Deep Learning - Introduction

Dr. Priyank Thakkar
Associate Professor
CSE Department
Institute of Technology
Nirma University

These slides are not original and have been prepared from various sources for teaching purpose - Priyank Thakkar

Syllabus

Syllabus
UNIT 1: Review of Visual Perception and Artificial Neural Networks Overview of Computer Vision, Preprocessing Images for Recognition, Feature Engineering for Conventional Image Classification, K-Nearest Neighbor, Linear Classification, Gradient Descent, Feed Forward Neural Network, Backpropagation, Unstable Gradient Problem
UNIT 2: Convolutional Neural Networks Introduction to Deep Supervised Learning, Convolution & Pooling, Dropout, LeNet, AlexNet, ZFNet, VGGNet, GoogleNet, ResNet and other State-of-the-art CNNs
UNIT 3: Transfer Learning Transfer Learning Scenarios, Applications of Transfer Learning, Transfer Learning Methods, Fine Tuning and Data Augmentation, Related Research Areas,
UNIT 4: Convolutional Neural Networks in Action for Computer Vision Semantic Segmentation, Object Detection, Instance Segmentation, Feature Visualization and Inversion, DeepDream and Style Transfer, Highway Networks, Image Recognition, Real Time CNN, Stereo Siamese Networks, Depth from Single Image, Image Generation, Domain Adaptation
UNIT 5: Review of other Deep Neural Networks Auto Encoders, Recurrent and Recursive Neural Networks, Boltzmann and Restricted Boltzmann Machine
UNIT 6: Practical Deep Learning and Case Studies Various Frameworks such as DIGITS, TensorFlow, Caffe and Theano, 2-3 Case Studies based on the Latest Developments in the Field

Referencess

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press

2. Adam Gibson, Josh Patterson, Deep Learning, O'Reilly Media, Inc.

3. Duda, R.O., Hart, P.E., and Stork, D.G., Pattern Classification, Wiley.

4. Theodoridis, S. and Koutroumbas, K., Pattern Recognition. Academic Press

5. Russell, S. and Norvig, N. Artificial Intelligence: A Modern Approach. Prentice Hall Series in Artificial Intelligence

References

6. Bishop, C. M. Neural Networks for Pattern Recognition. Oxford University Press.

7. Hastie, T., Tibshirani, R. and Friedman, J. The Elements of Statistical Learning, Springer

8. Koller, D. and Friedman, N. Probabilistic Graphical Models. MIT Press

9. Richard Szeliski, Computer Vision: Algorithms and Applications, Springer

10. Research Papers and Web Links

Blog and Course Site

Blog Link:

https://it7f4pbt.wordpress.com

Course Site:

https://sites.google.com/a/nirmauni.ac.in/it7f4---deep-learning/

Teaching & Evaluation Scheme

Teaching Scheme:

Theory	Tutorial	Practical	Credits
3	0	2	4

Evaluation Scheme:

	LPW	SEE	CE
Exam Duration	Continuous Evaluation + 2 Hrs. End Semester Exam	3.0 Hrs.	Continuous Evaluation
Component Weightage	0.2	0.4	0.4

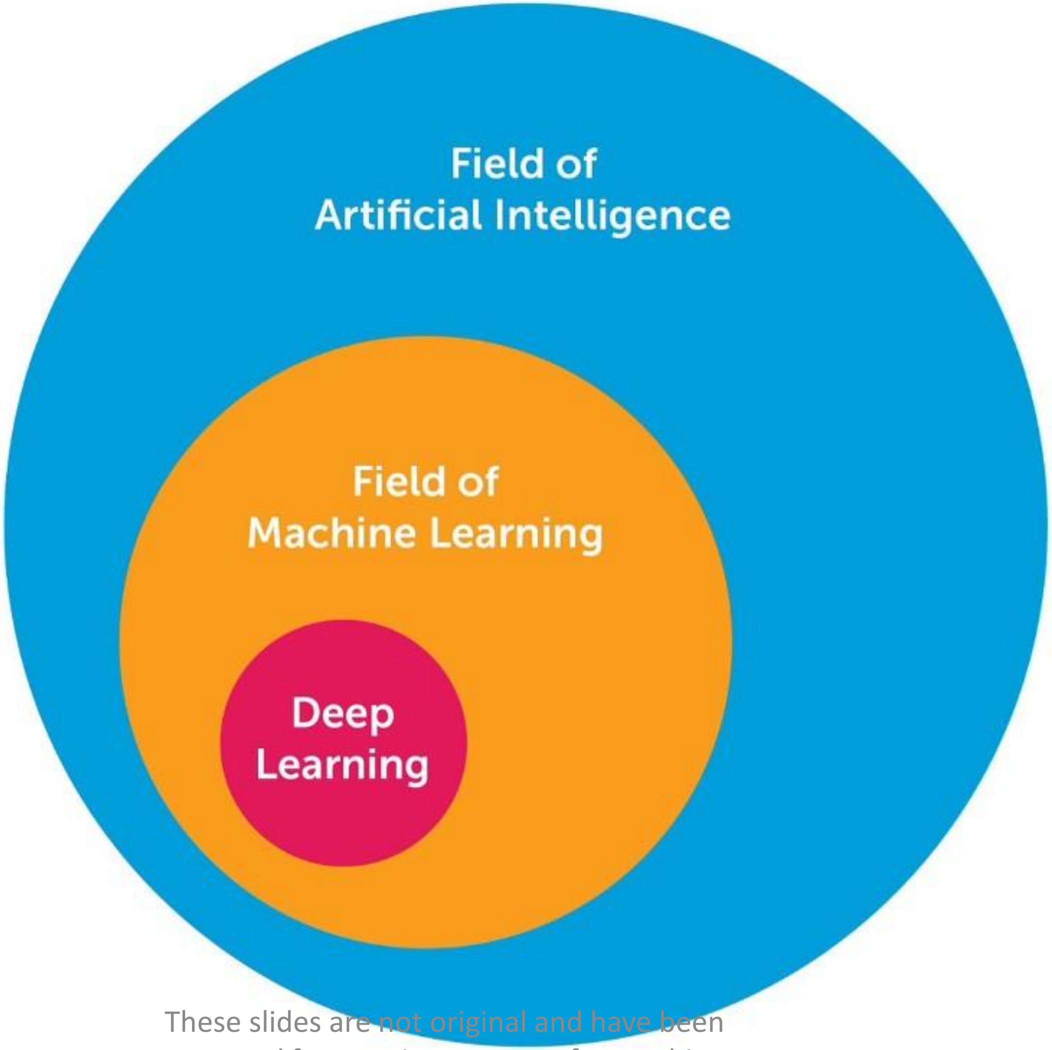
Teaching & Evaluation Scheme

Breakup of CE

	Unit 1	Unit 2	Unit 3
Exam	Class Test	Sessional Exam	Assignments
Inter Component Weightage	0.3	0.4	0.3
Numbers	1	1	2
Marks of Each	30	40	15

Introduction

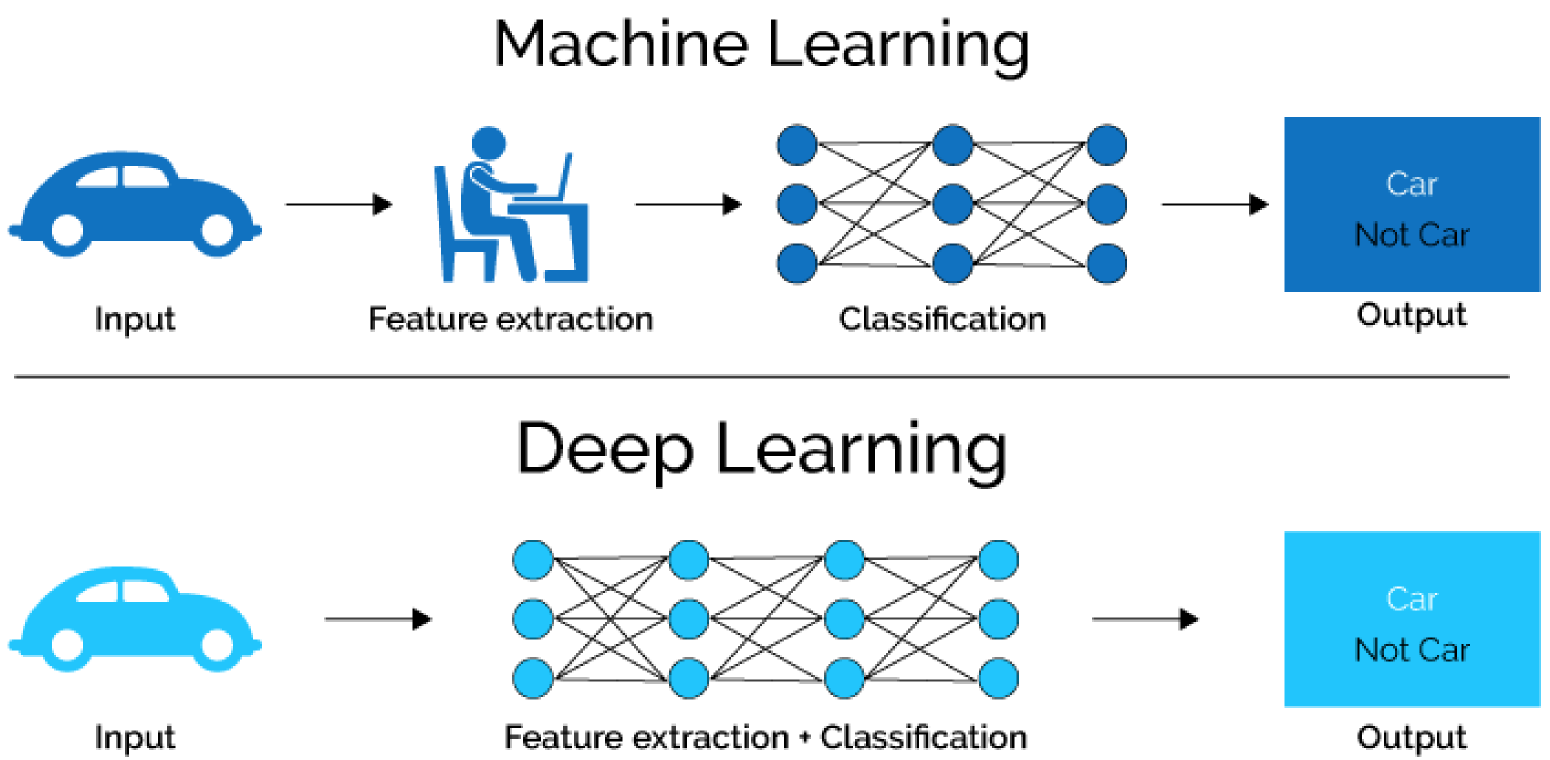
➢ AI, ML and DL



Source: [1]

