





























2

Name	Plot	Equation	Derivative (w/ respect to x)	Range	Order of continuity	Monotonic	Monotonic derivative	Approximates identity near the origin
Tanh		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	$C^\infty$	Yes	No	Yes
Score Nonlinearly (SQR) <sup>[20]</sup>		$f(x) = \begin{cases} 1 & : x > 2.0 \\ x - \frac{x^2}{4} & : 1 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & : -2.0 \leq x < 1 \\ -1 & : x < -2.0 \end{cases}$	$f'(x) = 1 \mp \frac{x}{2}$	$(-1, 1)$	$C^0$	Yes	No	Yes
Softplus <sup>[21]</sup>		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^x}$	$(0, \infty)$	$C^\infty$	Yes	Yes	No
SoftExponential <sup>[22]</sup>		$f(x, \alpha) = \begin{cases} \frac{\ln(-x) - \alpha}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^x - 1}{\alpha} + x & \text{for } \alpha > 0 \end{cases}$	$f'(x, \alpha) = \begin{cases} \frac{1}{1 - \alpha x} & \text{for } \alpha < 0 \\ 1 & \text{for } \alpha = 0 \\ e^x & \text{for } \alpha > 0 \end{cases}$	$(-\infty, \infty)$	$C^\infty$	Yes	Yes	Yes if $\alpha = 0$
SoftClipping <sup>[23]</sup>		$f(x, \alpha) = \frac{1}{\alpha} \log \frac{1 + e^{\alpha x}}{1 + e^{\alpha(1-x)}}$	$f'(x, \alpha) = \frac{1}{2} \operatorname{tanh}\left(\frac{\alpha}{2}\right) \operatorname{sech}\left(\frac{\alpha x}{2}\right) \operatorname{sech}\left(\frac{\alpha}{2}(1-x)\right)$	$(0, 1)$	$C^\infty$	Yes	No	No
Sine <sup>[24]</sup>		$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$	$C^\infty$	No	No	Yes
Sin		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{44x}{\pi} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{44}{\pi} & \text{for } x \neq 0 \end{cases}$	$[-2.0734, 1]$	$C^\infty$	No	No	No
Signif Linear Unit (SLU) <sup>[25]</sup> (aka SL <sup>[26]</sup> and Sweb-f <sup>[27]</sup> )			No	Approximates identity <sup>2</sup>				
Scored exponential linear unit (SELU) <sup>[28]</sup>		$f(x, \alpha) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ with $\lambda = 1.0507$ or $\alpha = 1.67326$	$f'(x, \alpha) = \lambda \begin{cases} \alpha e^x & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	$C^0$	Yes	No	No

2

2

2

2

2

PART III

2

S-shape rectified linear activation unit (SReLU) <sup>[2]</sup>		$f_{a,b,c,d}(x) = \begin{cases} a + b(x-d) & \text{for } x \leq d \\ x & \text{for } b < x < c \\ c + d(x-d) & \text{for } x \geq c \end{cases}$ $a, b, c, d, a, b$ are parameters	$f_{a,b,c,d}(x) = \begin{cases} a & \text{for } x \leq d \\ 1 & \text{for } b < x < c \\ c & \text{for } x \geq c \end{cases}$	$(-\infty, \infty)$	$C^0$	No	No	No
Rounded ReLU		$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$	$[1, \infty)$	$C^0$	Yes	Yes	No
Quadrant of leaky rectified linear unit (ReLU) <sup>[2]</sup>		$f(a, x) = \begin{cases} ax & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f(a, x) = \begin{cases} a & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	$C^0$	Yes	Yes	No
Parametric rectified linear unit (ReLU) <sup>[2]</sup>		$f(a, x) = \begin{cases} ax & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f(a, x) = \begin{cases} a & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)^2$	$C^0$	Yes if $a \geq 0$	Yes	Yes if $a = 1$
Logistic (aka Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f(x) = f(x)(1 - f(x))$	$(0, 1)$	$C^\infty$	Yes	No	No
Leaky rectified linear unit (Leaky ReLU) <sup>[2]</sup>		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	$C^0$	Yes	Yes	No
Inverse square root unit (ISRU) <sup>[2]</sup>		$f(x) = \frac{x}{\sqrt{1 + ax^2}}$	$f(x) = \left( \frac{1}{\sqrt{1 + ax^2}} \right)^2$	$\left( \frac{1}{\sqrt{a}}, \frac{1}{\sqrt{a}} \right)$	$C^\infty$	Yes	No	Yes
Inverse square root linear unit (ISRL) <sup>[2]</sup>		$f(x) = \begin{cases} \frac{x}{1+ax^2} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f(x) = \begin{cases} \left( \frac{1}{1+ax^2} \right)^2 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$\left( \frac{1}{\sqrt{a}}, \infty \right)$	$C^0$	Yes	Yes	Yes
Identity								
ReLU <sup>[2]</sup>		$f(x) = x \Phi(x) = x(1 + \text{erf}(x/\sqrt{2}))/2$	$f(x) = \Phi(x) + \text{erf}(x)$		$C^\infty$	No	No	No
Gaussian		$f(x) = e^{-x^2}$	$f(x) = -2xe^{-x^2}$	$(0, 1)$	$C^\infty$	No	No	No
Exponential linear unit (ELU) <sup>[2]</sup>		$f(a, x) = \begin{cases} a(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	$f'(a, x) = \begin{cases} f(a, x) + a & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$	$(-\infty, \infty)$	$\begin{cases} C^1 & \text{when } a = 1 \\ C^0 & \text{otherwise} \end{cases}$	Yes if $a \geq 0$	Yes if $0 \leq a < 1$	Yes if $a = 1$

2

2

Energy <sup>[2000]</sup> Soft		$f(x) = \frac{x}{1 +  x }$	$f'(x) = \frac{1}{(1 +  x )^2}$	$(-1, 1)$	$C^0$	Yes	No	Yes
Slope rectified linear unit (SReLU) <sup>[2]</sup>		$f(x) = \begin{cases} \text{ReLU}(x) & \text{if } \text{mod } 2 = 0 \\ -\text{ReLU}(-x) & \text{if } \text{mod } 2 \neq 0 \end{cases}$	$f'(x) = \begin{cases} \text{ReLU}'(x) & \text{if } \text{mod } 2 = 0 \\ -\text{ReLU}'(-x) & \text{if } \text{mod } 2 \neq 0 \end{cases}$	$(-\infty, \infty)$	$C^0$	Yes	Yes	No
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	$C^{-1}$	Yes	No	No
Soft Identity		$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2 + 1}} + 1$	$(-\infty, \infty)$	$C^\infty$	Yes	Yes	Yes
Arctan		$f(x) = \text{arctan}^{-1}(x) = \ln\left(\frac{x + \sqrt{x^2 + 1}}{x - \sqrt{x^2 + 1}}\right)$	$f'(x) = \frac{1}{\sqrt{x^2 + 1}}$	$(-\infty, \infty)$	$C^\infty$	Yes	No	Yes
Arctan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$\left( -\frac{\pi}{2}, \frac{\pi}{2} \right)$	$C^\infty$	Yes	No	Yes
Logistic piecewise linear (ReLU) <sup>[2]</sup>		$f(x) = \max(0, x) + \sum_{i=1}^k \max(0, -x + b_i)$	$f'(x) = W(x) = \sum_{i=1}^k \max(0, -x + b_i)$	$(-\infty, \infty)$	$C^0$	No	No	No

2

2

2





2.  $\sum_i \sum_j (E_{ij} - O_{ij})^2$

3.  $\Delta w = \eta (t - y) x_i$

4.  $\Delta w_{ij} = \eta x_i d_j$

**2+ 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.**

1.  $\Delta w = \eta (t - y) x_i$

2.  $\Delta w_{ij} = \eta x_i d_j$

3.  $\Delta w = \eta (t - y) x_i$

4.  $\Delta w_{ij} = \eta x_i d_j$

**2+ 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.**

1.  $\Delta w = \eta (t - y) x_i$

2.  $\Delta w_{ij} = \eta x_i d_j$

3.  $\Delta w = \eta (t - y) x_i$

**2+ 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.**

1.  $\Delta w = \eta (t - y) x_i$

2

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

(0.2) The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

$$w_{jk} = \begin{cases} \eta(y_k - w_{jk}) & \text{if node } j \text{ wins the competition} \\ 0 & \text{if node } j \text{ loses the competition} \end{cases}$$

2

2

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

, the output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

) The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

0. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

1. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

$$Err = \left( \sum_{k=1}^p \sum_{j=1}^m (o_j^{(k)} - y_j^{(k)})^2 \right) / p \cdot m$$

2

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

2. The output of a neuron is given by  $y = \max(0, w \cdot x)$ . If the input  $x$  is 1 and the weight  $w$  is 2, what is the output  $y$ ?

