Introduction to Artificial Intelligence

Unit #13-2

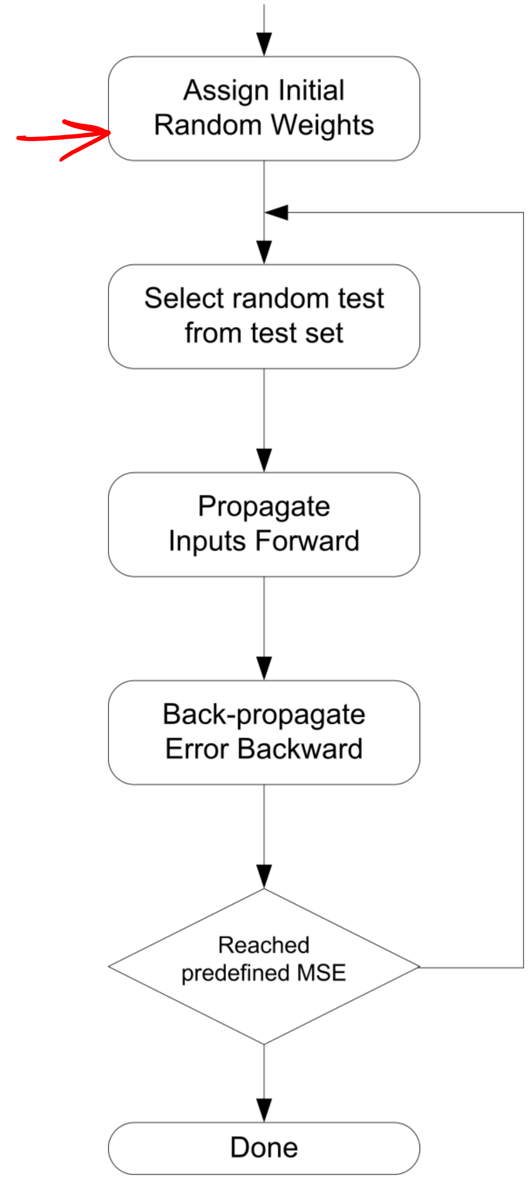
Acknowledgement

- The slides of this lecture have been taken from the lecture slides of CS307 – “Introduction to Artificial Intelligence” and CSE652 – “Knowledge Discovery and Data mining” by Dr. Sajjad Haider.

Learning a Neural Network

Working of ANN

- Learning is accomplished by modifying network connection weights while a set of input instances is repeatedly passed through the network.
- Once trained, an unknown instance passing through the network is classified according to the value(s) seen at the output layer.



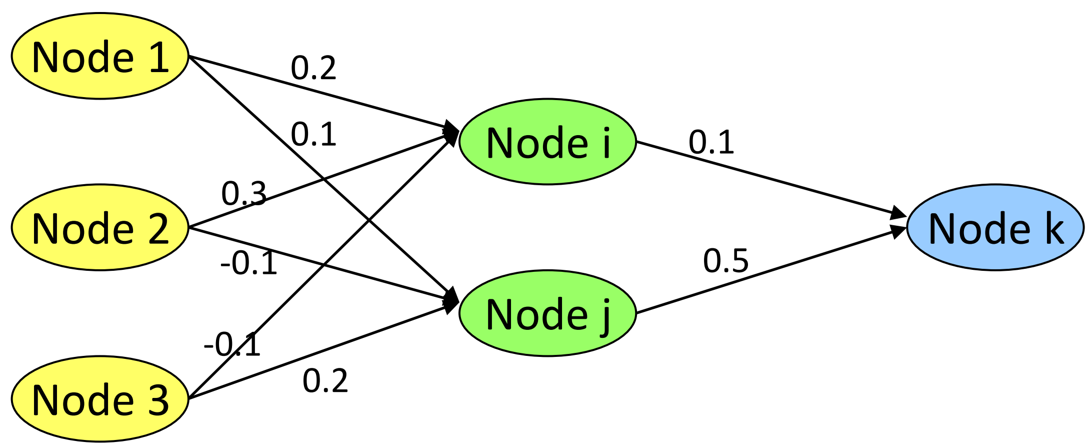
Feed-forwarding Data

w_{1i}=0.20, w_{1j}=0.10, w_{2i}=0.30, w_{2j}=-0.10, w_{3i}=-0.10, w_{3j}=0.20, w_{ik}=0.10, w_{jk}=0.50, T=0.65