Introduction

➢ Machine Learning vs Deep Learning



Source: [2]

Major Architectures of Deep Networks

- Four Major Architectures:
  - Unsupervised Pretrained Networks (UPNs)
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks
  - Recursive Neural Networks

Major Architectures of Deep Networks

- Four Major Architectures:
  - Unsupervised Pretrained Networks (UPNs)
    - Autoencoders
    - Deep Belief Networks (DBNs)
    - Generative Adversarial Networks (GANs)

    - Use Cases:
      - Feature Extraction
      - Initialization
      - Synthesizing

Source: [3]

Major Architectures of Deep Networks

- Four Major Architectures:
  - Convolutional Neural Networks (CNNs)
    - Lenet-5
    - AlexNet
    - VGGNet
    - GoogleNet (Inception)
    - ResNet
    - ResNext
    - DenseNet
    - RCNN (Region Based CNN)
    - YOLO (You Only Look Once)
    - SqueezeNet
    - SegNet

Major Architectures of Deep Networks

- Four Major Architectures:
  - Convolutional Neural Networks (CNNs)
  - Use Cases:
    - Computer Vision
    - Natural Language Processing

Major Architectures of Deep Networks

- Four Major Architectures:
  - Recurrent Neural Networks
    - Hopfield Network
    - Long Short-Term Memory (LSTM)
    - Gated Recurrent Unit (GRU)
  - Use Cases:
    - Sentiment Classification
    - Image Captioning
    - Language Translation
    - Video Captioning

Major Architectures of Deep Networks

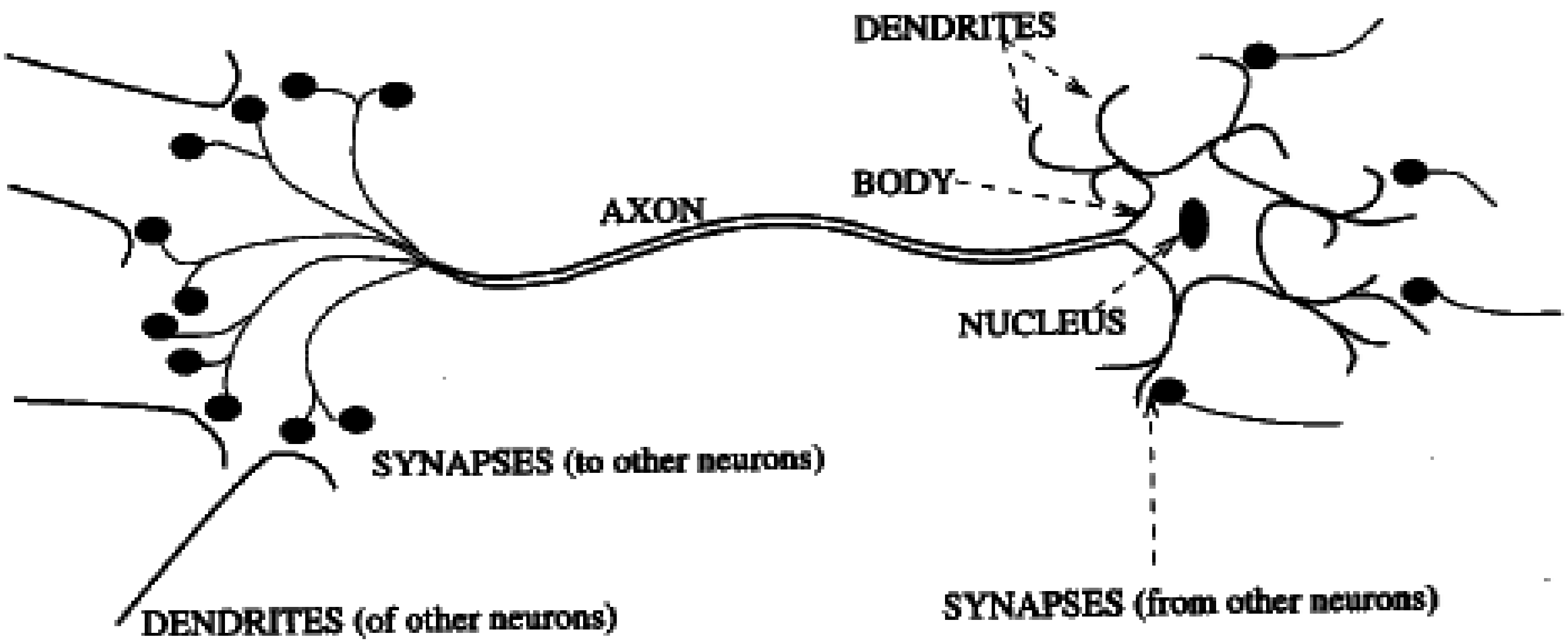
- Four Major Architectures:
  - Recursive Neural Networks
    - Recursive Autoencoder
    - Recursive Neural Tensor Network
  - Use Cases:
    - Image scene decomposition
    - NLP
    - Audio-to-text transcription

Artificial Neural Networks

- What?
  - Computing Systems inspired by Biological Neural Networks.

Biological Neural Networks

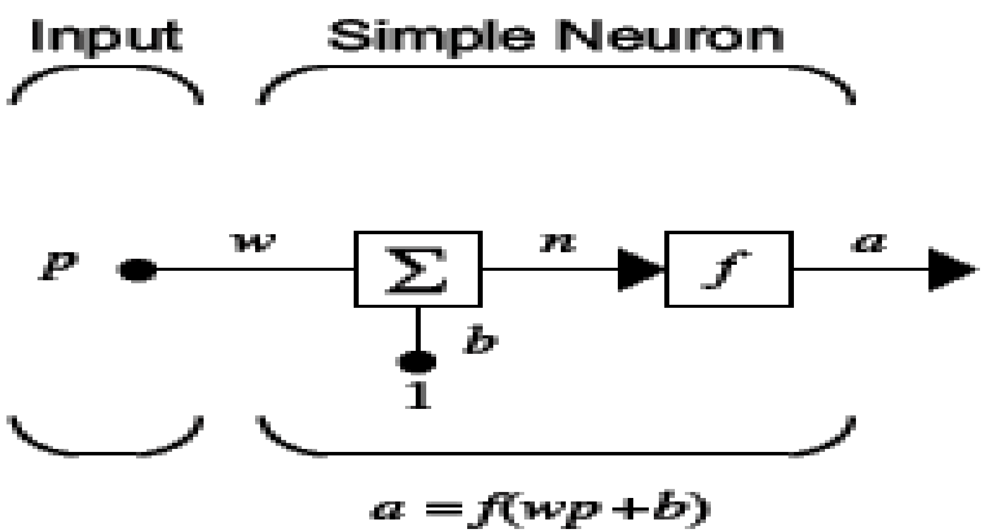
- Nervous System
  - Biological Neural Networks [5]
    - Biological Neurons
      - What?