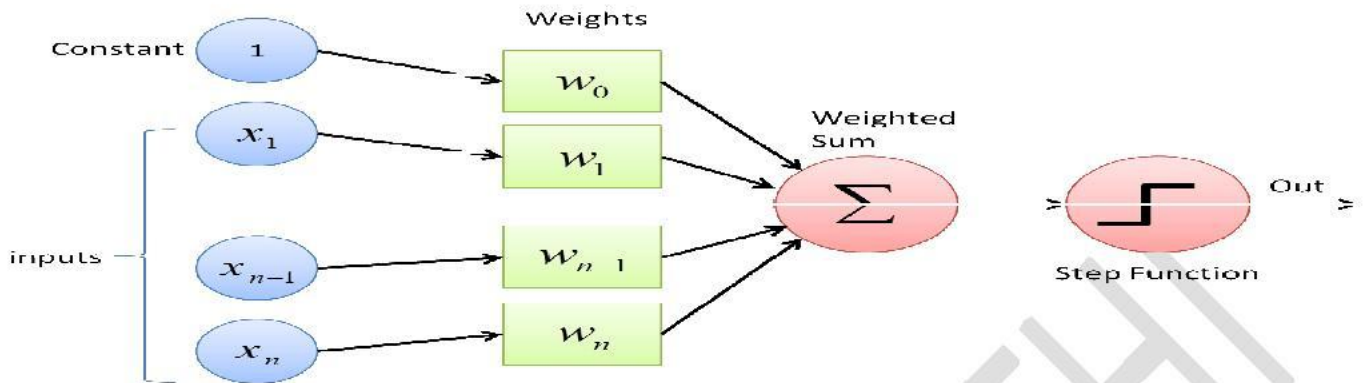
2

2

2

.....

2



2

2

3

- . (a) 2/3
- 2/3
- 2/3

6

- P1. Set a  $(n+1)$ -input,  $m$ -output perceptron. Randomize all network weights  $w_{ij}$ ,  $i=0,1,2,\dots,n$ ,  $j=1,2,\dots,m$ , to small numbers.
- P2. Apply an input feature vector  $x$  and calculate the net input signal  $u_j$  to each output perceptron neuron  $j$  using the standard formula:  

$$u_j = \sum (x_i \cdot w_{ij}), \text{ for } i = 0,1,2,\dots,n, \text{ for } j = 1,2,\dots,m, \text{ where } x_0=1 \text{ is the bias.}$$
- P3. Apply a hard-limited threshold activation function to the net input signals as follows:  

$$o_j = 1 \text{ if } u_j > \text{threshold, } o_j = 0 \text{ otherwise,}$$
 (Applying linear thresholding function is also possible).
- P4. Compute the error for each neuron by subtracting the actual output from the target output:  $\text{Err}_j = y_j - o_j$
- P5. Modify each weight  $w_{ij}$  by calculating its next value  $w_{ij}(t+1)$  from the previous one  $w_{ij}(t)$  and from the evaluated error  $\text{Err}_j$ :  

$$w_{ij}(t+1) = w_{ij}(t) + \alpha x_i \cdot \text{Err}_j,$$
 where:  $\alpha$  is a learning coefficient - a number between 0 and 1;
- P6. Repeat steps P2 through P5 until the error vector  $\text{Err}$  is sufficiently low, i.e. the perceptron goes into a convergence.

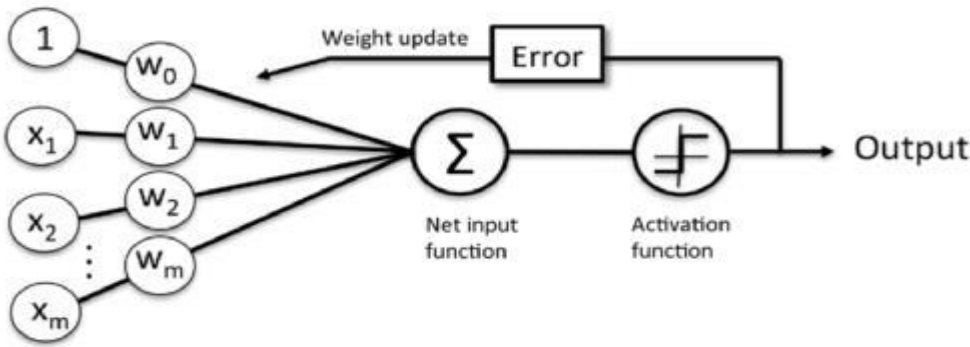
2

2 2 2 2 2 2 2 2 2 2

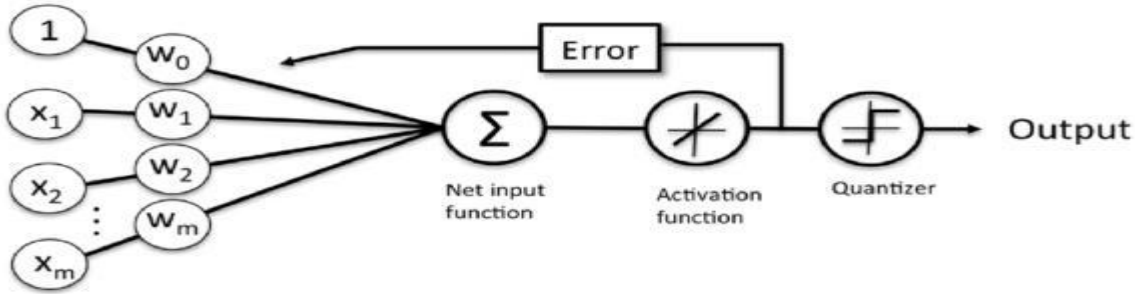
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

2





### Perceptron



### Adaptive linear neuron

Adaptive linear neuron is a type of neuron that uses a linear activation function. It is used in the perceptron learning rule. The output of the neuron is compared with the target value to calculate the error. The error is used to update the weights of the neuron.

The perceptron learning rule is a supervised learning algorithm. It is used to train a single-layer perceptron. The algorithm starts with a set of input-output pairs. The weights of the neuron are initialized to random values. The algorithm then iterates over the training data, calculating the error for each input-output pair. The weights are updated based on the error. The algorithm stops when the error is zero for all training data.

#### 2. Error function

The error function is a mathematical function that is used to calculate the error of a neuron. It is defined as the square of the difference between the target value and the output of the neuron. The error function is used to calculate the error for each input-output pair in the training data. The error is then used to update the weights of the neuron.

$$J(\mathbf{w}) = \frac{1}{2} \sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2 \quad \text{output}^{(i)} \in \mathbb{R}$$

The error function is a convex function. This means that it has a single global minimum. This property is useful for finding the optimal weights for the neuron.

The error function is also differentiable. This means that we can use gradient descent to find the optimal weights for the neuron.

Gradient descent is an optimization algorithm that is used to find the minimum of a function. It starts with an initial guess for the weights and iteratively updates them in the direction of the negative gradient of the error function. The algorithm stops when the error is minimized.

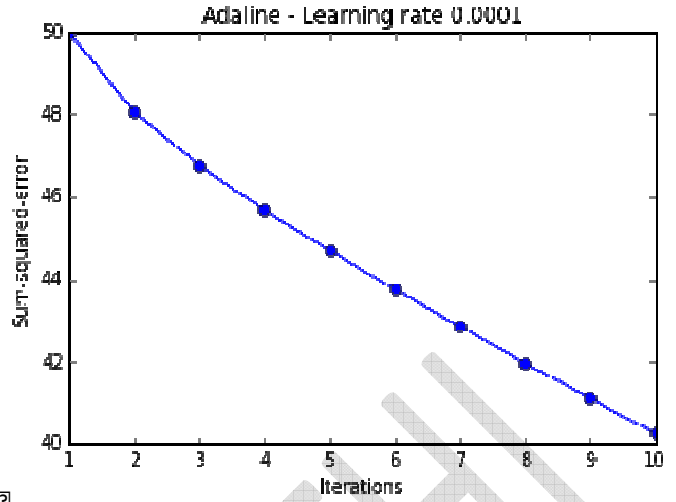
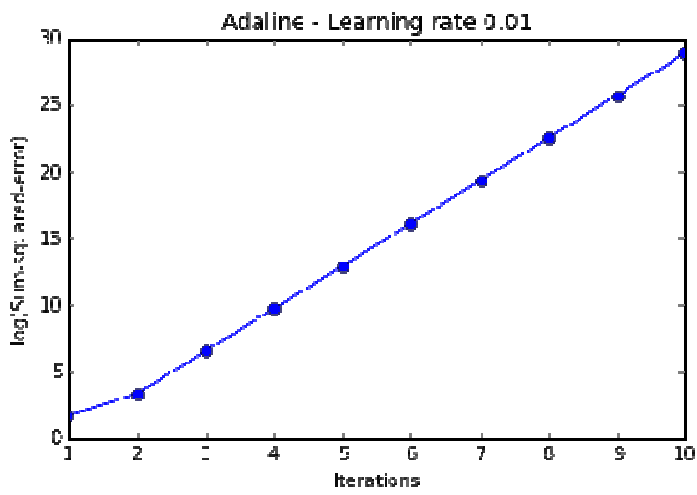
The error function is a key component of the perceptron learning rule. It is used to calculate the error for each input-output pair and to update the weights of the neuron.