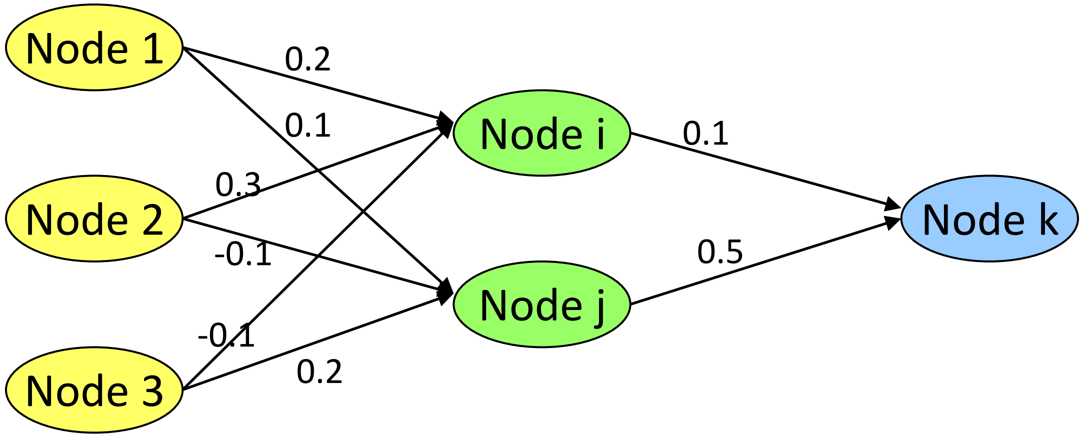
- Input = {1.0, 0.4, 0.7}
- Input to node i = 0.2x1.0 + 0.3x0.4 - 0.1x0.7 = 0.25
- Now apply the sigmoid function: = 0.562

- Input to node j = ?
- Output of node j = ?

- Input to node k = ?
- Output of node k = ?



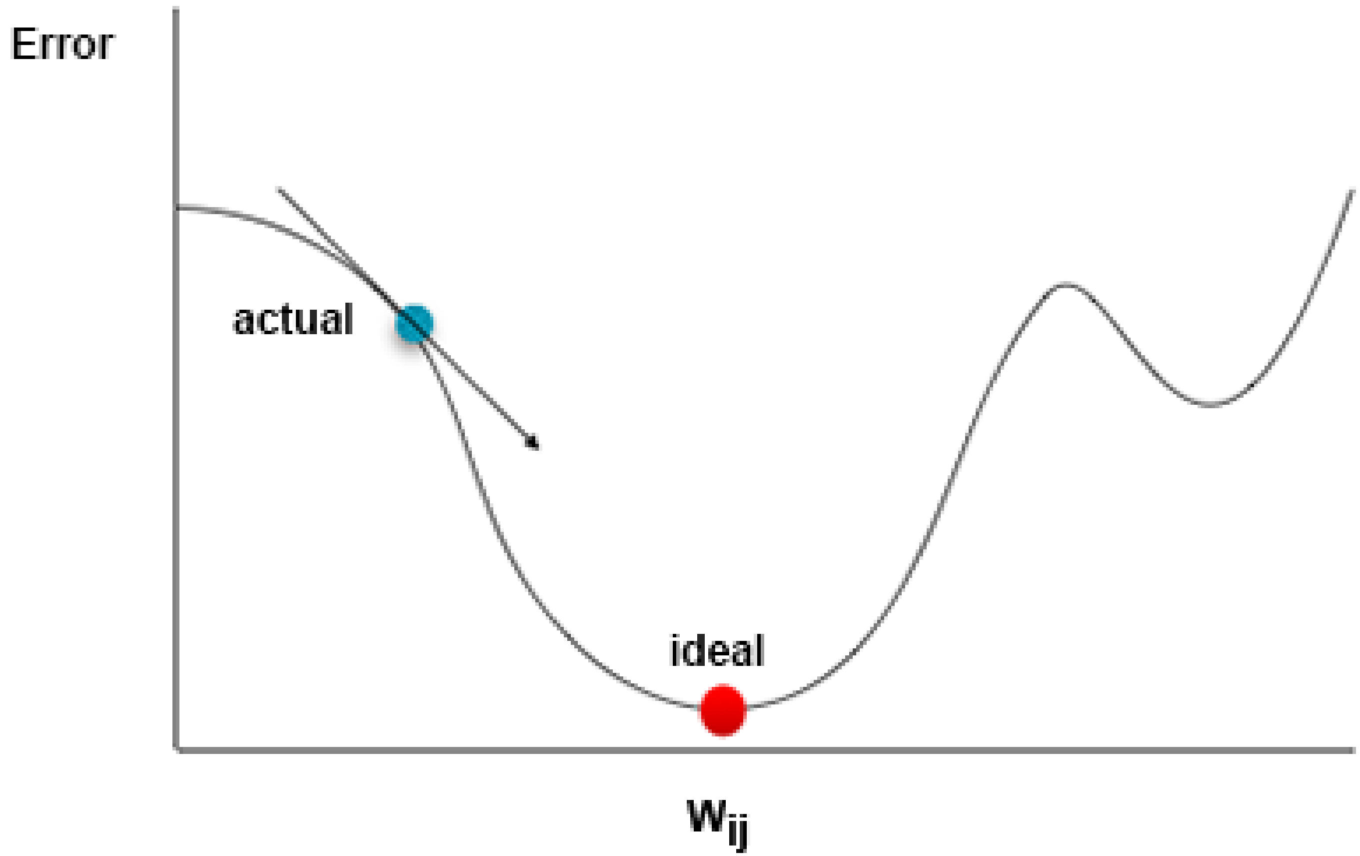
- Input node k =?
- Output of node k =?


