COURSE NAME:
DATA WAREHOUSING & DATA MINING

LECTURE 18
TOPICS TO BE COVERED:

- Knowledge discovery through Neural Networks

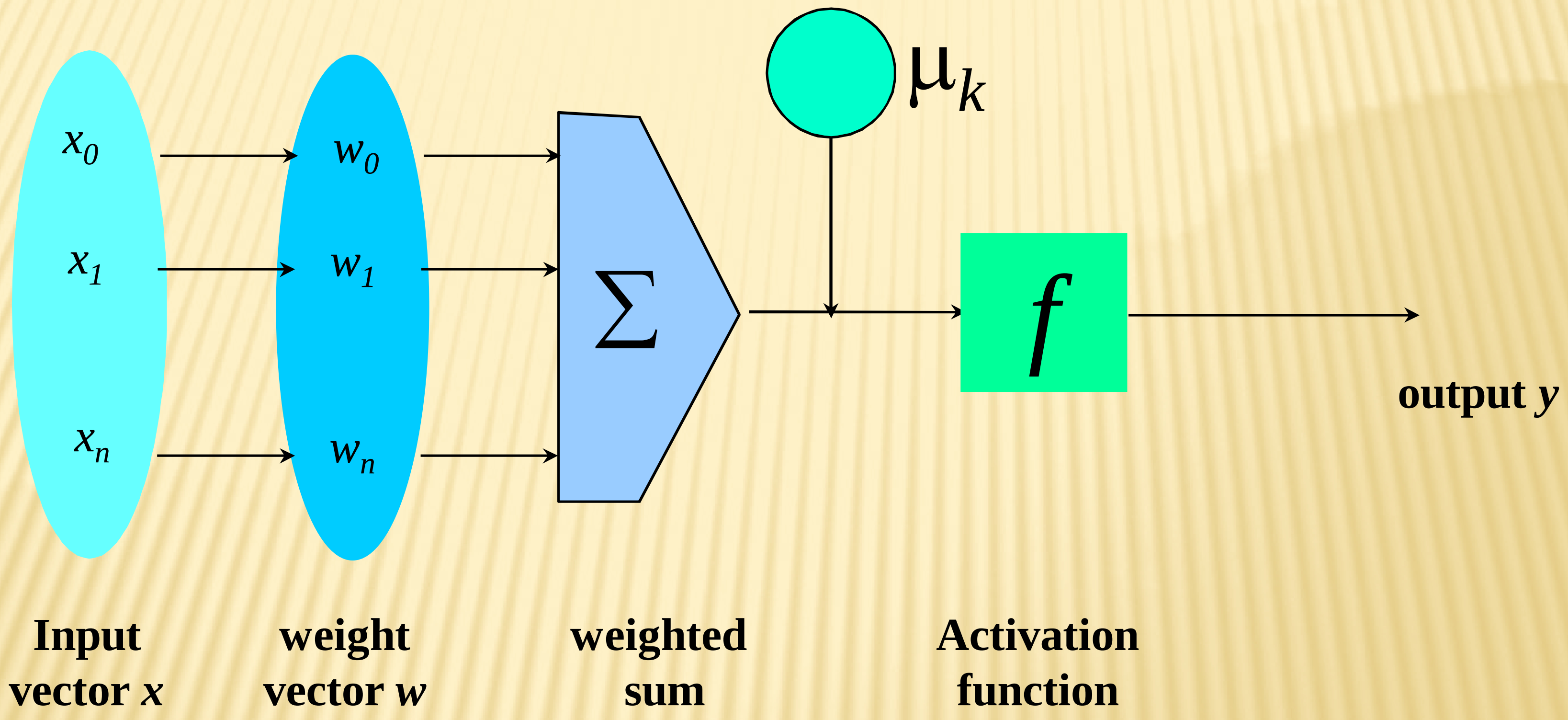
CLASSIFICATION BY BACKPROPAGATION

- Backpropagation is a neural network algorithm for classification that employs a method of gradient descent. It searches for a set of weights that can model the data so as to minimize the mean squared distance between the network's class prediction and the actual class label of data tuples. Rules may be extracted from trained neural networks in order to help improve the interpretability of the learned network .
- The field of neural networks was originally kindled by psychologists and neurobiologists who sought to develop and test computational analogues of neurons.
- A neural network is a set of connected input/output units in which each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples. Neural network learning is also referred to as connectionist learning due to the connections between units..