The error function is also used in other types of neural networks, such as multi-layer perceptrons and convolutional neural networks.

The error function is a simple but powerful tool for training neural networks. It is used to calculate the error for each input-output pair and to update the weights of the neuron.

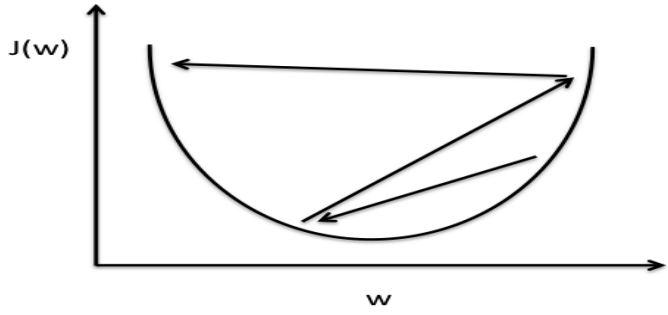
The error function is a key component of the perceptron learning rule. It is used to calculate the error for each input-output pair and to update the weights of the neuron.



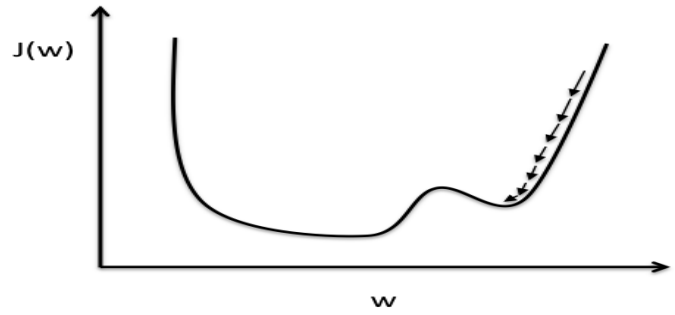


When the learning rate is too large, the error function is not smooth, and the algorithm oscillates around the minimum. When the learning rate is too small, the algorithm converges very slowly.

- 4. The error function is not smooth, and the algorithm oscillates around the minimum.
- 5. The error function is not smooth, and the algorithm oscillates around the minimum.



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

The error function is not smooth, and the algorithm oscillates around the minimum. When the learning rate is too small, the algorithm converges very slowly.

$$x_{j,std} = \frac{x_j - \mu_j}{\sigma_j}$$

The error function is not smooth, and the algorithm oscillates around the minimum. When the learning rate is too small, the algorithm converges very slowly.

1. The error function is not smooth, and the algorithm oscillates around the minimum. When the learning rate is too small, the algorithm converges very slowly.

The error function is not smooth, and the algorithm oscillates around the minimum. When the learning rate is too small, the algorithm converges very slowly.

2

ପ୍ରକାରର ସମସ୍ତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।

2

**8 ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।**

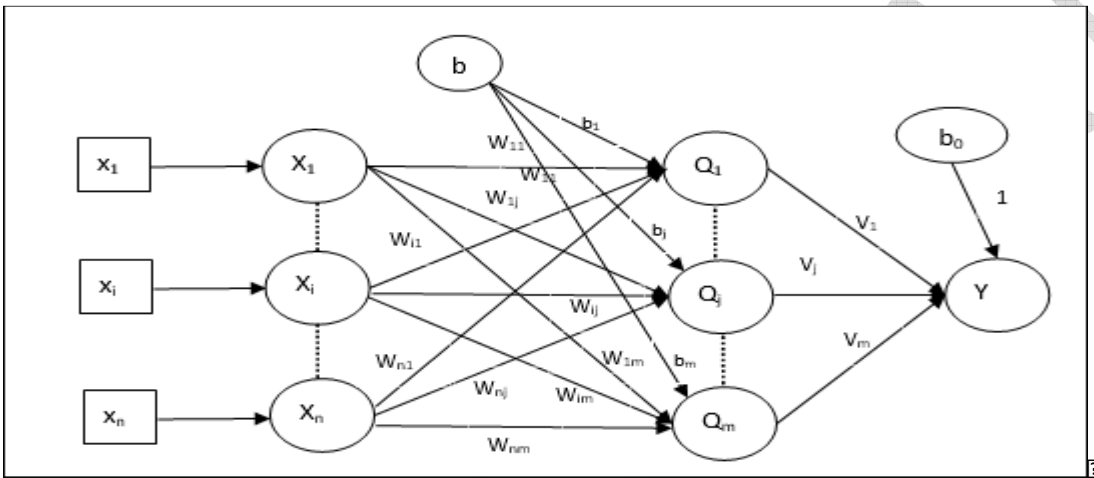
1 ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।

- ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।
- ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।
- ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।
- ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।

ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।

ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।

2



2

2

2

2

**8 ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।**

ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।

2

ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।

2

ପ୍ରକାରର ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବା ପରେ କେବଳ ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ । ସମ୍ପର୍କିତ ସମ୍ପର୍କ ସମ୍ପର୍କିତ ହେବ ।

2

2

2



2

10) ... (D) ...

2

2) ...

10) ...

. ...

2

2) ...

2

10) ...

2

2) ...

2) ...

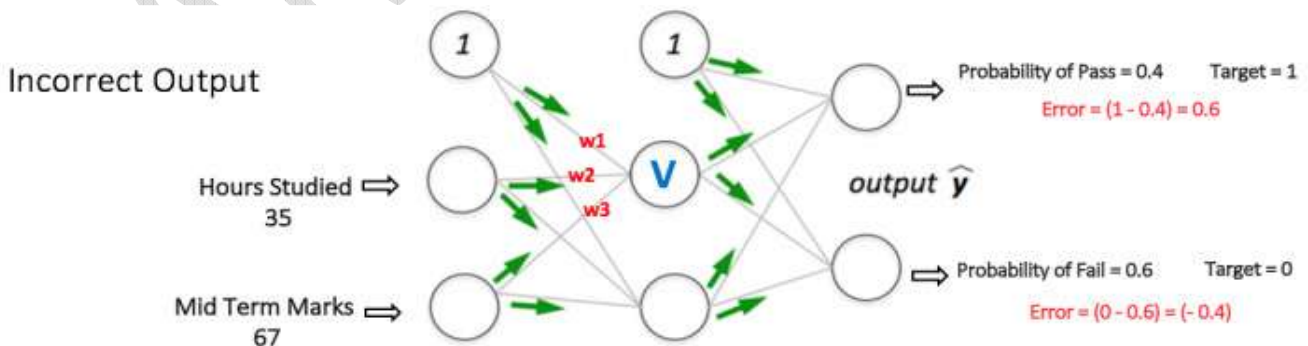
- ...
- ...

2) ...

( ... )

2

( ... )



2) ...

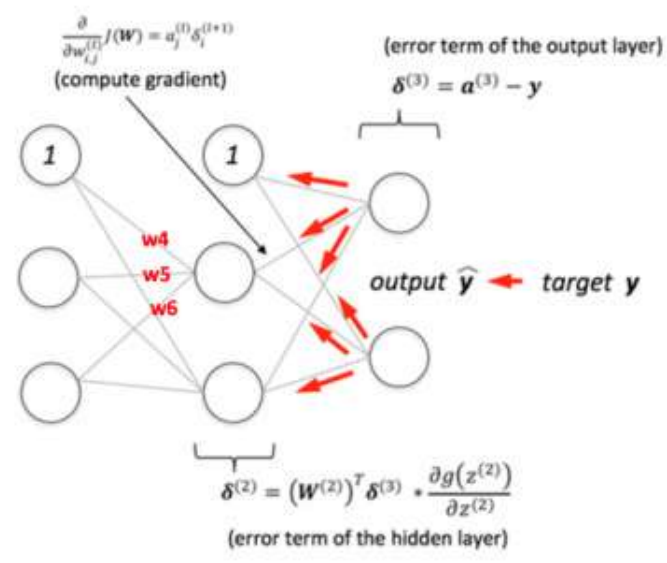
2) ...