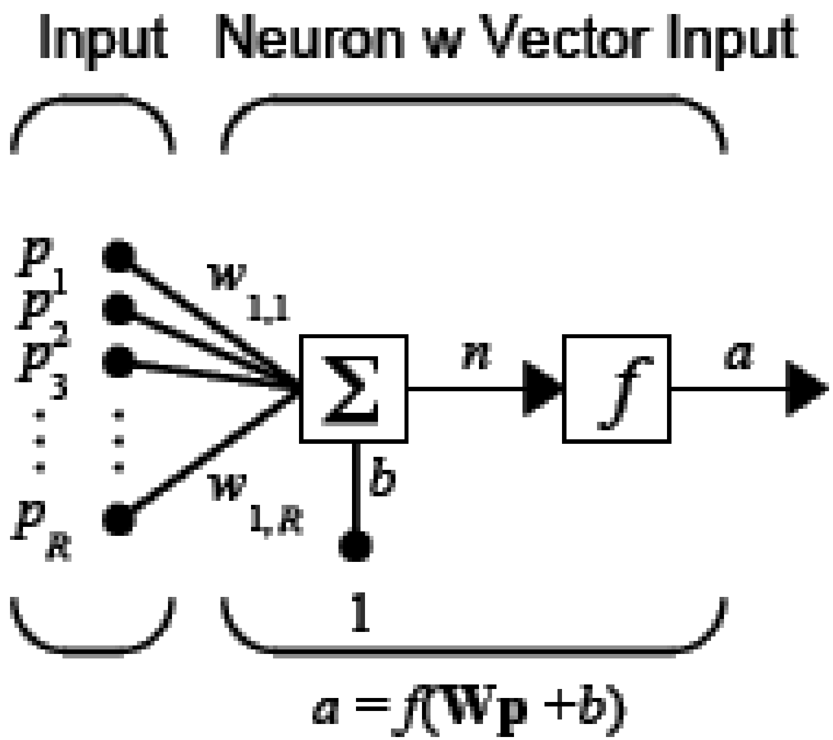
    - Features

Artificial Neuron Model

- Simple Neuron [6]
  - Weight Function, Net Input Function & Transfer Function



Neuron with Vector Input [6]

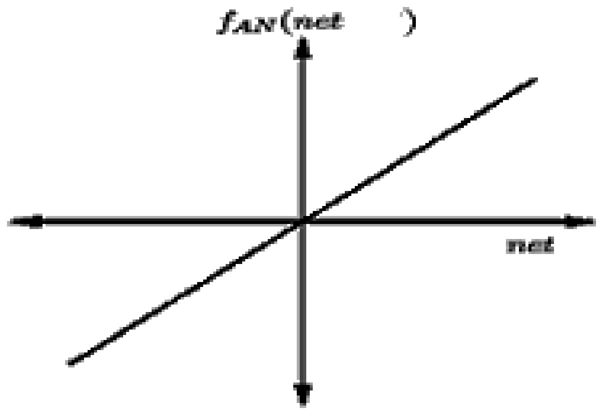


Where

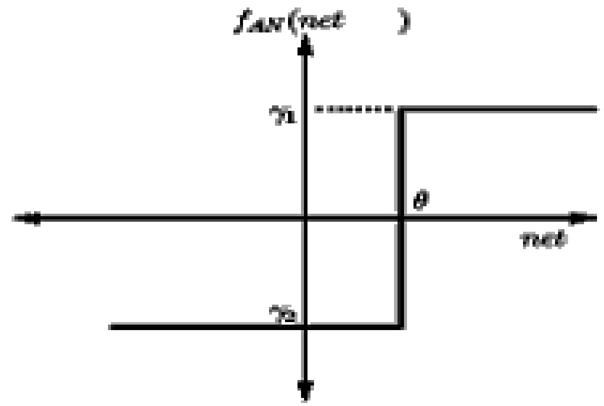
R = number of elements in input vector

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b$$

Activation Functions (Source: Not Known)

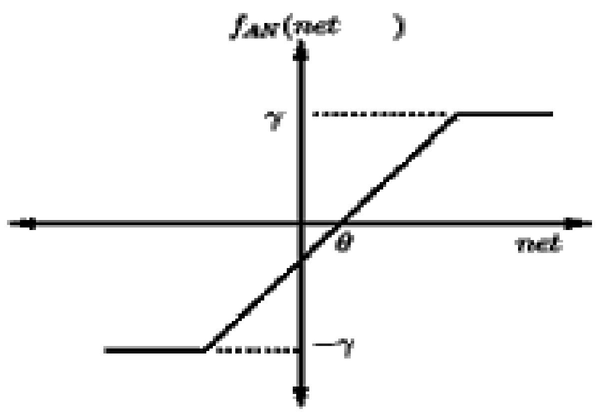


Linear function

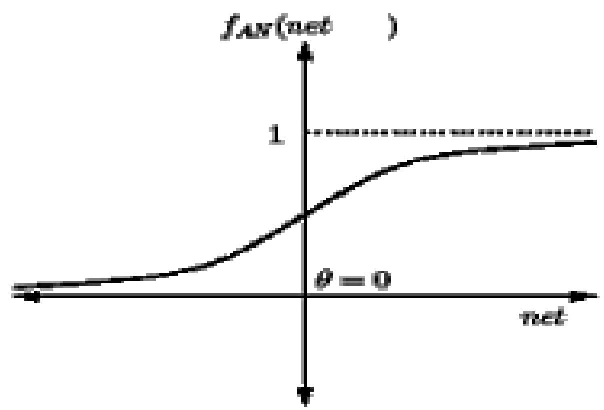


Step function

< & >=

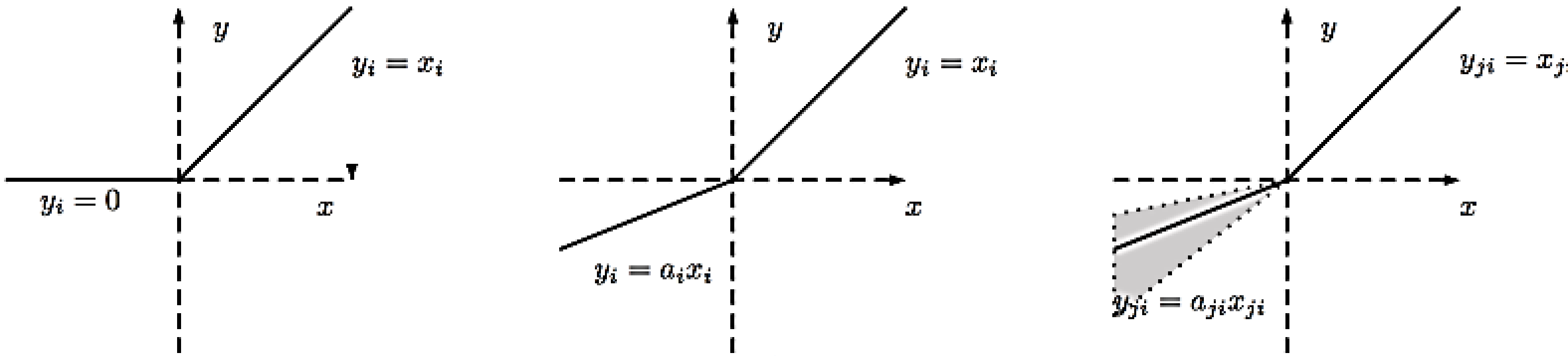


Ramp function



Sigmoid function

Activation Functions [12]



ReLU — Leaky ReLU/PReLU — Randomized Leaky ReLU

f(net) = max(0, net)

$$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ 0 & \text{if } x_i < 0. \end{cases}$$

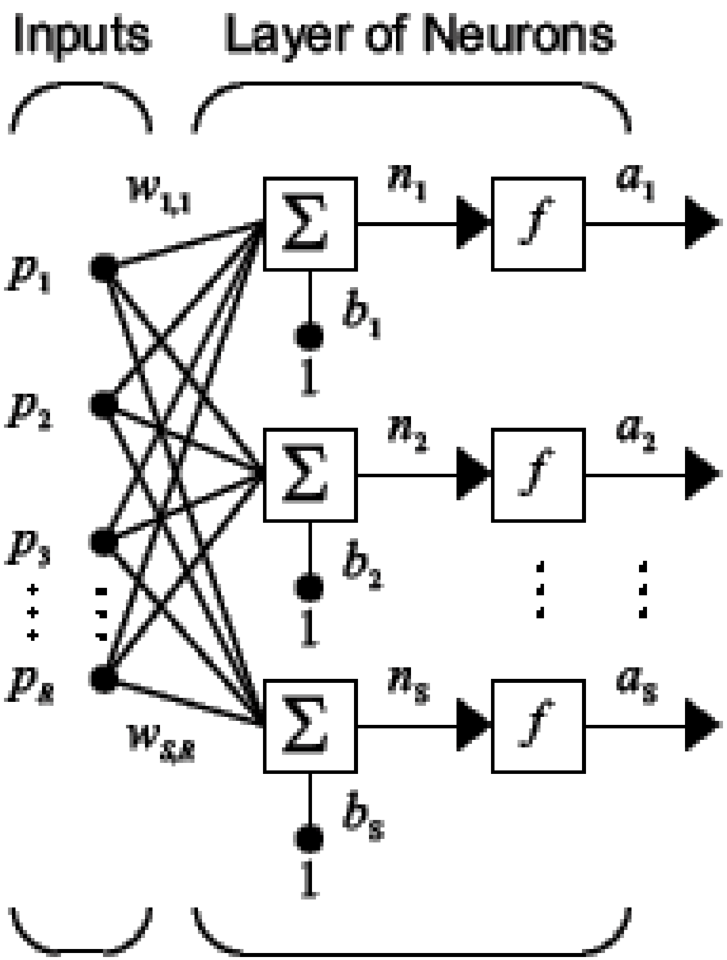
$0.01 * x_i / a_i * x_i$

$$y_{ji} = \begin{cases} x_{ji} & \text{if } x_{ji} \geq 0 \\ a_{ji} x_{ji} & \text{if } x_{ji} < 0, \end{cases}$$

$$a_{ji} \sim U(l, u), l < u \text{ and } l, u \in [0, 1)$$

a_{ji} is a random number sampled from a uniform distribution U(l, u).