Node 1
Node 2
Node 3
Node i
Node j
Node k
0.2
0.1
0.3
-0.1
-0.1
0.2
0.1
0.5

Gradient Descent

- At a theoretical level, gradient descent is an algorithm that minimizes functions. Given a function defined by a set of parameters, gradient descent **starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimize the function.** This iterative minimization is achieved using calculus, **taking steps in the negative direction of the function gradient**.

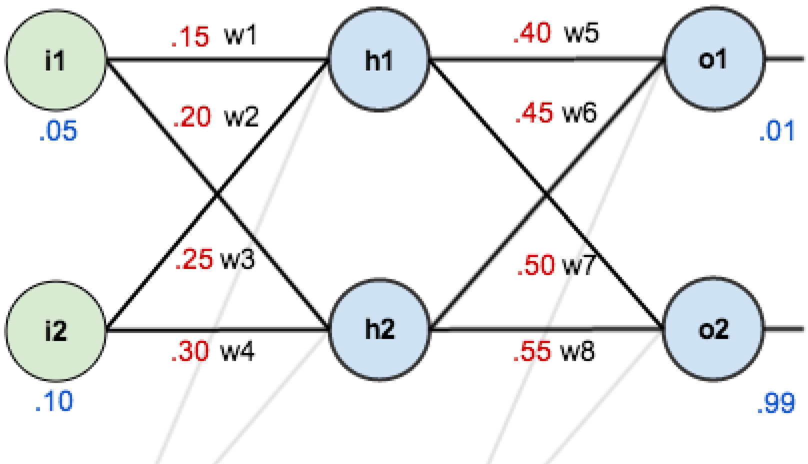
Gradient

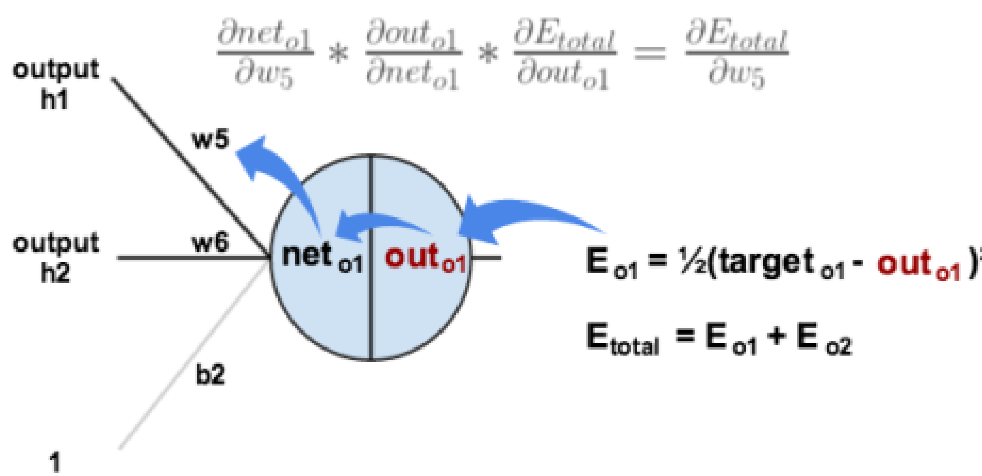


Gradient

- The gradient of each weight gives an indication about how to modify each weight to achieve the expected output (or reduce the error).
- Each weight has a gradient that is slope of the error function.
  - Zero gradient implies theta the weight is not contributing to the error
  - Negative gradient implies that the weight should be increased to achieve a lower error
  - Positive gradient implies that the weight should be decreased to achieve a lower error