2

2  
2

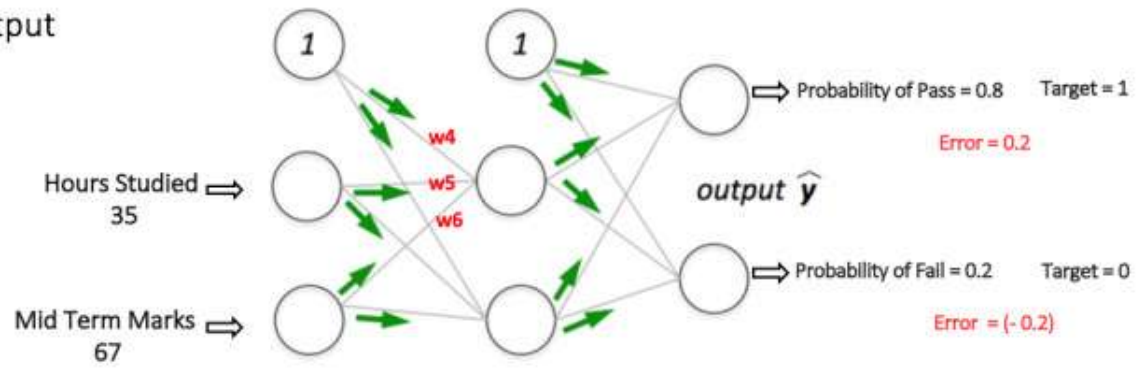
(The diagram shows a neural network with 6 weights. The output is compared to the target. The error term of the output layer is  $\delta^{(3)} = a^{(3)} - y$ . The error term of the hidden layer is  $\delta^{(2)} = (W^{(2)})^T \delta^{(3)} * \frac{\partial g(z^{(2)})}{\partial z^{(2)}}$ . The gradient is computed as  $\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$ . Backpropagation + Weights Adjusted.

Backpropagation  
+  
Weights Adjusted



The diagram illustrates the backpropagation process in a neural network. It shows the flow of error terms from the output layer back to the hidden layer. The error term of the output layer is  $\delta^{(3)} = a^{(3)} - y$ . The error term of the hidden layer is  $\delta^{(2)} = (W^{(2)})^T \delta^{(3)} * \frac{\partial g(z^{(2)})}{\partial z^{(2)}}$ . The gradient is computed as  $\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$ . Backpropagation + Weights Adjusted.

Correct Output



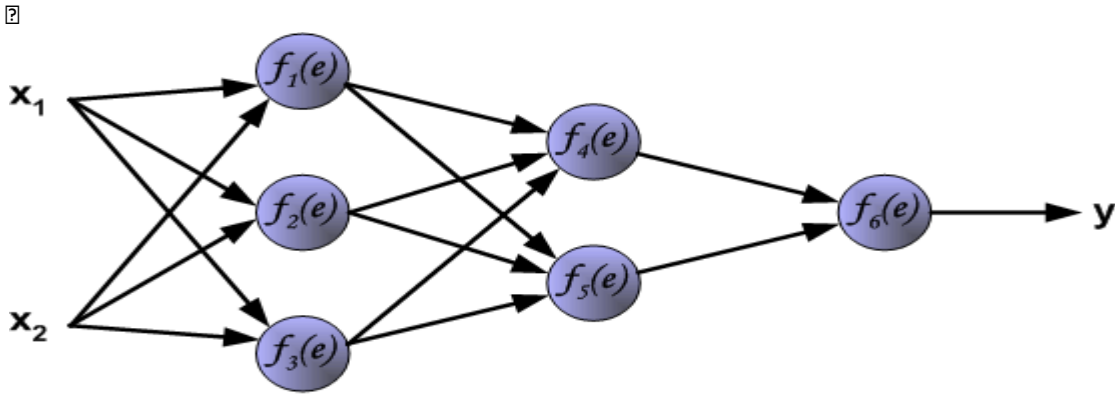
The diagram shows the correct output of the neural network. The input is Hours Studied = 35 and Mid Term Marks = 67. The output is Probability of Pass = 0.8 and Probability of Fail = 0.2. The target is 1 for Pass and 0 for Fail. The error is 0.2 for the first output and -0.2 for the second output.

The diagram shows the correct output of the neural network. The input is Hours Studied = 35 and Mid Term Marks = 67. The output is Probability of Pass = 0.8 and Probability of Fail = 0.2. The target is 1 for Pass and 0 for Fail. The error is 0.2 for the first output and -0.2 for the second output.

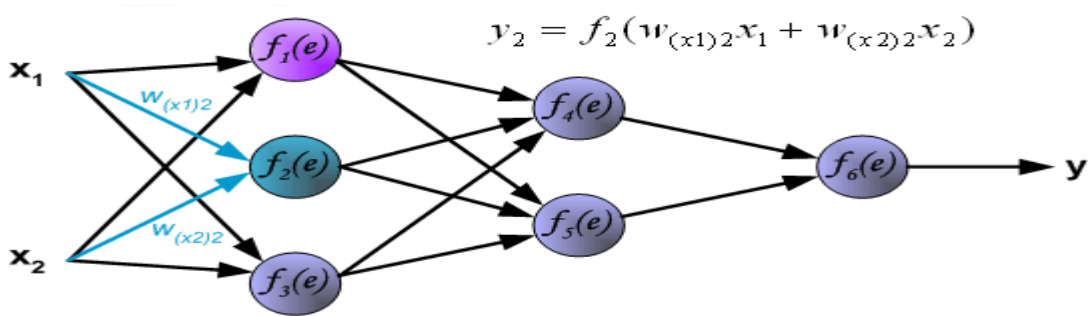
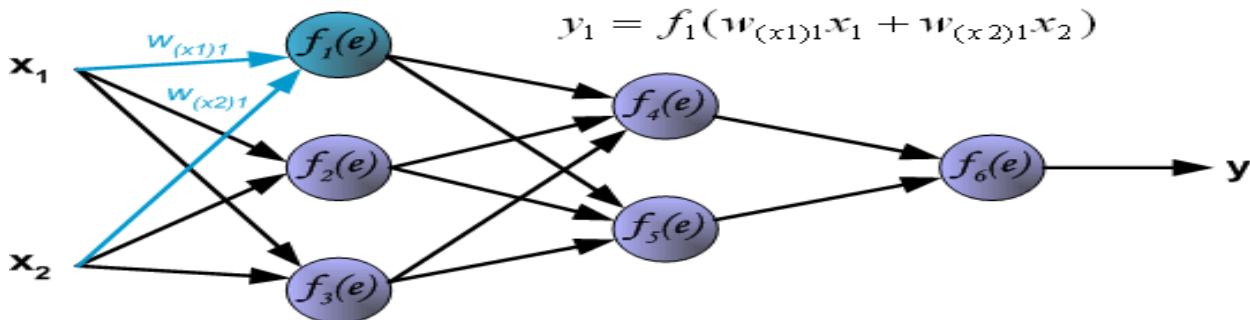
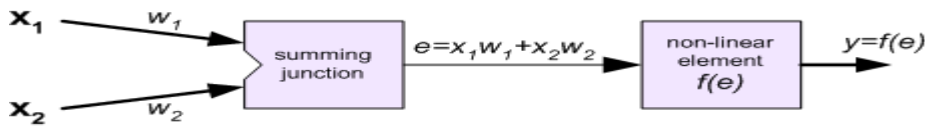
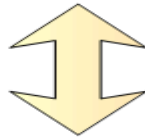
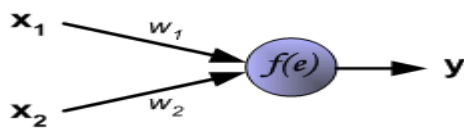
2  
2

The diagram shows the correct output of the neural network. The input is Hours Studied = 35 and Mid Term Marks = 67. The output is Probability of Pass = 0.8 and Probability of Fail = 0.2. The target is 1 for Pass and 0 for Fail. The error is 0.2 for the first output and -0.2 for the second output.