A Layer of Neurons [6]



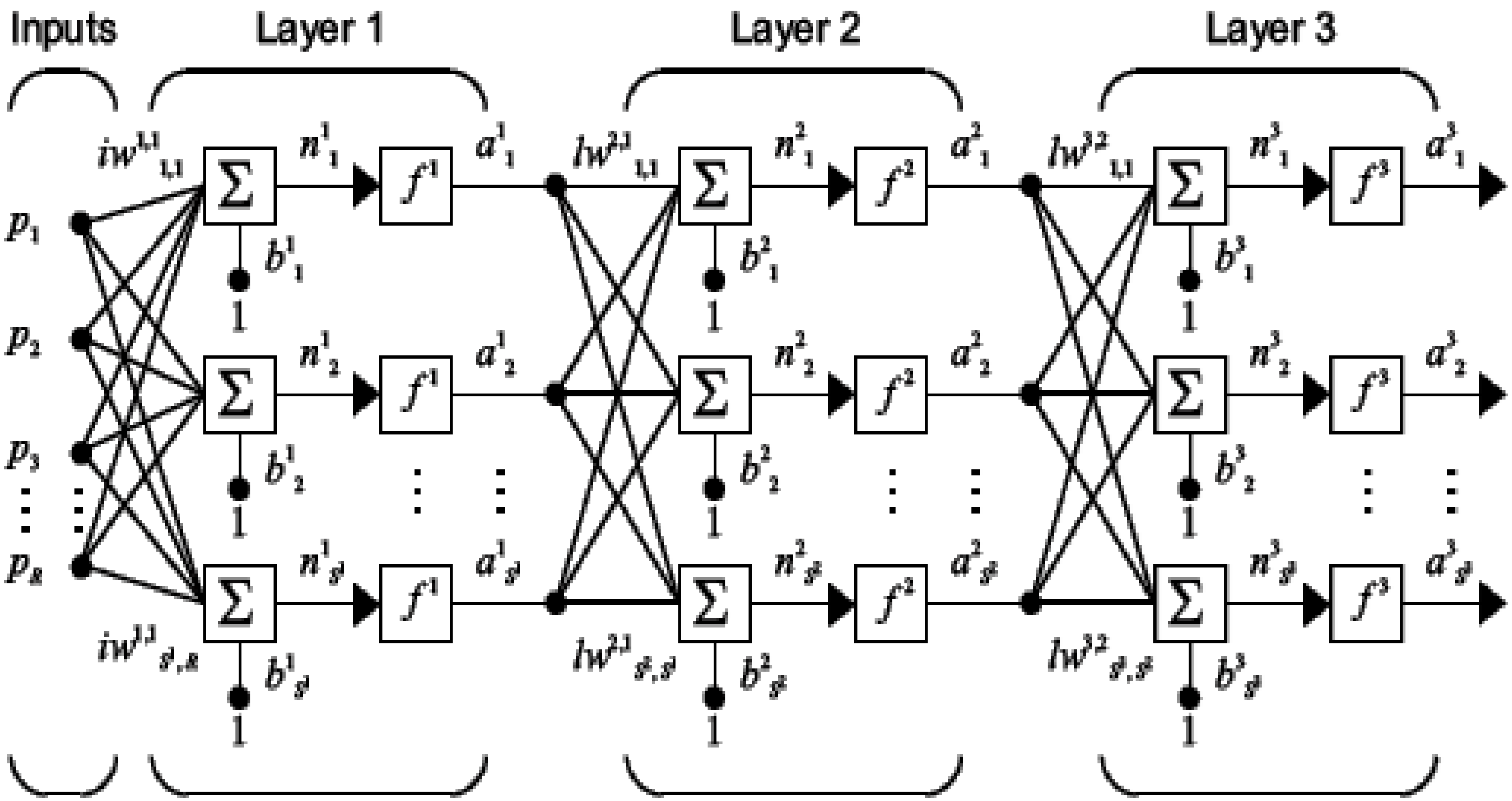
$$a = f(Wp + b)$$

Where

R = number of elements in input vector

S = number of neurons in layer

Multiple Layers of Neurons [6]

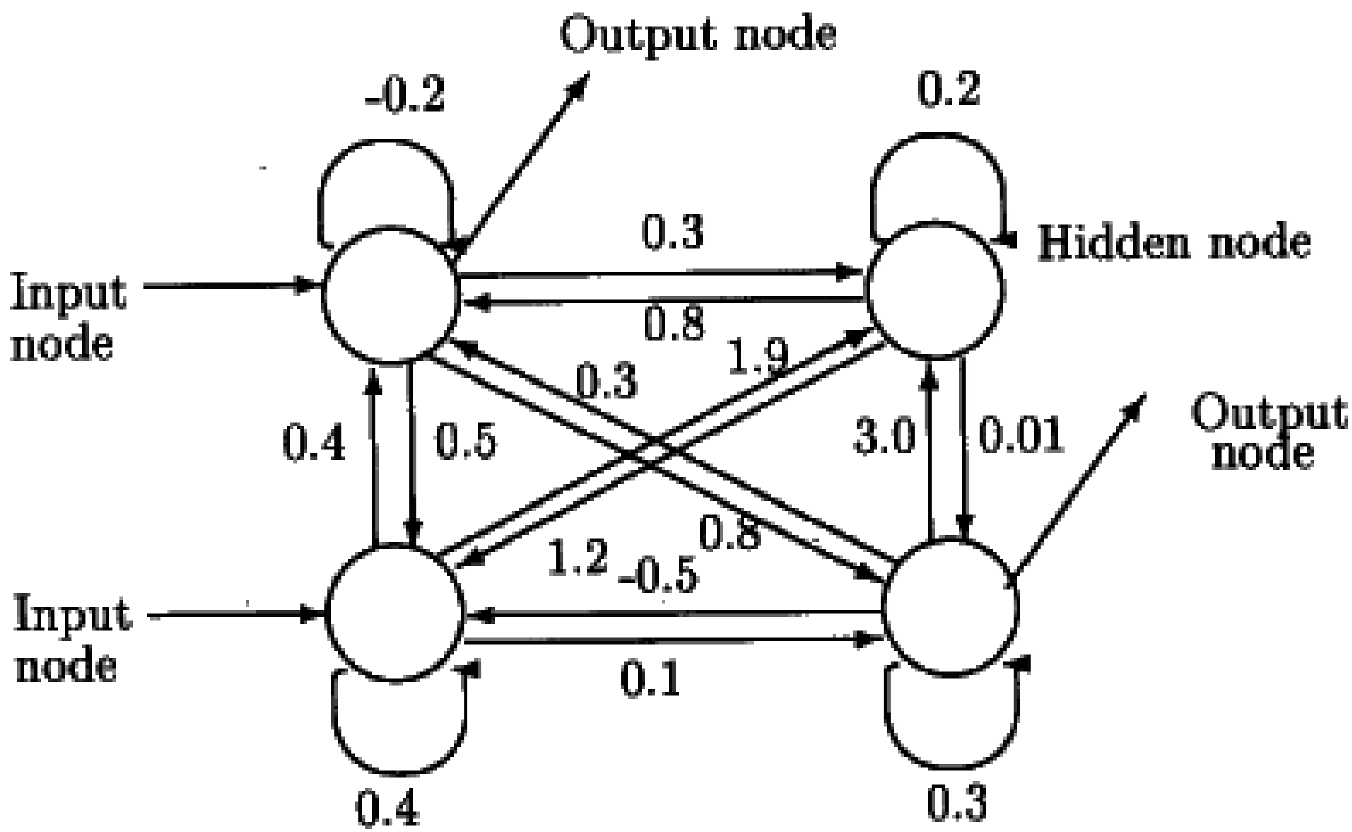


$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p}+\mathbf{b}^1)$ $\quad$ $\mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1+\mathbf{b}^2)$ $\quad$ $\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2+\mathbf{b}^3)$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

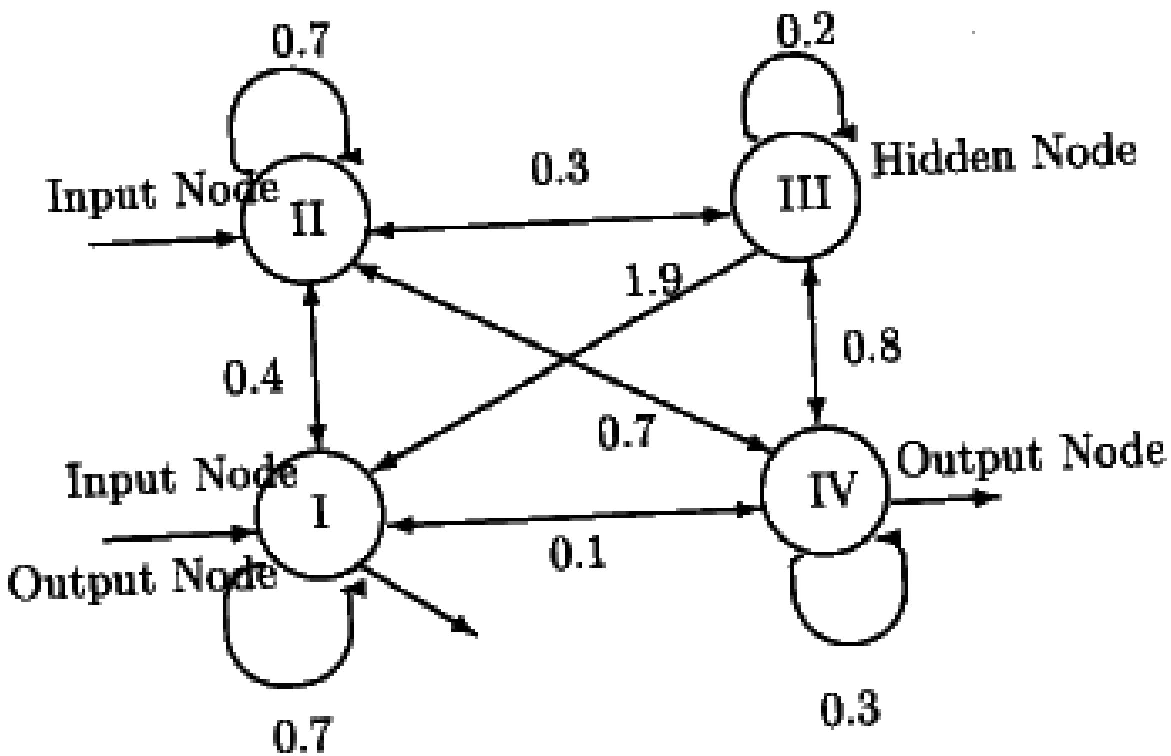
ANN Architectures [5]