NEURAL NETWORKS

- Advantages
  - prediction accuracy is generally high
  - robust, works when training examples contain errors
  - output may be discrete, real-valued, or a vector of several discrete or real-valued attributes
  - fast evaluation of the learned target function
- Criticism
  - long training time
  - difficult to understand the learned function (weights)
  - not easy to incorporate domain knowledge

A NEURON



- The n-dimensional input vector x is mapped into variable y by means of the scalar product and a nonlinear function mapping

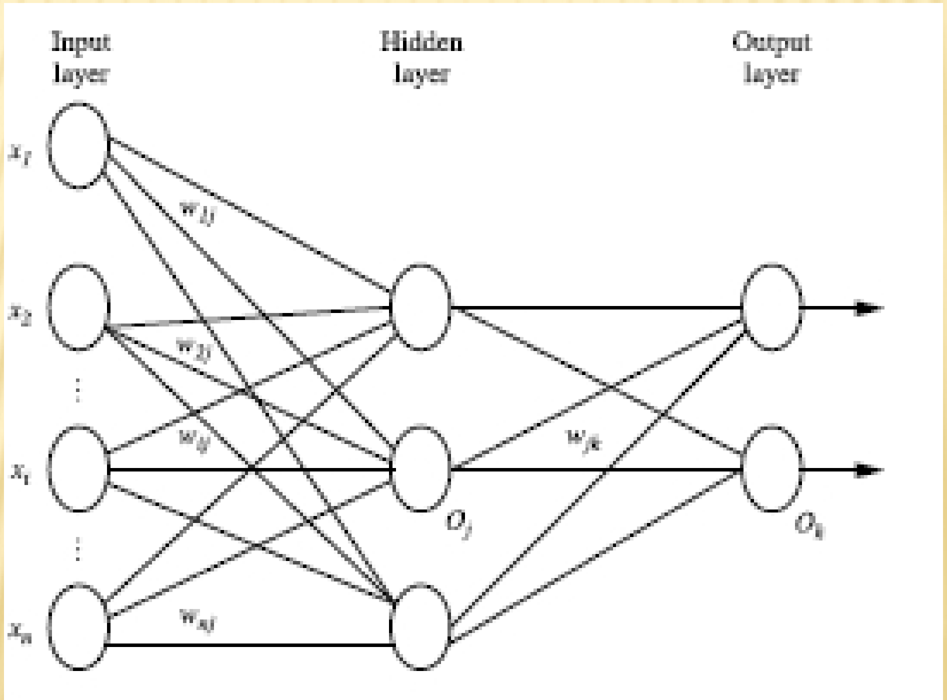
NETWORK TRAINING

- The ultimate objective of training
  - obtain a set of weights that makes almost all the tuples in the training data classified correctly
- Steps
  - Initialize weights with random values
  - Feed the input tuples into the network one by one
  - For each unit
    - Compute the net input to the unit as a linear combination of all the inputs to the unit
    - Compute the output value using the activation function
    - Compute the error
    - Update the weights and the bias

A MULTILAYER FEED-FORWARD NEURAL NETWORK

- The backpropagation algorithm performs learning on a *multilayer feed-forward neural* network. It iteratively learns a set of weights for prediction of the class label of tuples. A multilayer feed-forward neural network consists of an *input layer, one or more hidden layers, and an output layer.*
- Each layer is made up of units. The inputs to the network correspond to the attributes measured for each training tuple. The inputs are fed simultaneously into the units making up the input layer. These inputs pass through the input layer and are then weighted and fed simultaneously to a second layer of “neuronlike” units, known as a hidden layer. The outputs of the hidden layer units can be input to another hidden layer, and so on.
- The numberof hidden layers is arbitrary, although in practice, usually only one is used. The weighted outputs of the last hidden layer are input to units making up the output layer, which emits the network’s prediction for given tuples.