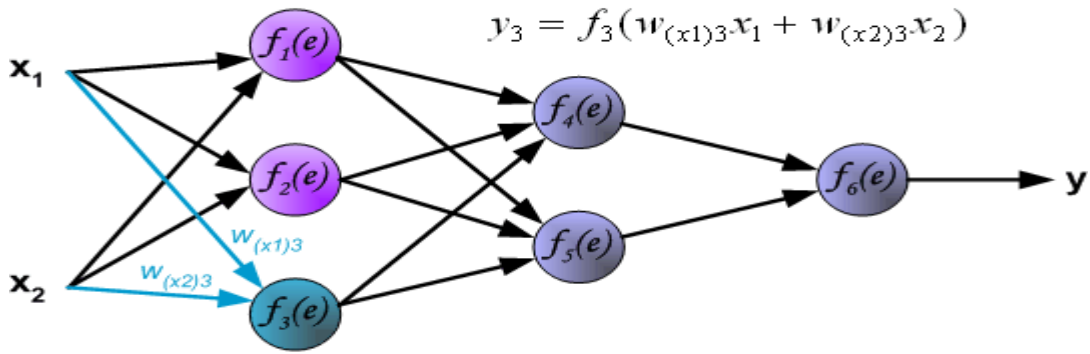
2



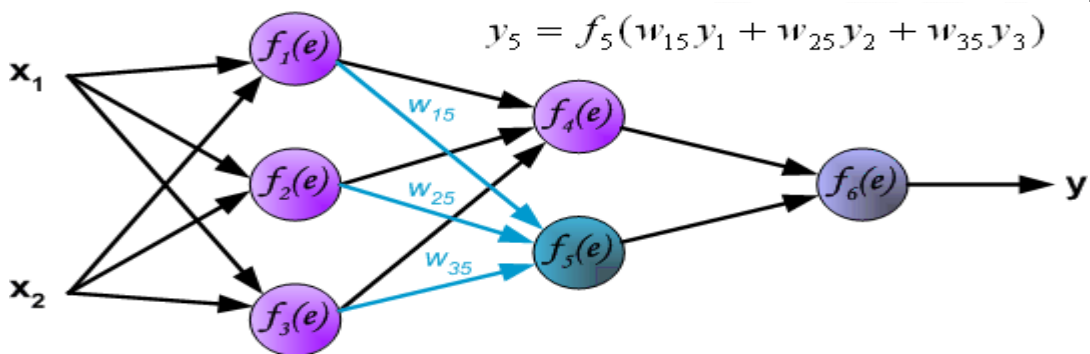
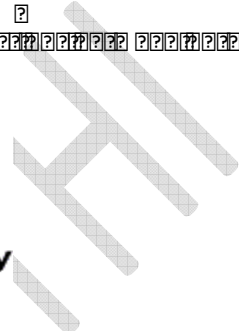
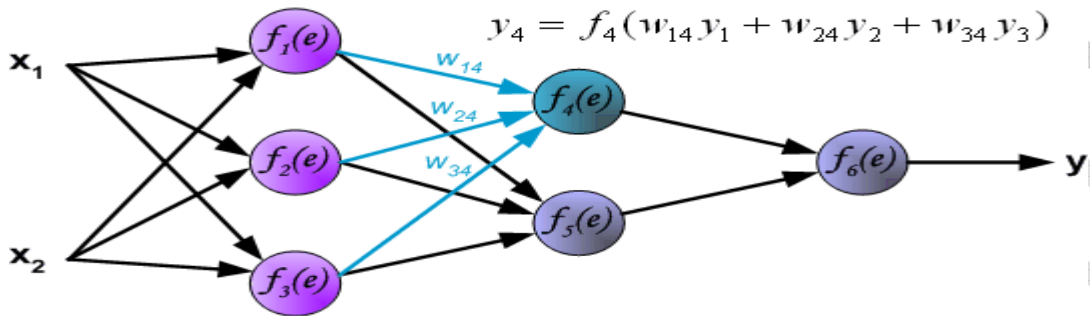
-



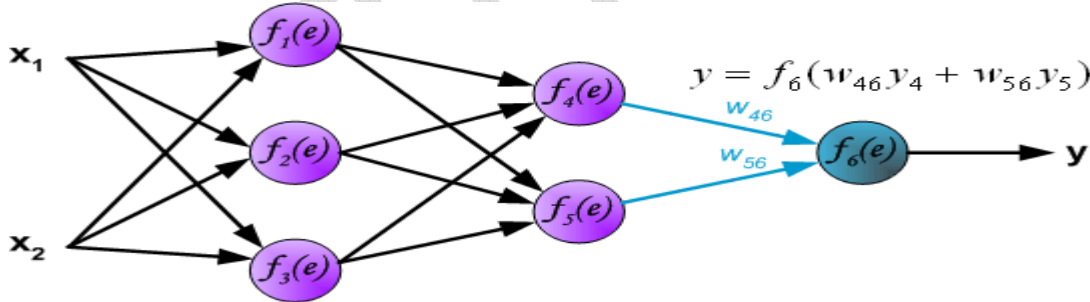
2



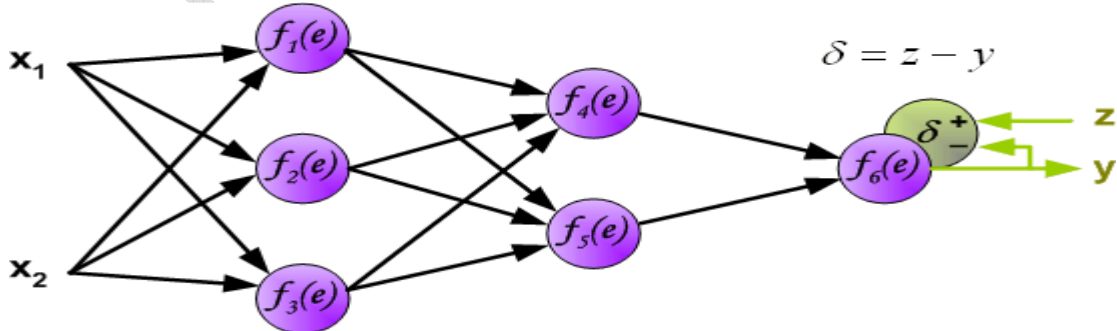
3



3



0

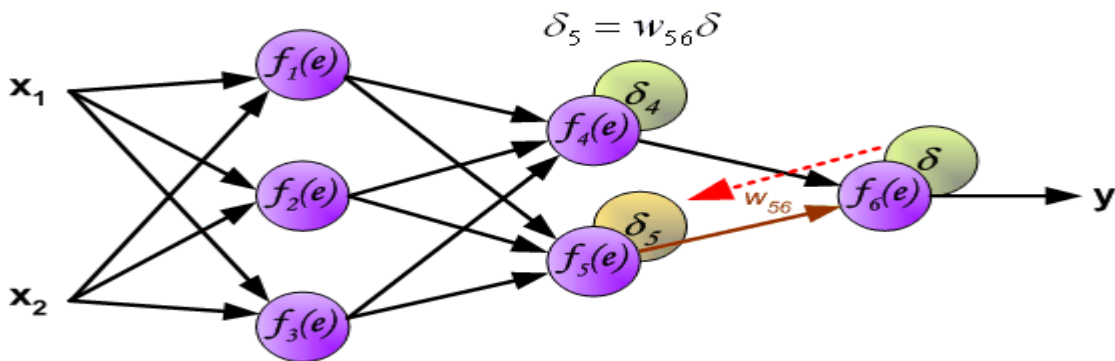
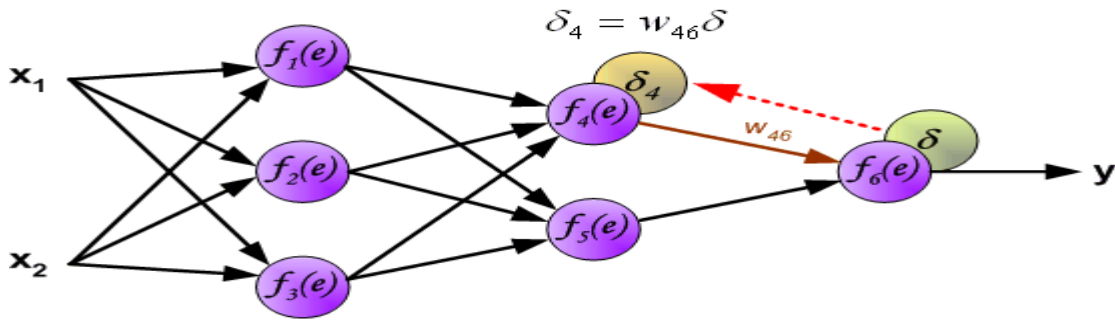