Backward Pass





- https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

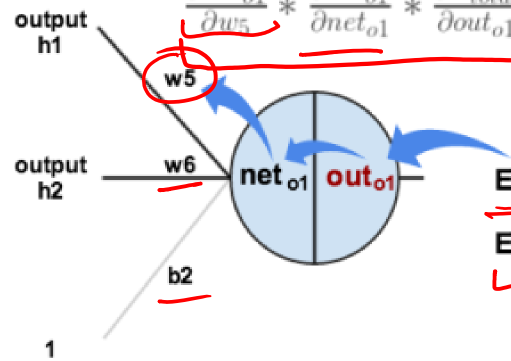
$$\phi = \frac{1}{1+e^{-\phi}}$$

$$\phi(1-\phi)$$

$$\frac{d}{dx}$$

Taking derivative

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

$$\frac{\partial E_{Total}}{\partial out_{o1}} = \frac{1}{2} \times 2(target_{o1} - out_{o1})(-1)$$

$$= (out_{o1} - target)$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1})$$

$$\frac{\partial net_{o1}}{\partial w_5} = w_5 h_1 + w_6 h_2$$

$$= h_1 + 0$$

Explanation of the Backpropagation Algorithm

$w_{1i}=0.20$, $w_{1j}=0.10$, $w_{2i}=0.30$, $w_{2j}=-0.10$, $w_{3i}=-0.10$, $w_{3j}=0.20$, $w_{ik}=0.10$, $w_{jk}=0.50$, $T=0.65$

1000

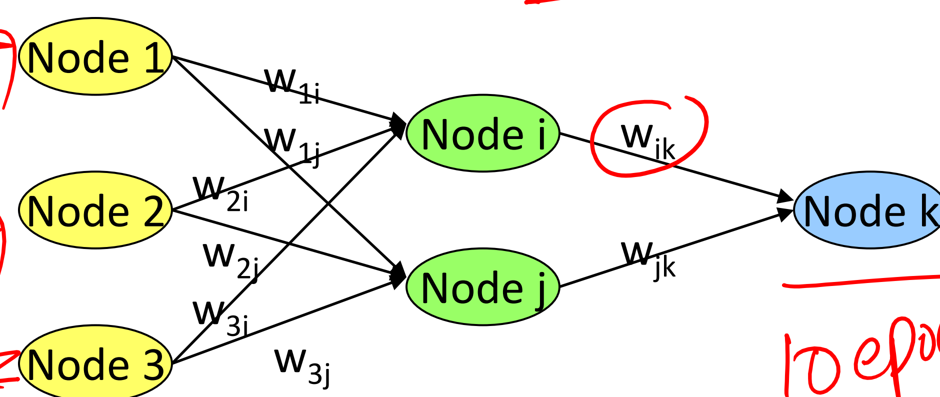
1 epoch

- Input = {1.0, 0.4, 0.7}
- Input to node i = 0.2x1.0 + 0.3x0.4 - 0.1x0.7 = 0.25
- Now apply the sigmoid function: f(0.25) = 0.562
- Input to node j = ? 0.549

  $O_j = 0.549$
- Input to node k = ? ~~0.582~~

  $O_K = 0.582$
- Error(k) = $(T - O_k)\, O_k\, (1 - O_k)$
  - T = the target output
  - O_k = the computed output at node k
- Error(k) = ? $(0.65 - 0.582)\, 0.582\, (1 - 0.582)$

  $= 0.4$[illegible]



10 epoch

Explanation of the Backpropagation Algorithm

w_{1i}=0.20, w_{1j}=0.10, w_{2i}=0.30, w_{2j}=-0.10, w_{3i}=-0.10, w_{3j}=0.20, w_{ik}=0.10, w_{jk}=0.50

- Error(i) = Error(k) w_{ik} O_i (1 – O_i)