- The units in the input layer are called input units. The units in the hidden layers and output layer are sometimes referred to as neurodes, due to their symbolic biological basis, or as output units.



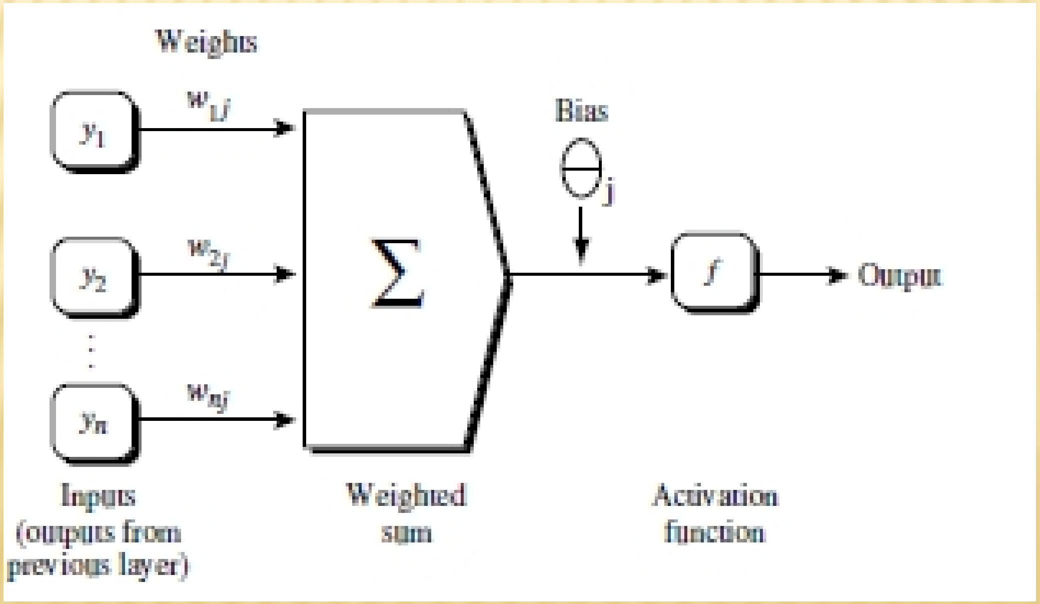
A multilayer feed-forward neural network.

BACKPROPAGATION

- Propagate the inputs forward: First, the training tuple is fed to the input layer of the network. The inputs pass through the input units, unchanged. That is, for an input unit, *j, its output, O_j, is equal to its input value, I_j. Next, the net input and output of each* unit in the hidden and output layers are computed. The net input to a unit in the hidden or output layers is computed as a linear combination of its inputs.
- To compute the net input to the unit, each input connected to the unit is multiplied by its corresponding weight, and this is summed. Given a unit *j in a hidden or output layer, the net input, O_I, to unit j is*

$$I_j = \sum_i w_{ij} O_i + \theta_j,$$

- where w_{ij} *is the weight of the connection from unit i in the previous layer to unit j;*
- O_i *is* the output of unit *i from the previous layer; and*
- θ_j *is the bias of the unit. The bias acts* as a threshold in that it serves to vary the activity of the unit.