2

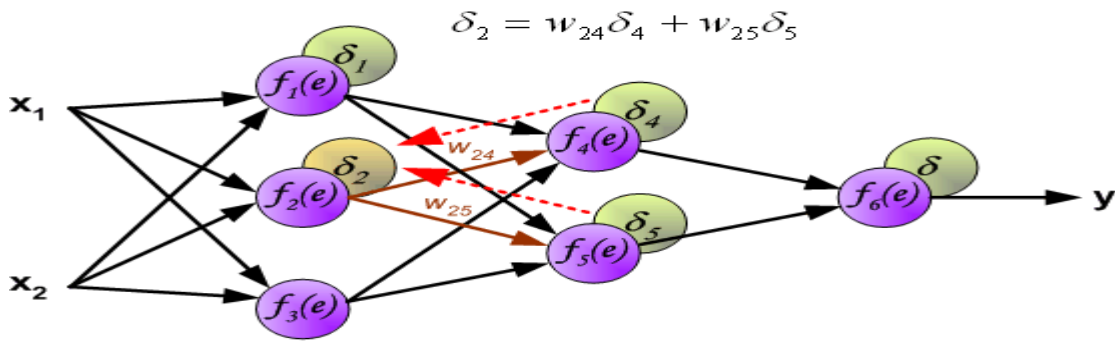
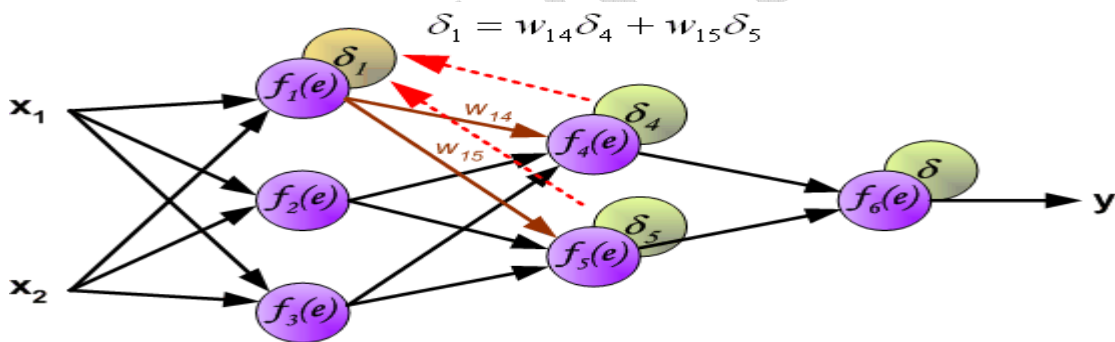
2

2

QUESTION 2: Explain the backpropagation algorithm for a feedforward neural network. (10 marks)



QUESTION 3: Explain the backpropagation algorithm for a recurrent neural network. (10 marks)



2





2

Forward pass:

BF1. Apply an input vector  $x$  and its corresponding output vector  $y$  (the desired output).

BF2. Propagate forward the input signals through all the neurons in all the layers and calculate the output signals.

BF3. Calculate the  $Err_j$  for every output neuron  $j$  as for example:  
 $Err_j = y_j - o_j$ , where  $y_j$  is the  $j$ th element of the desired output vector  $y$ .

Backward pass:

BB1. Adjust the weights between the intermediate neurons  $i$  and output neurons  $j$  according to the calculated error:

$$\Delta w_{ij}(t+1) = \text{lr} \cdot o_j(1 - o_j) \cdot Err_j + \text{momentum} \cdot \Delta w_{ij}(t)$$

BB2. Calculate the error  $Err_i$  for neurons  $i$  in the intermediate layer:

$$Err_i = \sum Err_j \cdot w_{ij}$$

BB3. Propagate the error back to the neurons  $k$  of lower level:

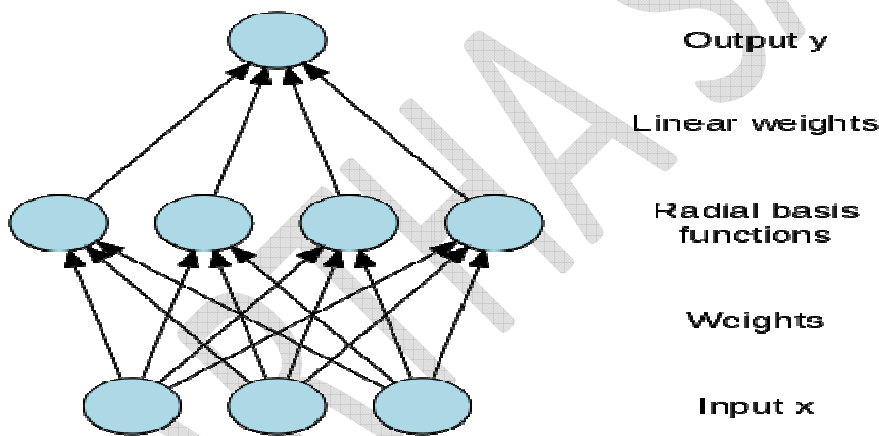
$$\Delta w_{ki}(t+1) = \text{lr} \cdot o_i(1 - o_i) \cdot Err_i \cdot x_k + \text{momentum} \cdot \Delta w_{ki}(t)$$

2

2

2

5



2

2

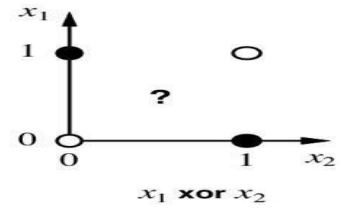
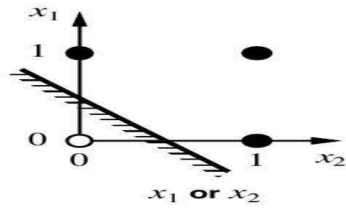
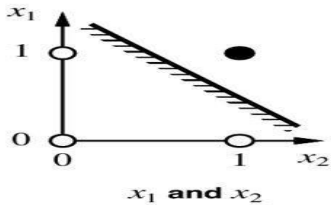
-

-

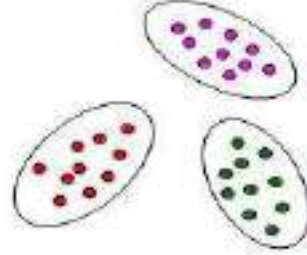
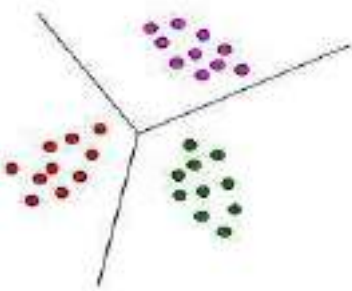
2

2

# Linear separability



- [unreadable]
- [unreadable]
- [unreadable]

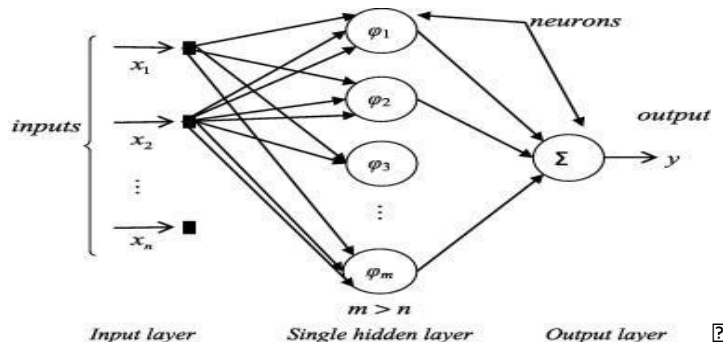
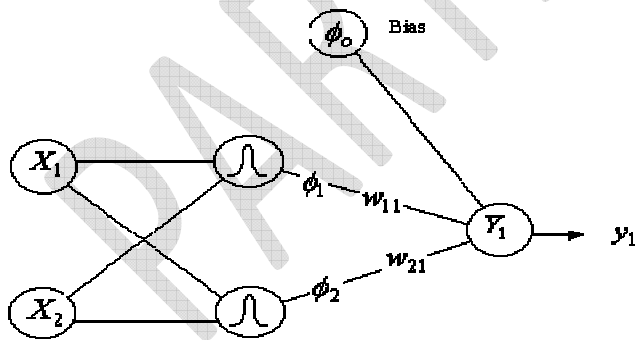


MLP

RBF

- [unreadable]
- [unreadable]

[unreadable]



- [unreadable]
- [unreadable]

[unreadable]

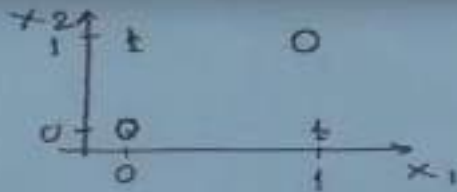
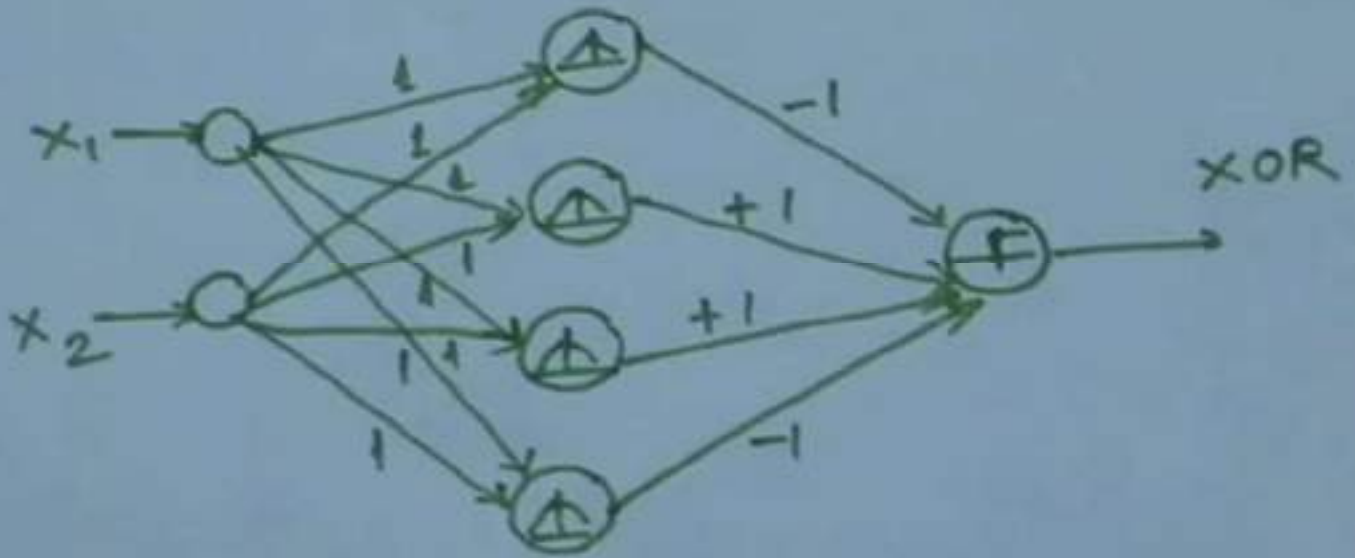
[unreadable]