  = ? $0.418 \times 0.1\,(0.562)\,(1 - 0.562) = 0.01$

- Error(j) = ? $Err(K)\ w_{jK}\ O_j(1-O_j)$ $\quad$ 0.0517

  $0.418 \times 0.5 \times 0.549\,(1 - 0.549) = \cancel{0.517}$

- The next step is to update the weights associated with the individual node connections.
- Weight adjustments are made using the delta rule
  – To minimize the sum of the square errors, where error is defined as the distance between computed and actual output

Explanation of the Backpropagation Algorithm

w_{1i}=0.20, w_{1j}=0.10, w_{2i}=0.30, w_{2j}=-0.10, w_{3i}=-0.10, w_{3j}=0.20, w_{ik}=0.10, w_{jk}=0.50

- $w_{ik} = w_{ik}\text{ (current)} + \Delta w_{ik}$ 0.1 +
- $\Delta w_{ik} = r \times \text{Error}(k) \times O_i$ 0.8 $= (0.8 \times 0.418 \times 0.562)$ $= 0.187$
  - where r is learning rate parameter, 0 < r < 1   100%
- Compute: Δw_{ik} Δw_{1i} Δw_{2i} Δw_{3i}

$w'_{jk} = 0.5 + (0.8 \times 0.418 \times 0.549)$

$= 0.683$

$w'_{1i} = W_{1i} + \cancel{(r \times Err}(i) \times O_1) =$

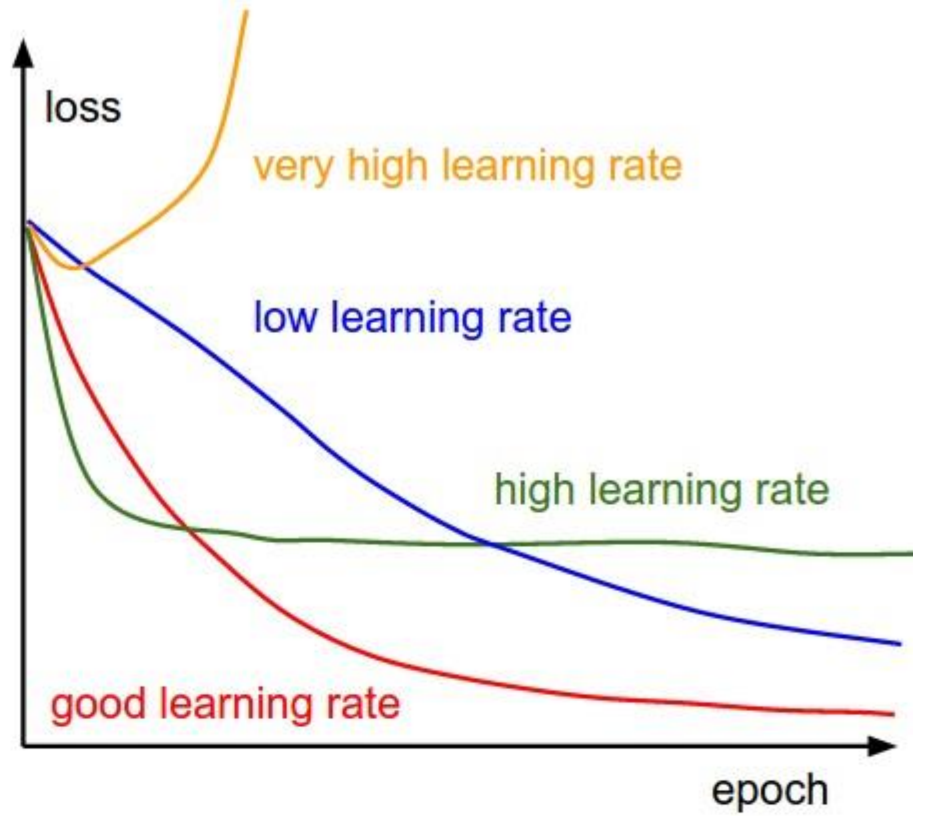
Algorithm

- Initialize the network:
  - Create the network topology by choosing the number of nodes for the input, hidden, and output layers.
  - Initialize weights for all node connections to arbitrary values between -1.0 and 1.0.
  - Choose a value between 0 and 1 for the learning parameter.
  - Choose a terminating condition.
- For all the training instances:
  - Feed the training instance through the network.
  - Determine the output error.
  - Updated the network weights.
- If the terminating condition has not been met, repeat step 2.
- Test the accuracy of the network on a test dataset. If the accuracy is less than optimal, change one or more parameters of the network topology and start over.