- The logistic, or sigmoid, function is used. Given the net input I_j *to unit j, then* O_j*, the output of unit j, is computed as*

$$O_j = \frac{1}{1+e^{-I_j}}.$$

- Backpropagate the error: The error is propagated backward by updating the weights and biases to reflect the error of the network's prediction. For a unit *j in the output layer, the* error Err_j *is computed by*

$$Err_j = O_j(1-O_j)(T_j-O_j),$$

- where O_j *is the ac* … *d* T_j *is the known target value of the given* training tuple.
- Note that O_j $(1- O_j)$ *is the derivative of the logistic function.*

- To compute the error of a hidden layer unit *j, the weighted sum of the errors of the* units connected to unit *j in the next layer are considered. The error of a hidden layer* unit *j is*

$$Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk},$$

- where w_{jk} *is the weight of the connection from unit j to a unit k in the next higher layer,*and
- Err_k *is the error of unit k.*
- The weights and biases are updated to reflect the propagated errors. Weights are updated by the following equations, where Δw_{ij} *is the change in weight* w_{ij}

$$\Delta w_{ij} = (l) Err_j O_i$$
$$w_{ij} = w_{ij} + \Delta w_{ij}$$

- Biases are updated by the following equations below, where $\Delta\ \theta_j$ *is the change in* bias θ_j *:*

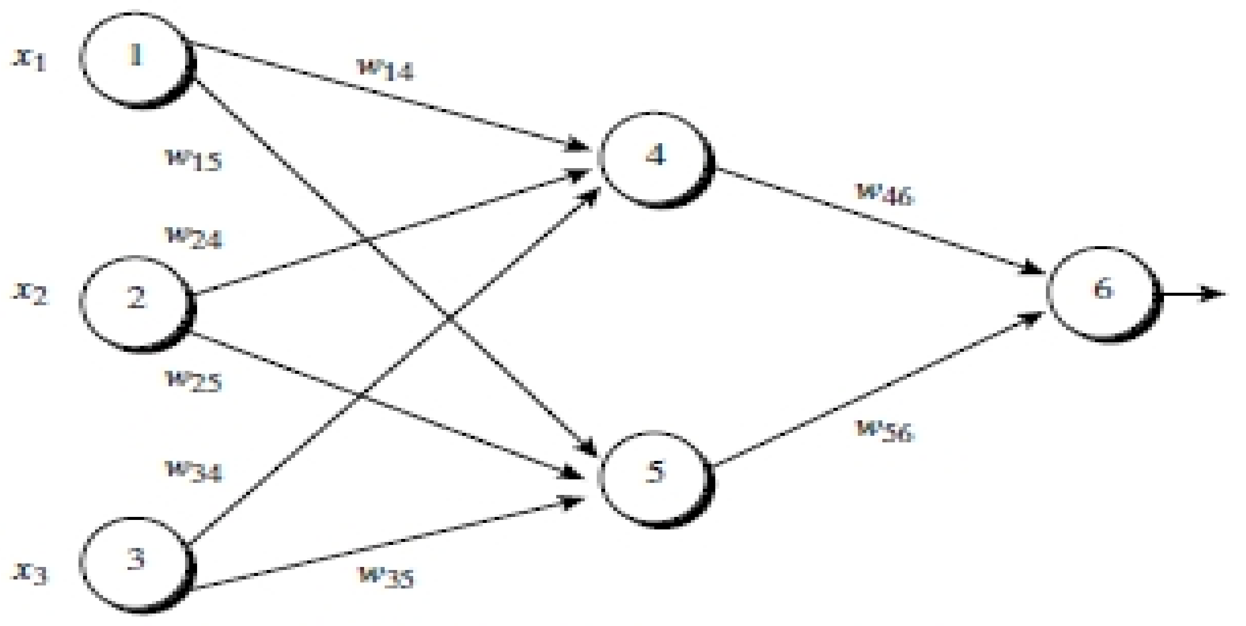
$$\Delta\theta_j = (l)Err_j$$

$$\theta_j = \theta_j + \Delta\theta_j$$

NETWORK PRUNING AND RULE EXTRACTION

- Network pruning
  - Fully connected network will be hard to articulate
  - *N* input nodes, *h* hidden nodes and *m* output nodes lead to *h(m+N)* weights
  - Pruning: Remove some of the links without affecting classification accuracy of the network
- Extracting rules from a trained network
  - Discretize activation values; replace individual activation value by the cluster average maintaining the network accuracy
  - Enumerate the output from the discretized activation values to find rules between activation value and output
  - Find the relationship between the input and activation value
  - Combine the above two to have rules relating the output to input