- [unreadable]
- [unreadable]

[unreadable]







$P=2$

- $\phi_1 \rightarrow t_1 = (0,0) ; \sigma_1 = 1$
- $\phi_2 \rightarrow t_2 = (0,1) ; \sigma_2 = 1$
- $\phi_3 \rightarrow t_3 = (1,0) ; \sigma_3 = 1$
- $\phi_4 \rightarrow t_4 = (1,1) ; \sigma_4 = 1$

$$\phi_1(x) = e^{-\frac{\|x-t_1\|^2}{2}}$$

$$\phi_2(x) = e^{-\frac{\|x-t_2\|^2}{2}}$$

$$\phi_3(x) = e^{-\frac{\|x-t_3\|^2}{2}}$$

$$\phi_4(x) = e^{-\frac{\|x-t_4\|^2}{2}}$$

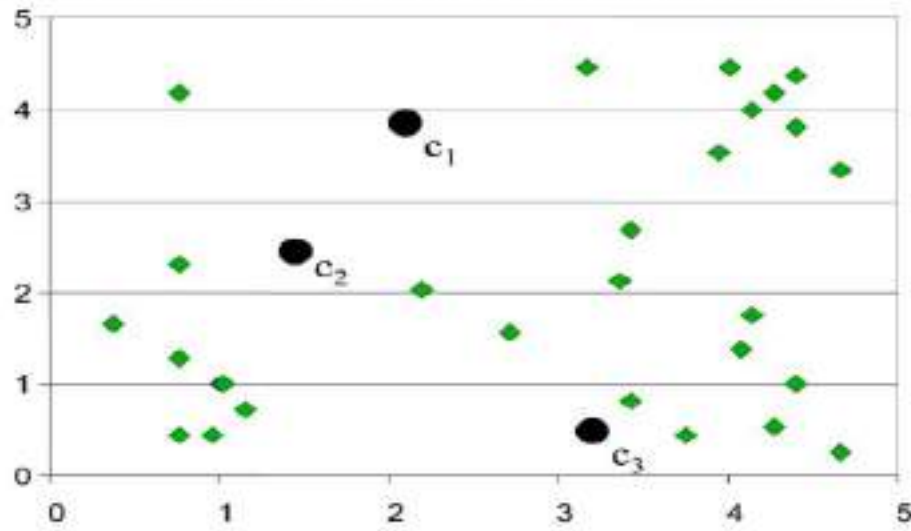
Input	$\phi_1$	$\phi_2$	$\phi_3$	$\phi_4$	$\sum W_i \phi_i$	Output
0 0	1.0	0.6	0.6	0.4	-0.2	0
0 1	0.6	1.0	0.4	0.6	0.2	1
1 0	0.6	0.4	1.0	0.6	0.2	1
1 1	0.4	0.6	0.6	1.0	-0.2	0
	-1	+1	+1	-1		





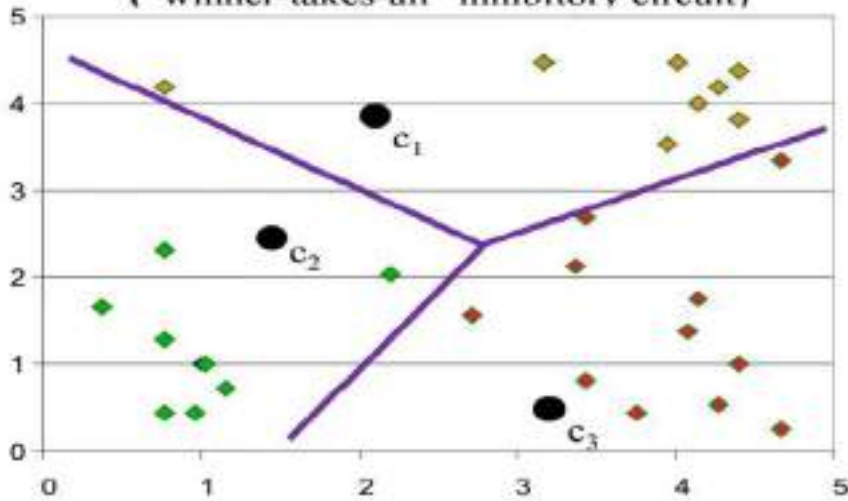
# step 1

Randomly initialize the cluster centers (synaptic weights)



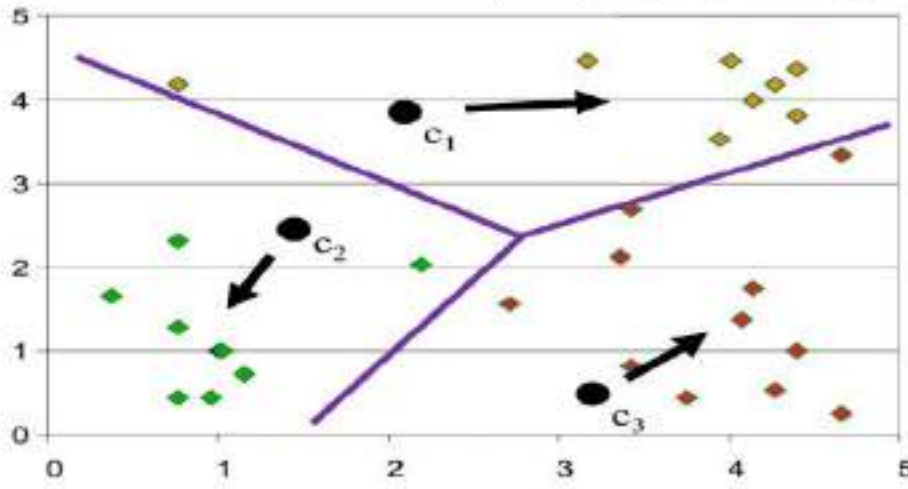
# step 2

Determine cluster membership for each input ("winner-takes-all" inhibitory circuit)