Training/Testing of ANN

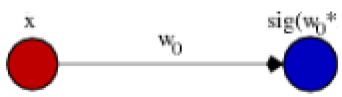
- During the training phase, training instances are repeatedly passed through the network while individual weight values are modified.
- The purpose of changing the connection weights is to minimize training set error rate.
- Network training continues until a specific terminating condition is satisfied.
- The terminating condition can be convergence of the network to a minimum total error value, a specific time criterion, or a maximum number of iterations.

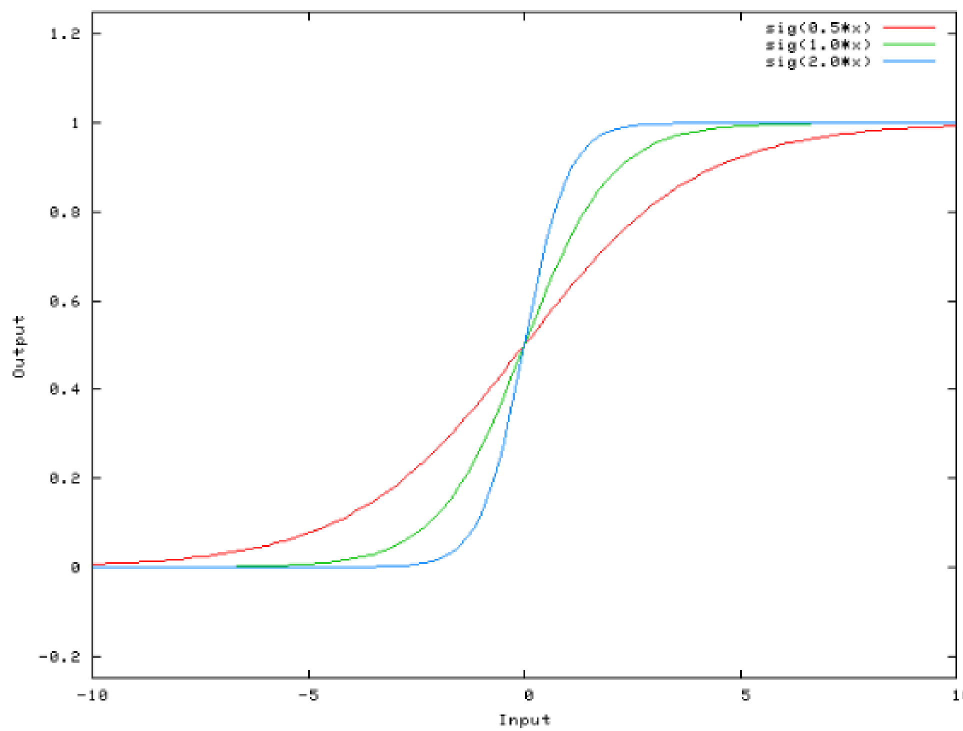
Learning Rate



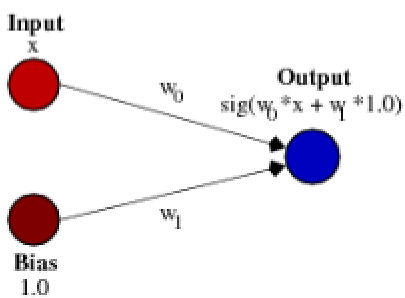
Bias in ANN

- **A bias value allows you to shift the activation function to the left or right**, which may be critical for successful learning.