- Fully Connected Network (Asymmetric)



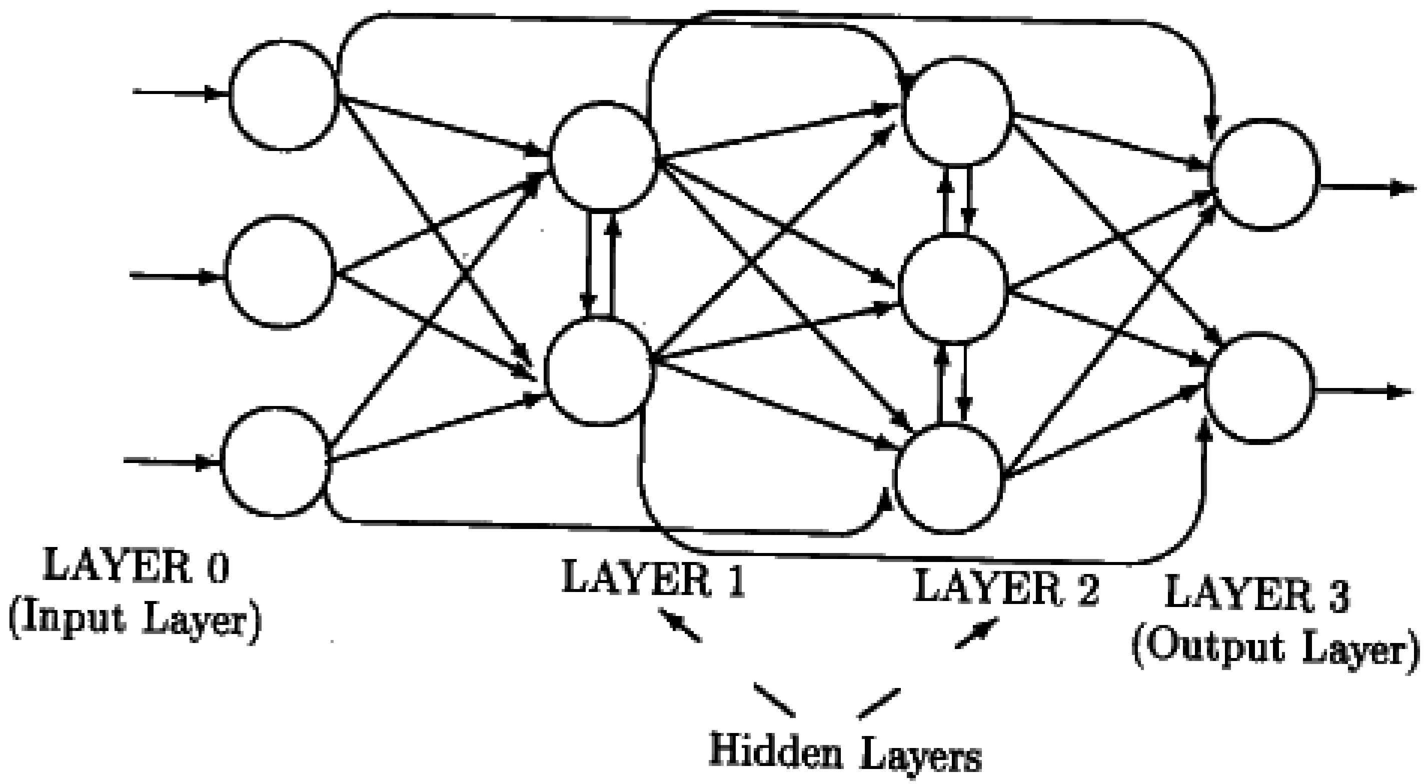
ANN Architectures

- Fully Connected Network (Symmetric)