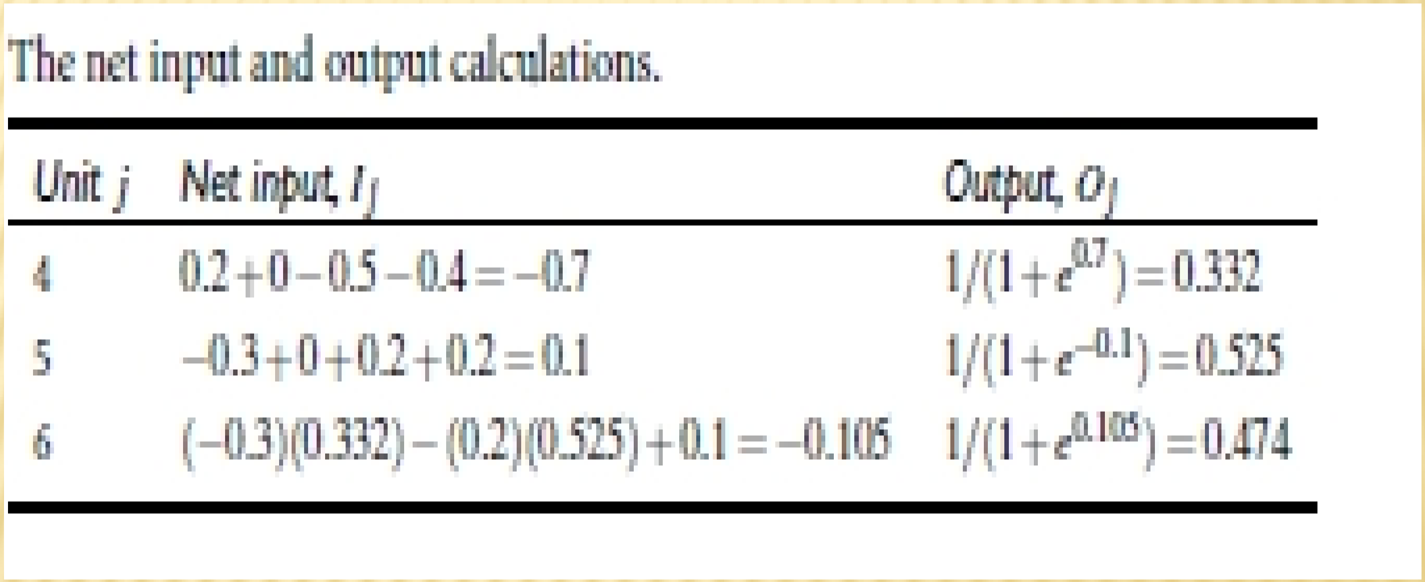
EXAMPLE: SAMPLE CALCULATIONS FOR LEARNING BY THE BACKPROPAGATION ALGORITHM.



An example of a multilayer feed-forward neural network.