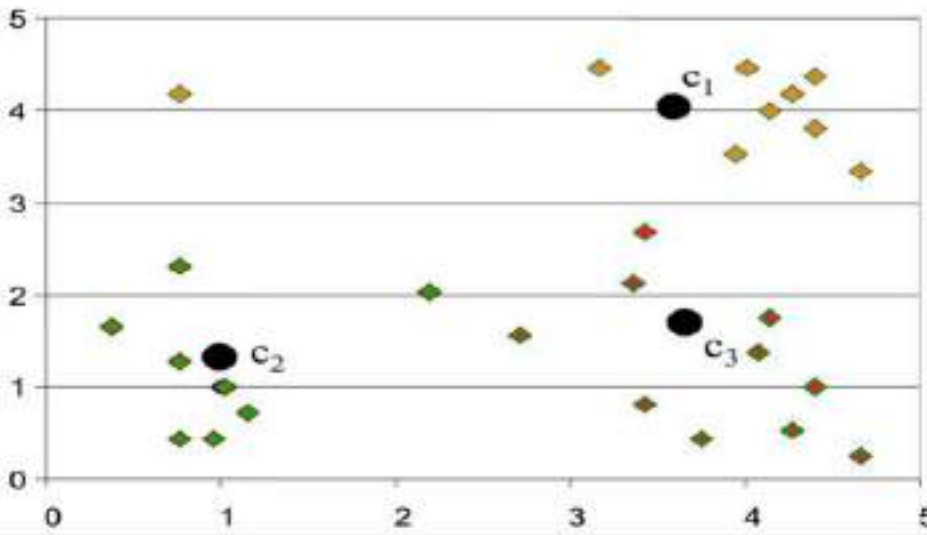


### step 3

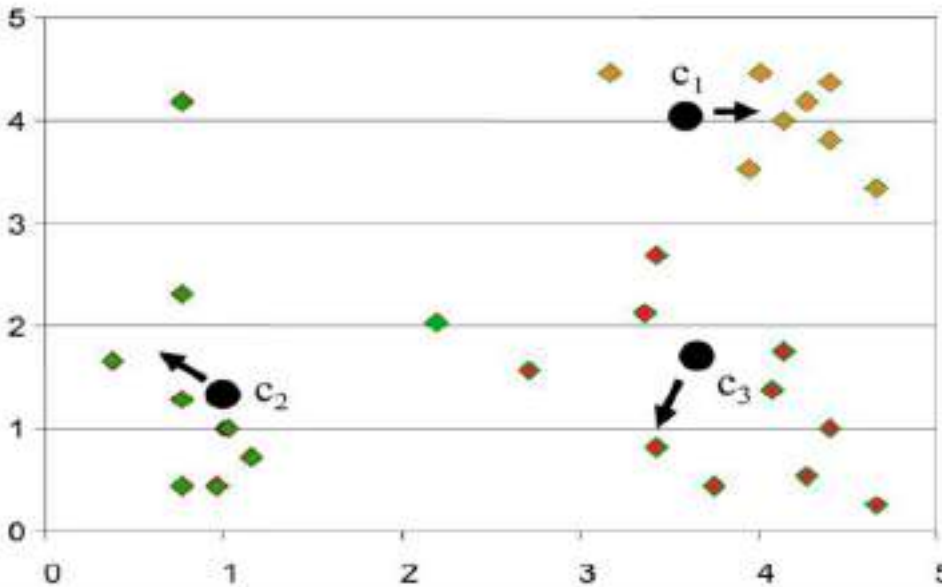
Re-estimate cluster centers (adapt synaptic weights)



Result of first iteration



Second iteration

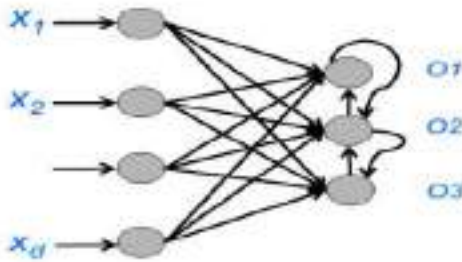






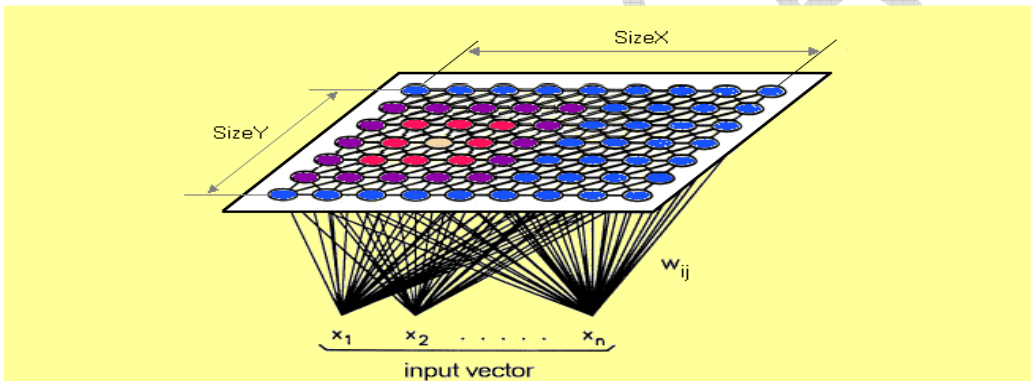
**A form of unsupervised training where output units are said to be in competition for input patterns**

- During training, the output unit that provides the highest activation to a given input pattern is declared the winner and is moved closer to the input pattern, whereas the rest of the neurons are left unchanged
- This strategy is also called winner-take-all since only the winning neuron is updated
  - Output units may have lateral inhibitory connections so that a winner neuron can inhibit others by an amount proportional to its activation level



**Figure 18**

Figure 18 illustrates a simple neural network where output units are in competition for input patterns. The network consists of input nodes  $x_1, x_2, \dots, x_d$  and output nodes  $O_1, O_2, O_3$ . The output nodes are connected to each other, representing lateral inhibitory connections. This setup allows for a winner-take-all strategy where the most activated output unit inhibits others.



**Figure 18** illustrates a Self-Organizing Map (SOM) with dimensions  $SizeX$  and  $SizeY$ , and an input vector  $x_1, x_2, \dots, x_n$ .

**9.7 Self-Organizing Map (SOM)**

The Self-Organizing Map (SOM) is a type of artificial neural network that is used for data visualization and dimensionality reduction. It consists of a grid of nodes, each representing a point in the input space. The nodes are arranged in a 2D grid with dimensions  $SizeX$  and  $SizeY$ . The input vector  $x_1, x_2, \dots, x_n$  is fed into the network, and the weights  $w_{ij}$  are updated based on the input patterns. The SOM is trained using an unsupervised learning algorithm called the Kohonen learning rule. The SOM is used to visualize the input patterns and to find clusters of similar patterns. The SOM is also used for dimensionality reduction, where the high-dimensional input patterns are mapped onto the 2D grid of nodes.

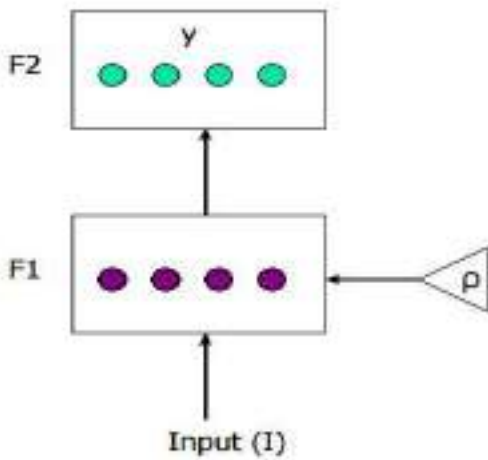






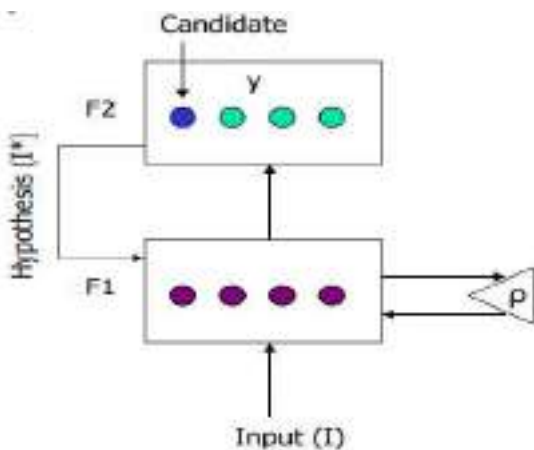
2  
 ????? 2) ?? 22 ??22 222??3??222222222222??

2  
 -)?? 2??|2



Step 1: Send input from the F1 layer to F2 layer for processing. The first node within the F2 layer is chosen as the closest match to the input and a hypothesis is formed. This hypothesis represents what the node will look like after learning has occurred, assuming it is the correct node to be updated.

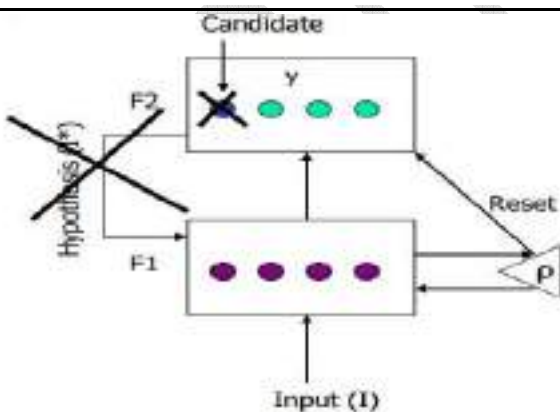
F1 (short term memory) contains a vector of size M, and there are N nodes within F2. Each node within F2 is a vector of size M. The set of nodes within F2 is referred to as "y".



Step 2: Once the hypothesis has been formed, it is sent back to the F1 layer for matching. Let  $T_j(I^*)$  represent the level of matching between I and  $I^*$  for node j. Then:

$$T_j(I^*) = \frac{I \wedge I^*}{I} \quad \text{where } A \wedge B = \min(A, B)$$

If  $T_j(I^*) \geq \rho$  then the hypothesis is accepted and assigned to that node. Otherwise, the process moves on to Step 3.



Step 3: If the hypothesis is rejected, a "reset" command is sent back to the F2 layer. In this situation, the  $j^{\text{th}}$  node within F2 is no longer a candidate so the process repeats for node  $j+1$ .