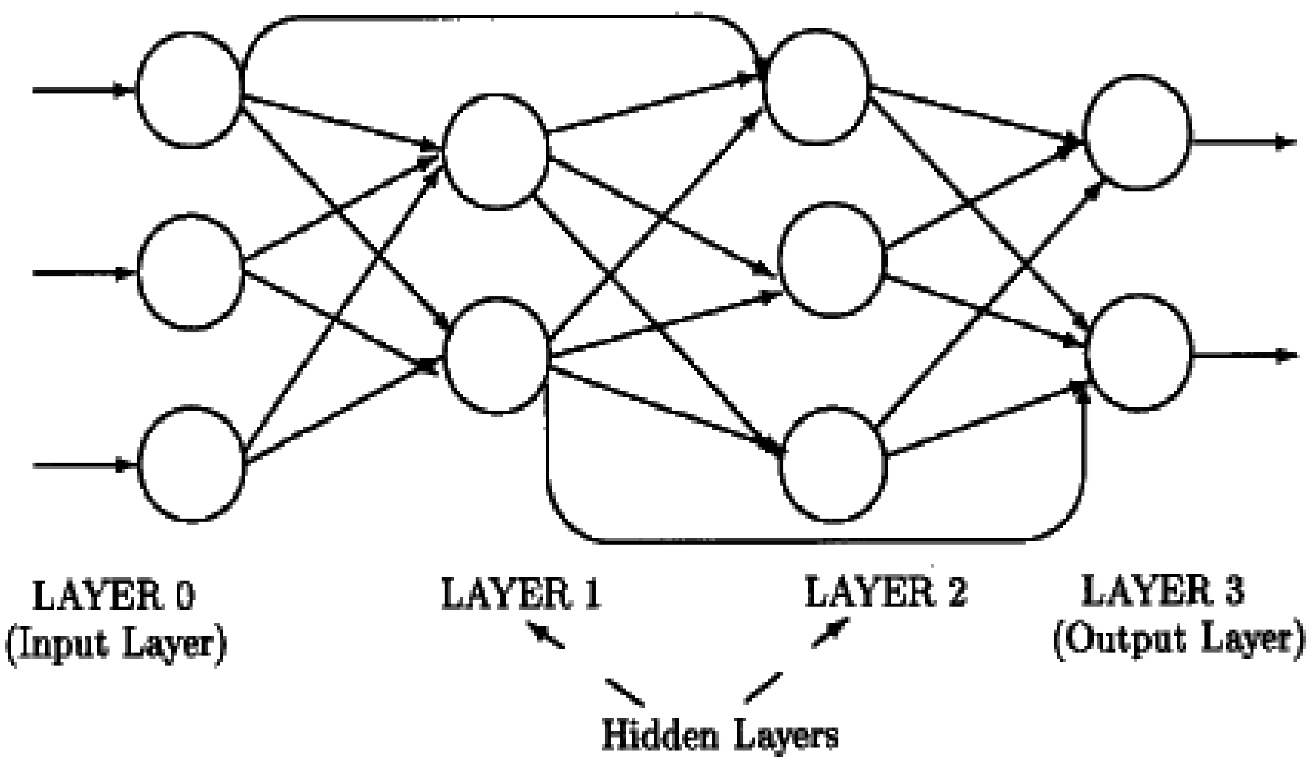
ANN Architectures

- Layered Network



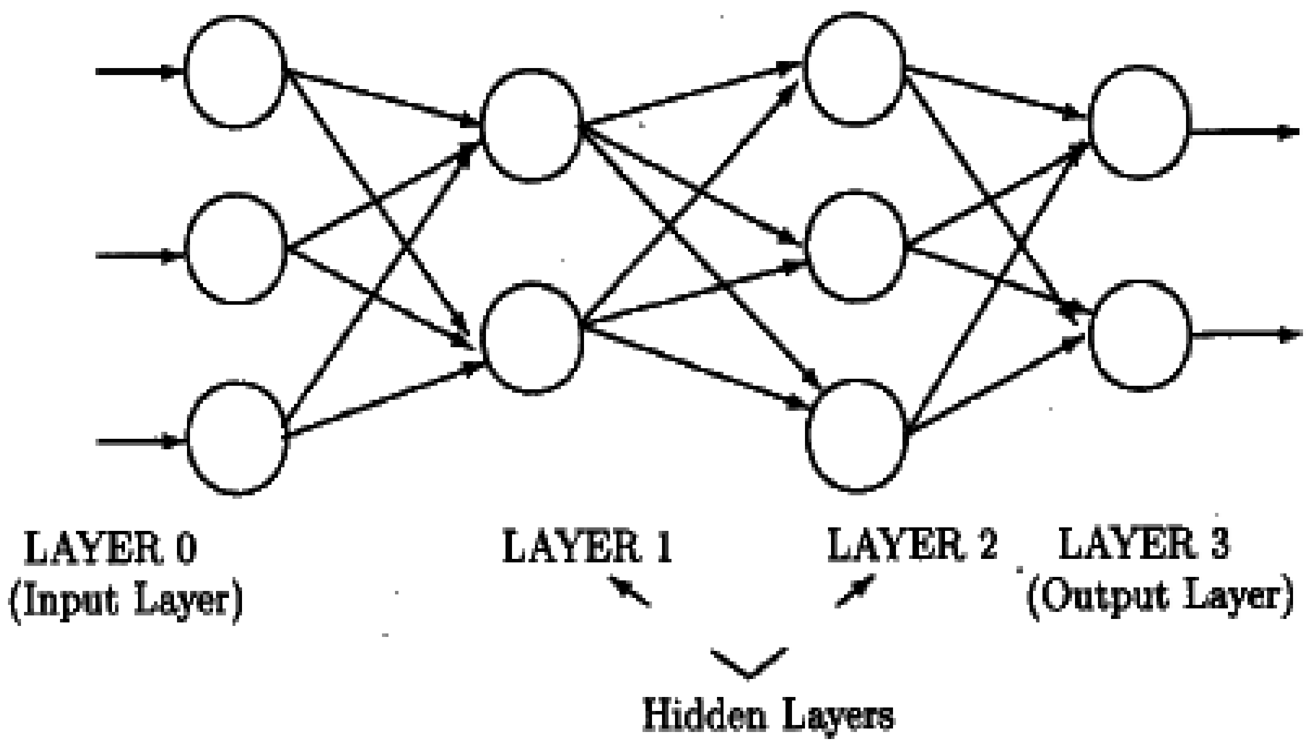
ANN Architectures

- Acyclic Network



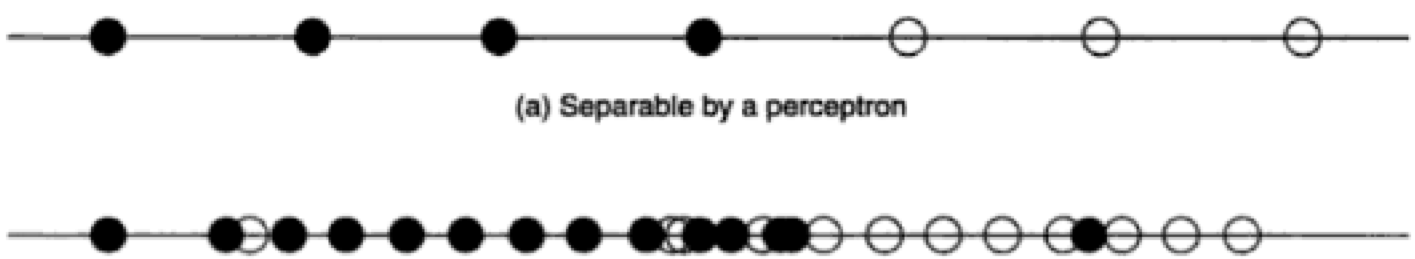
ANN Architectures

- Feedforward Network



Linear Separability

- 1 – D Case
  - 7/5 Students data – Weight Values & Obese/Not Obese
  - Learning a separating point/line [5]

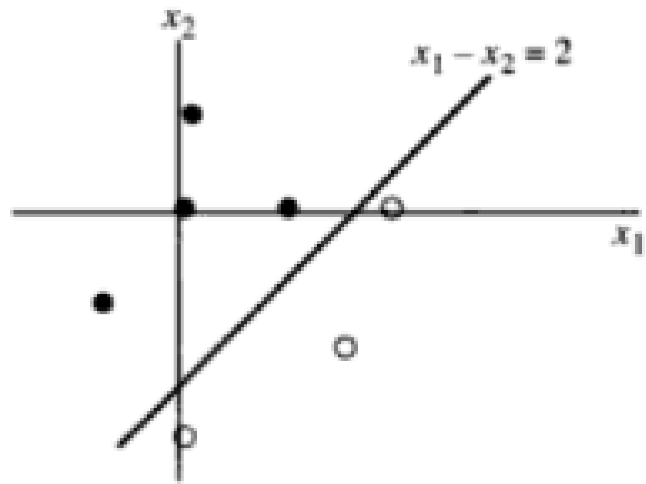


(a) Separable by a perceptron

(b) Not separable by a perceptron

Linear Separability

- 2 – D Case
  - Learning a separating line



(a)

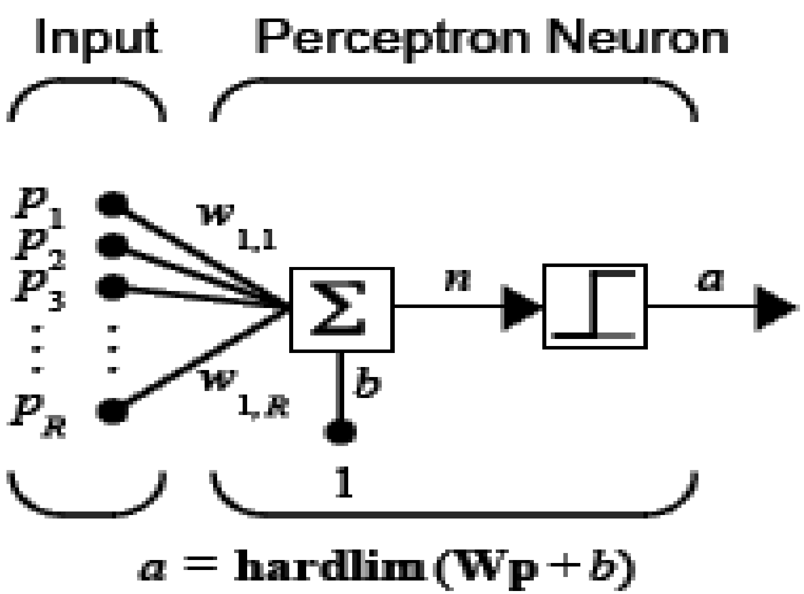
Note: Image source not known

Linear Separability

- 3 – D Case
  - Learning a separating plane
- Higher Dimensional Case
  - Learning a separating hyperplane

Perceptron Model [6]

- What is Perceptron?
- What can it do?
  - 2-class linear classification problem
    - What?
    - Process

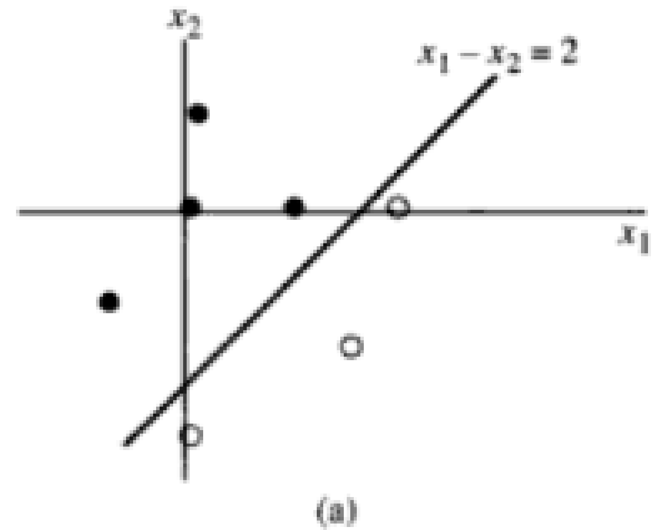


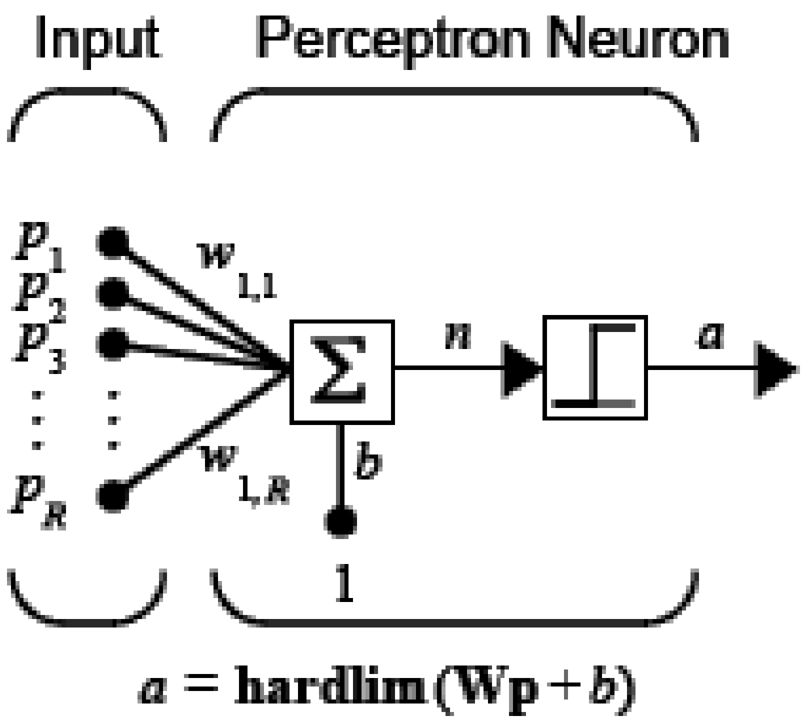
Where

R = number of elements in input vector

Perceptron Learning Rule [5, 6]

