Output
  - neural network output nodes always return a continuous value between zero and 1 as output
  - Denormalization:
    - Prediction = output (data range) + minimum
      - where

        output represents the neural network output in the (0,1) range, *data range* represents the range of the original attribute values on the nonnormalized scale

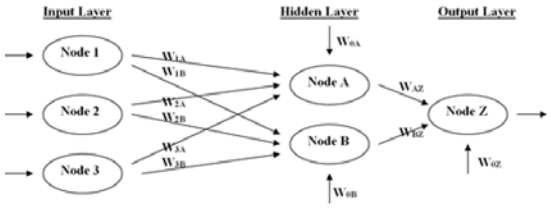
        minimum represents the smallest attribute value on the nonnormalized scale.

Neural Networks



Neural Networks

- Simple example



  - Each connection between nodes has a weight (e.g., W_{1A}) associated with it.
  - a *combination function* (usually summation, Σ) produces a linear combination of the node inputs and the connection weights into a single scalar value, which we will term *net*

Neural Networks

– For a given node j

$$\text{net}_j = \sum_i W_{ij} x_{ij} = W_{0j} x_{0j} + W_{1j} x_{1j} + \cdots + W_{Ij} x_{Ij}$$

where x_{ij} represents the ith input to node j, W_{ij} represents the weight associated with the ith input to node j, and there are $I + 1$ inputs to node j

$x_1, x_2, \ldots, x_I$ represent inputs from upstream nodes, while x_0 represents a *constant* input, analogous to the constant factor in regression models, which by convention uniquely takes the value $x_{0j} = 1$

$x_0 = 1.0$	$W_{0A} = 0.5$	$W_{0B} = 0.7$	$W_{0Z} = 0.05$
$x_1 = 0.4$	$W_{1A} = 0.6$	$W_{1B} = 0.9$	$W_{AZ} = 0.9$
$x_2 = 0.2$	$W_{2A} = 0.8$	$W_{2B} = 0.8$	$W_{BZ} = 0.9$
$x_3 = 0.7$	$W_{3A} = 0.6$	$W_{3B} = 0.4$	

Neural Networks

- Node A
  - combination function $\text{net}_A = 1.32$ is then used as an input to an activation function
    - signals are sent between neurons when the combination of inputs to a particular neuron cross a certain threshold, and the neuron "fires."
  - neural networks model this behavior through a nonlinear activation function.
    - Sigmoid function: $y = \frac{1}{1 + e^{-x}}$

Neural Networks

- Node B

$$net_g = \sum_i W_{iB}x_{iB} = W_{0B}(1) + W_{1B}x_{1B} + W_{2B}x_{2B} + W_{3B}x_{3B}$$

$$= 0.7 + 0.9\,(0.4) + 0.80\,(0.2) +).4(0.7) = 1.5$$

$$f(net_B) = \frac{1}{1 + e^{(-1.5)}} = 0.8176$$

Neural Networks

- Node Z: final node
  - combines these outputs from nodes A and B, through netZ, a weighted sum:

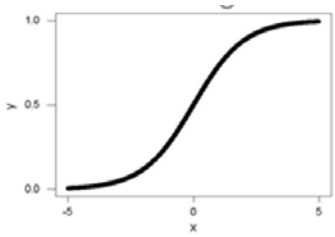
$$net_Z = \sum_i W_{iZ}x_{iZ} = W_{0Z}(1) + W_{AZ}x_{AZ} + W_{BZ}x_{BZ}$$

$$= 0.5 + 0.9(0.7892) + 0.9\,(0.8176) = 1.9461$$

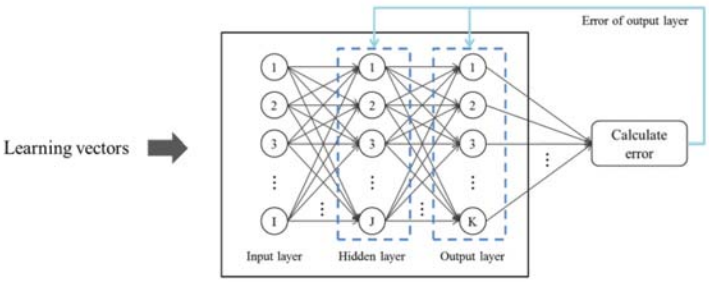
$$f(net_Z) = \frac{1}{1 + e^{(-1.9461)}} = 0.8750$$

SIGMOID Activation Function

- Why use the sigmoid function?
  - combines nearly linear behavior, curvilinear behavior, and nearly constant behavior, depending on the value of the input



How does the neural network learn?