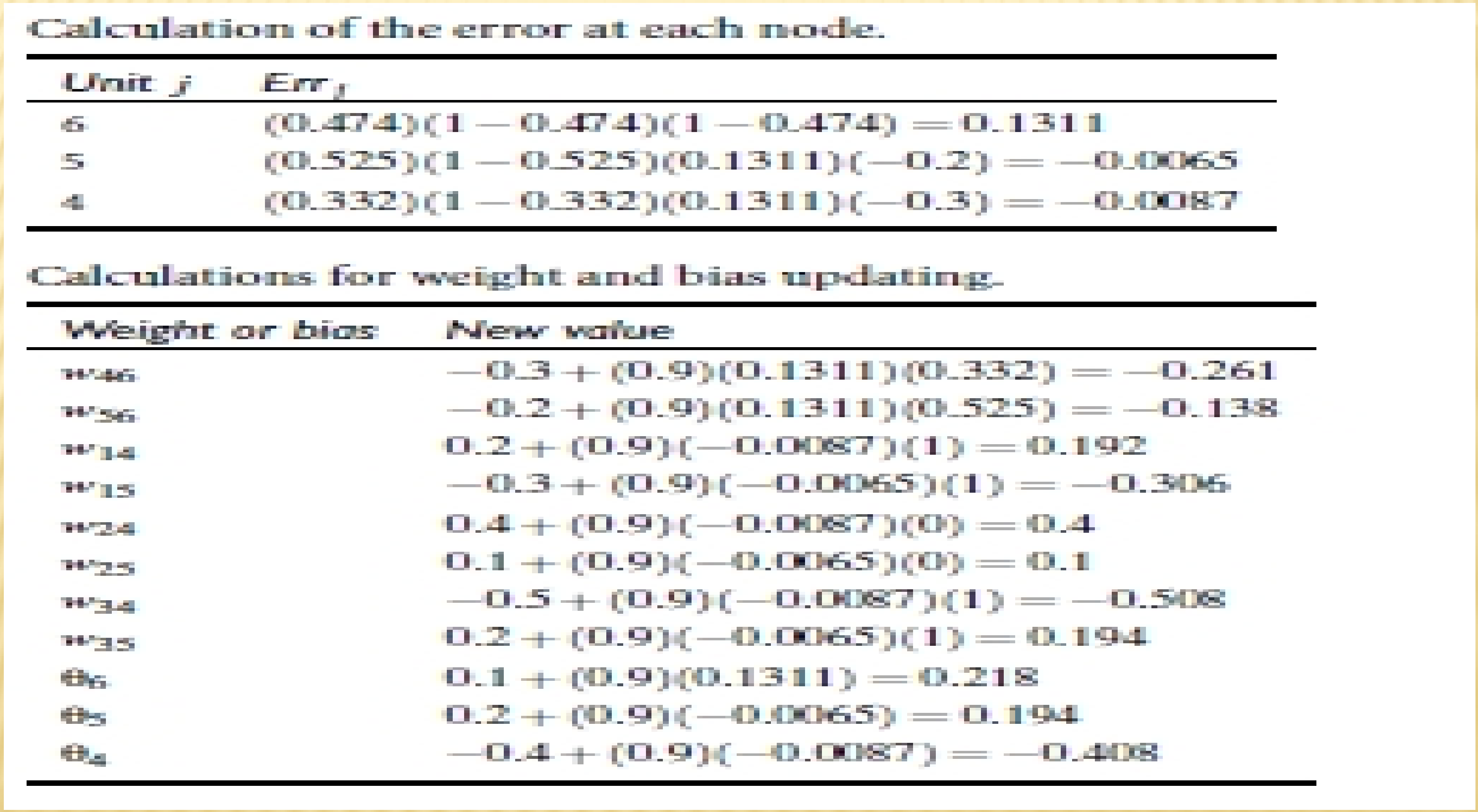
Initial input, weight, and bias values.

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1



The net input and output calculations.

Unit j	Net input, I_j	Output, O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$



Calculation of the error at each node.

Unit j	Err_j
6	$(0.474)(1-0.474)(1-0.474) = 0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2) = -0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3) = -0.0087$

Calculations for weight and bias updating.

Weight or bias	New value
w_{46}	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
w_{56}	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
w_{14}	$0.2 + (0.9)(-0.0087)(1) = 0.192$
w_{15}	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
w_{24}	$0.4 + (0.9)(-0.0087)(0) = 0.4$
w_{25}	$0.1 + (0.9)(-0.0065)(0) = 0.1$
w_{34}	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
w_{35}	$0.2 + (0.9)(-0.0065)(1) = 0.194$
θ_6	$0.1 + (0.9)(0.1311) = 0.218$
θ_5	$0.2 + (0.9)(-0.0065) = 0.194$
θ_4	$-0.4 + (0.9)(-0.0087) = -0.408$