- Learning Process





R = number of elements in input vector

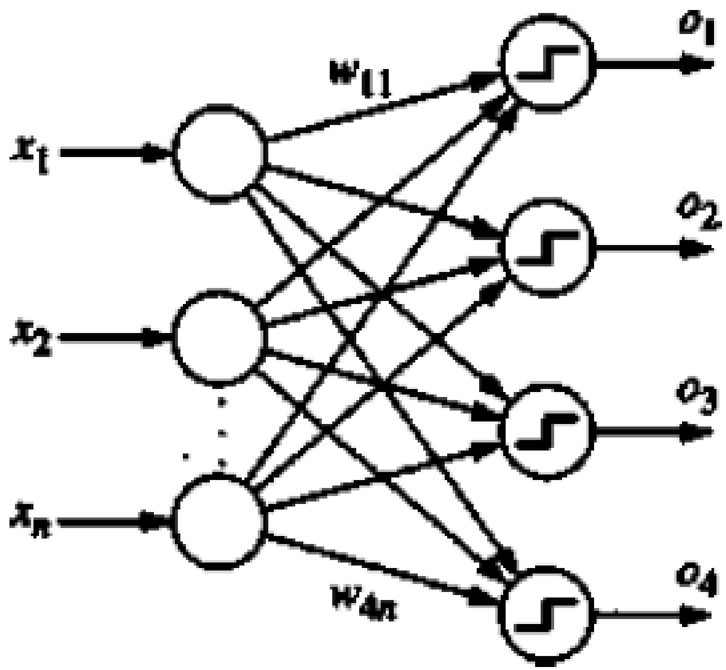
- $W_{new} = W_{old} + \eta e P$
- $b_{new} = b_{old} + \eta e$

Some Issues

- Why to use bias?
- Termination Criterion
- Learning Rate
- Non-numeric Inputs
- Epoch

Multiclass Discrimination

- Layer of Perceptron
  - To distinguish among n classes, a layer of n perceptrons can be used
  -



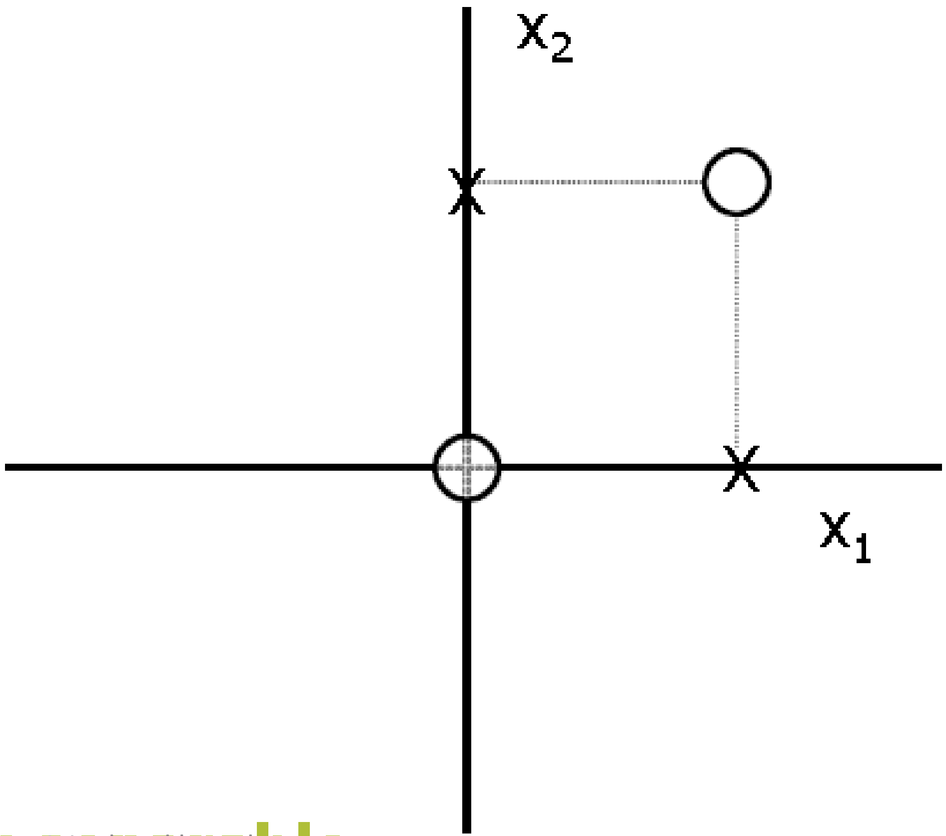
Note: Image source not known

Linearly Inseparable - Ex Or

world is not that simple…

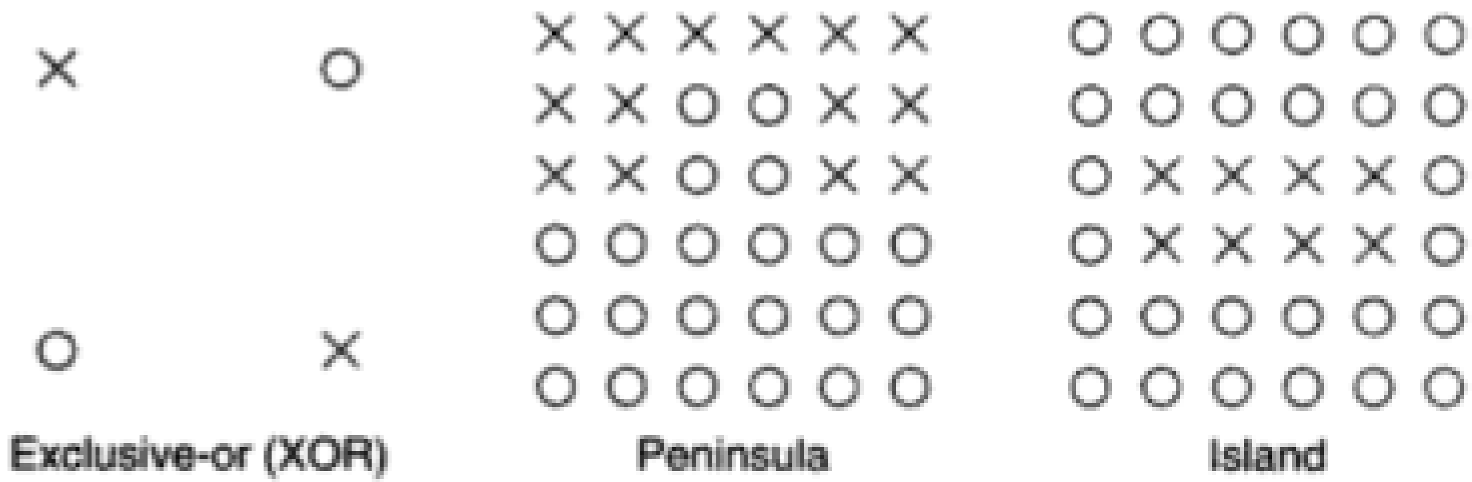
- **Ex-OR gate**

P	X_1	X_2	D
1	0	0	-1
2	0	1	+1
3	1	0	+1
4	1	1	-1



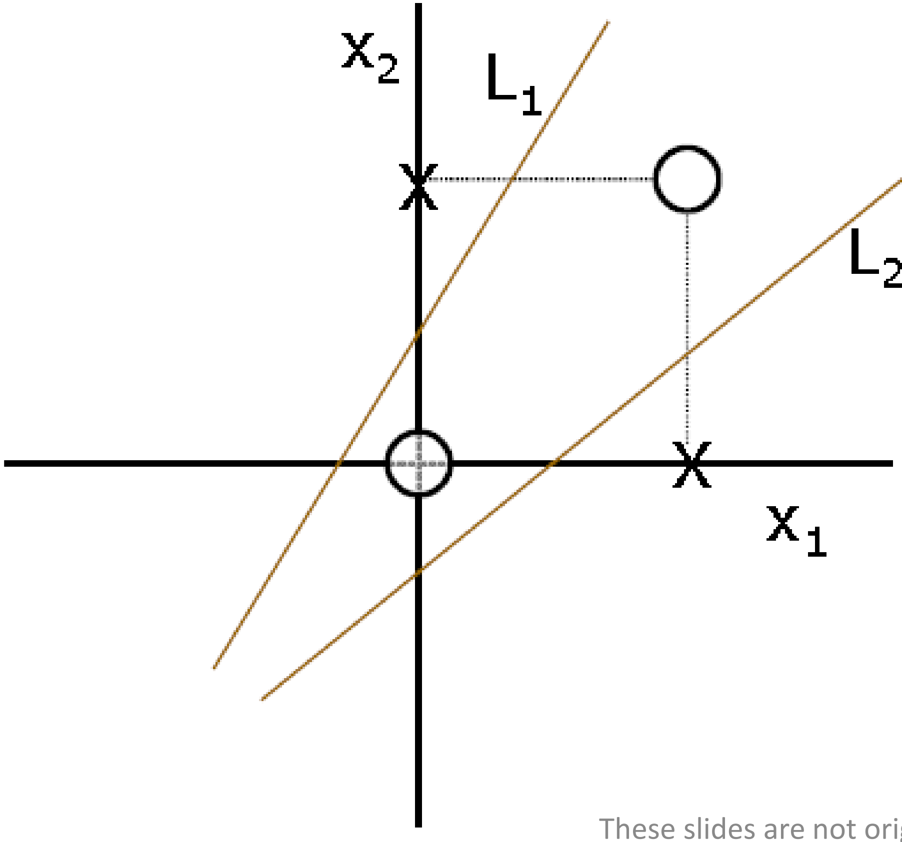
Patterns are not linearly separable

Linearly Inseparable – Ex Or



Ex-Or

Hidden transformation



L1: $-2x1+x2-1/2=0$
L2: $x1-x2-1/2=0$

$o_1=\text{sgn}(-2x1+x2-1/2)$
$o_2=\text{sgn}(x1-x2-1/2)$

$$\text{sgn}(x) = \begin{cases} -1 & \text{for } x < 0 \\ 0 & \text{for } x = 0 \\ 1 & \text{for } x > 0. \end{cases}$$