Feedforward Neural Network

How does the neural network learn?

- Back propagation
  - As each observation from the training set is processed through the network, an output value is produced from the output node
  - This output value is then compared to the actual value of the target variable for this training set observation, and the error (actual –output) is calculated
  - This prediction error is analogous to the residuals in regression models
  - Use the sum of squared errors:

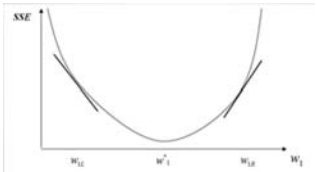
$$SSE = \sum_{records} \sum_{output\ nodes} (actual - output)^2$$

How does the neural network learn?

- Gradient Descent Method
  - use gradient descent method
    - *adjust the weights* in order to decrease SSE
      - The gradient of SSE with respect to the vector of weights **w** is the vector derivative:

$$\nabla SSE(W) = [\frac{\partial SSE}{\partial w_0}, \frac{\partial SSE}{\partial w_1}, \ldots\ldots, \frac{\partial SSE}{\partial w_m}]$$

    - move our current value of w_1 closer to the optimal value as follows: $w_{new} = w_{current} + \Delta w_{current}$, where $\Delta w_{current}$ is the "change in the current location of w."



How does the neural network learn?

- the error responsibility δ_Z for node Z is found. Since node Z is an output node,

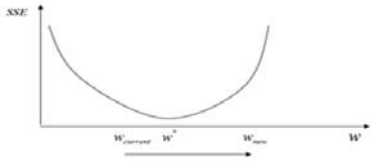
$$\delta_z = Output_Z(1 - Output_Z)(Actual_Z - Output_Z)$$
$$= 0.8750\ (1 - 0.8750)(0.8 - 0.8750) = -0.0082$$

- We may now adjust the "constant" weight W_{0Z} (which transmits an "input" of 1) using the back-propagation rules as follows:

$$W_{0Z,new} = W_{0Z,current} + \Delta W_{0Z} = 0.5 - 0.00082 = 0.49918$$

Learning Rate

- The learning rate η, 0 < η < 1, is a constant chosen to help us move the network weights toward a global minimum for SSE



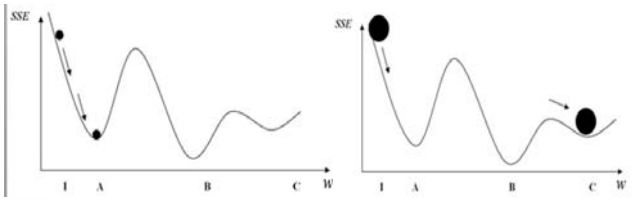
  - allow the learning rate η to change values as the training moves forward
  - At the start of training, η should be initialized to a relatively large value to allow the network to quickly approach the general neighborhood of the optimal solution
  - when the network is beginning to approach convergence, the learning rate should gradually be reduced, thereby avoiding overshooting the minimum

Momentum Term

- Momentum term:

$$\Delta w_{current} = -\eta\left(\frac{\partial SSES}{\partial w_{current}}\right) + \alpha\Delta w_{previous}$$

  - the momentum term represents *inertia*
  - Large values of α will influence the adjustment in the current weight, $\Delta w_{current}$, to move in the same direction as previous adjustments



Termination Criteria

- Termination criterion: criteria used to stop the neural network algorithm to stop passing through the data
  - 1) set the number of passes through the data
  - 2) assesses when the SSE on the training data has been reduced

- cross-validation termination procedure:
  1. Retain part of the original data set as a holdout validation set
  2. Proceed to train the neural network as above on the remaining training data
  3. Apply the weights learned from the training data on the validation data
  4. Monitor *two sets of weights*, one "current" set of weights produced by the training data, and one "best" set of weights, as measured by the lowest SSE so far on the validation data
  5. When the current set of weights has significantly greater SSE than the best set of weights, then terminate the algorithm