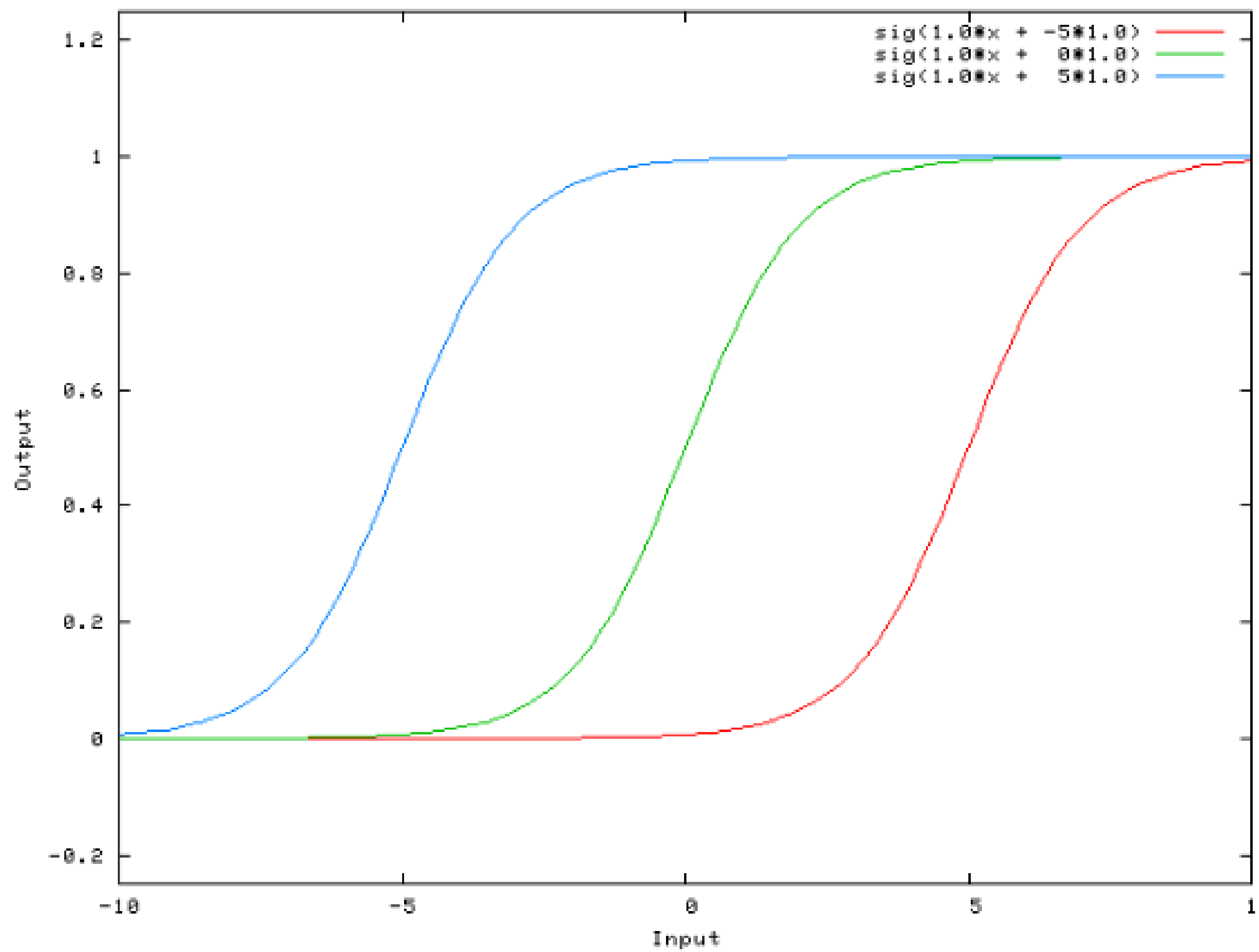




Bias in ANN



Input
x
Output
sig(w0 *x + w1 *1.0)
w0
w1
Bias
1.0



sig(1.0*x + -5*1.0)
sig(1.0*x + 0*1.0)
sig(1.0*x + 5*1.0)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
Output
-10
-5
0
5
10
Input

General Considerations

- What input attributes will be used to build the network?
- How will the network output be represented?
- How many hidden layers should the network contain?
- How many nodes should there be in each hidden layer?
- What conditions will terminate network training?

Weakness

- The biggest criticism of neural networks is that they lack the ability to explain their behavior.
- The algorithm is not guaranteed to converge to an optimal solution.
  - Manipulation of various learning parameter
- Neural networks can easily be over trained to the point of working well on the training data but poorly on test data.
  - Division of data into training and testing sets.

Evolving Neural Networks

Evolving Neural Networks

- Evolving parameters for the neural network training
- Evolving the features to be fed into the network
- Evolving the weights of the network with a predefined architecture
- Evolving the network architecture together with the weights

- Evolving Neural Networks. For the past decade, deep learning has… | by Riley Lazarou | Towards Data Science

Resources

- https://www.coursera.org/learn/machine-learning/lecture/du981/backpropagation-intuition
- https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
- https://scikit-neuralnetwork.readthedocs.io/en/latest/index.html