Ex-Or

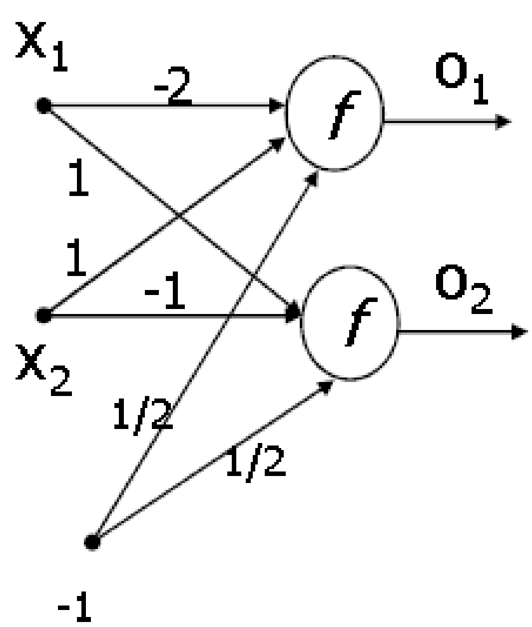
Image space

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{for } x < 0 \\ 0 & \text{for } x = 0 \\ 1 & \text{for } x > 0. \end{cases}$$

$o_1 = \operatorname{sgn}(-2x1 + x2 - 1/2)$

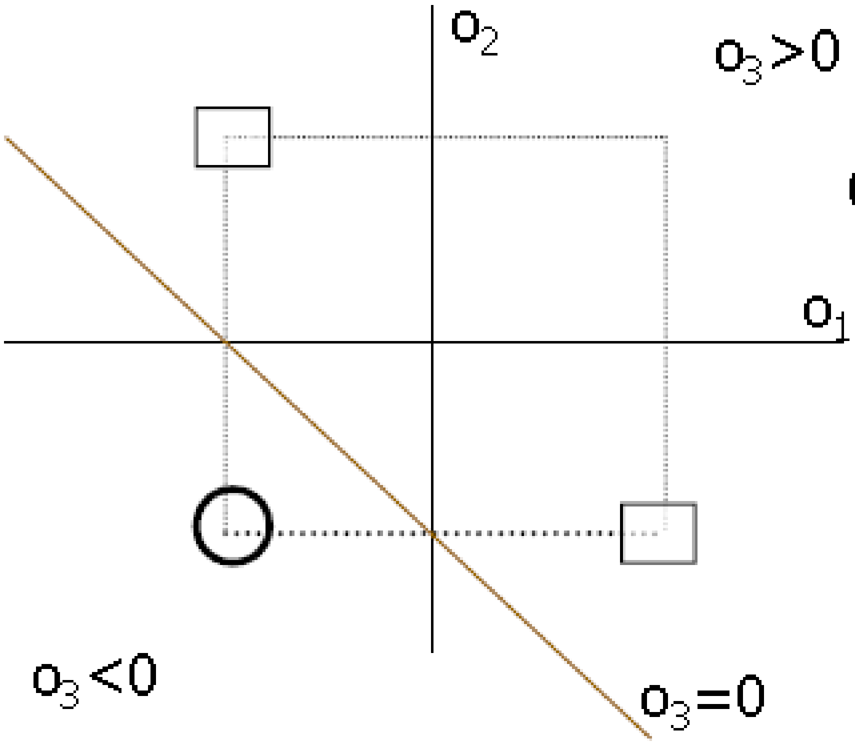
$o_2 = \operatorname{sgn}(\ x1 - x2 - 1/2)$

Pattern Space		Image Space		Class
x_1	x_2	o_1	o_2	-
0	0	-1	-1	2
0	1	1	-1	1
1	0	-1	1	1
1	1	-1	-1	2

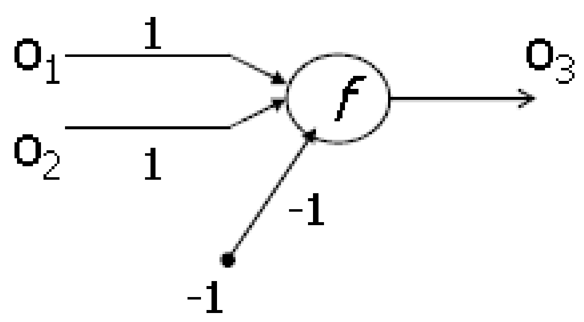


Ex-Or

Image Space



$$O_3=\mathrm{sgn}(o_1+o_2+1)$$



Ex-Or

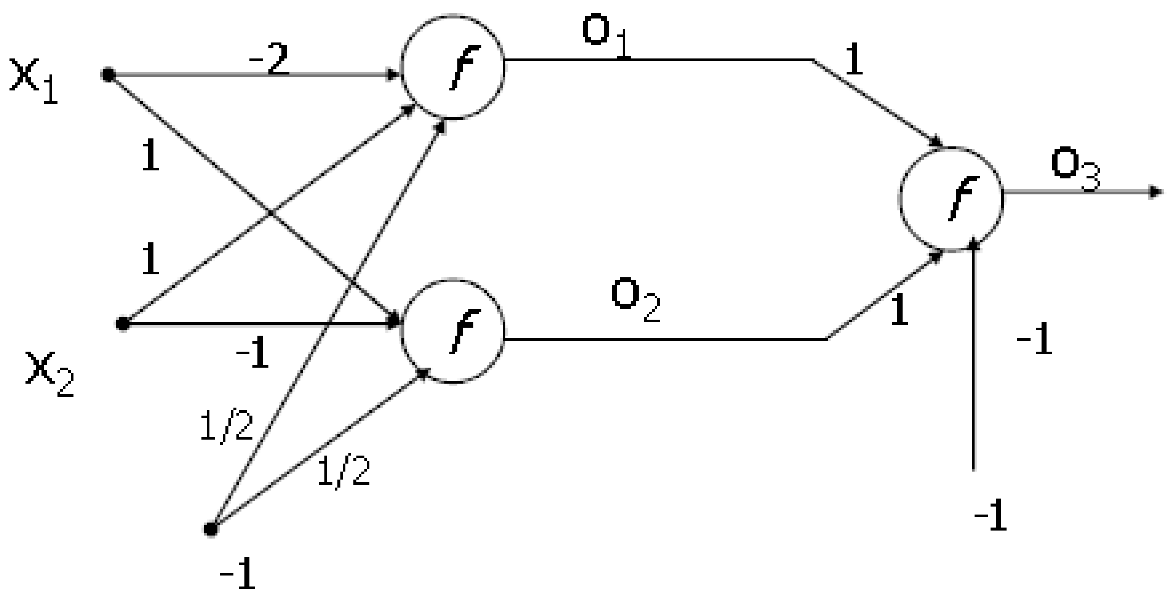
$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{for } x < 0 \\ 0 & \text{for } x = 0 \\ 1 & \text{for } x > 0. \end{cases}$$

Finally…

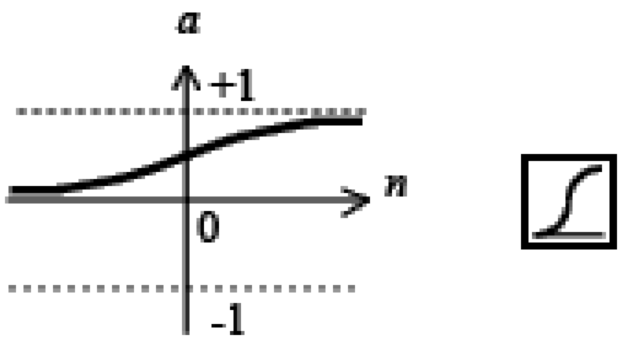
Pattern Space		Image Space		o_1+o_2+1	0_3	Class
x1	x2	o_1	o_2	-	-	-
0	0	-1	-1	-ve	-1	2
0	1	1	-1	+ve	+1	1
1	0	-1	1	+ve	+1	1
1	1	-1	-1	-ve	-1	2

Ex-Or

Two Layer Network

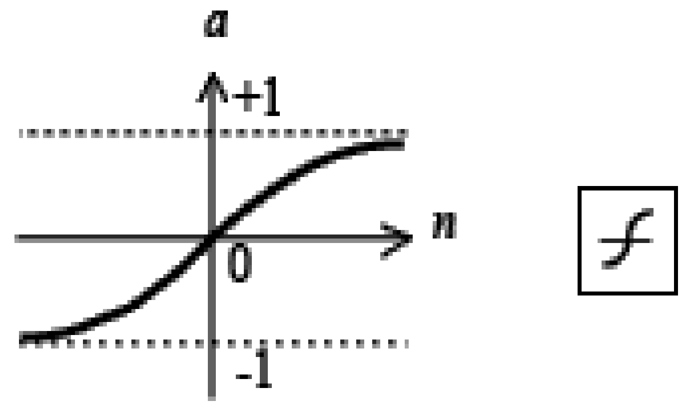


Multilayer Networks – Typical Transfer Functions [6]

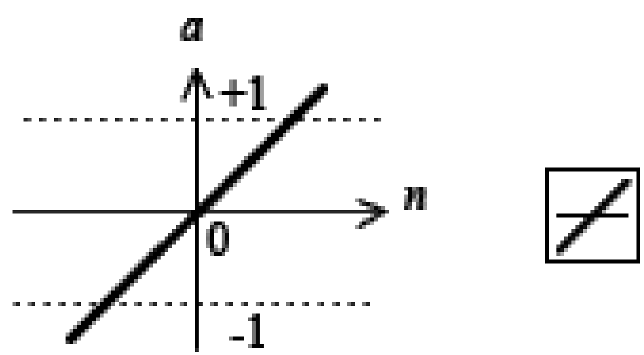


$a = logsig(n)$

Log-Sigmoid Transfer Function



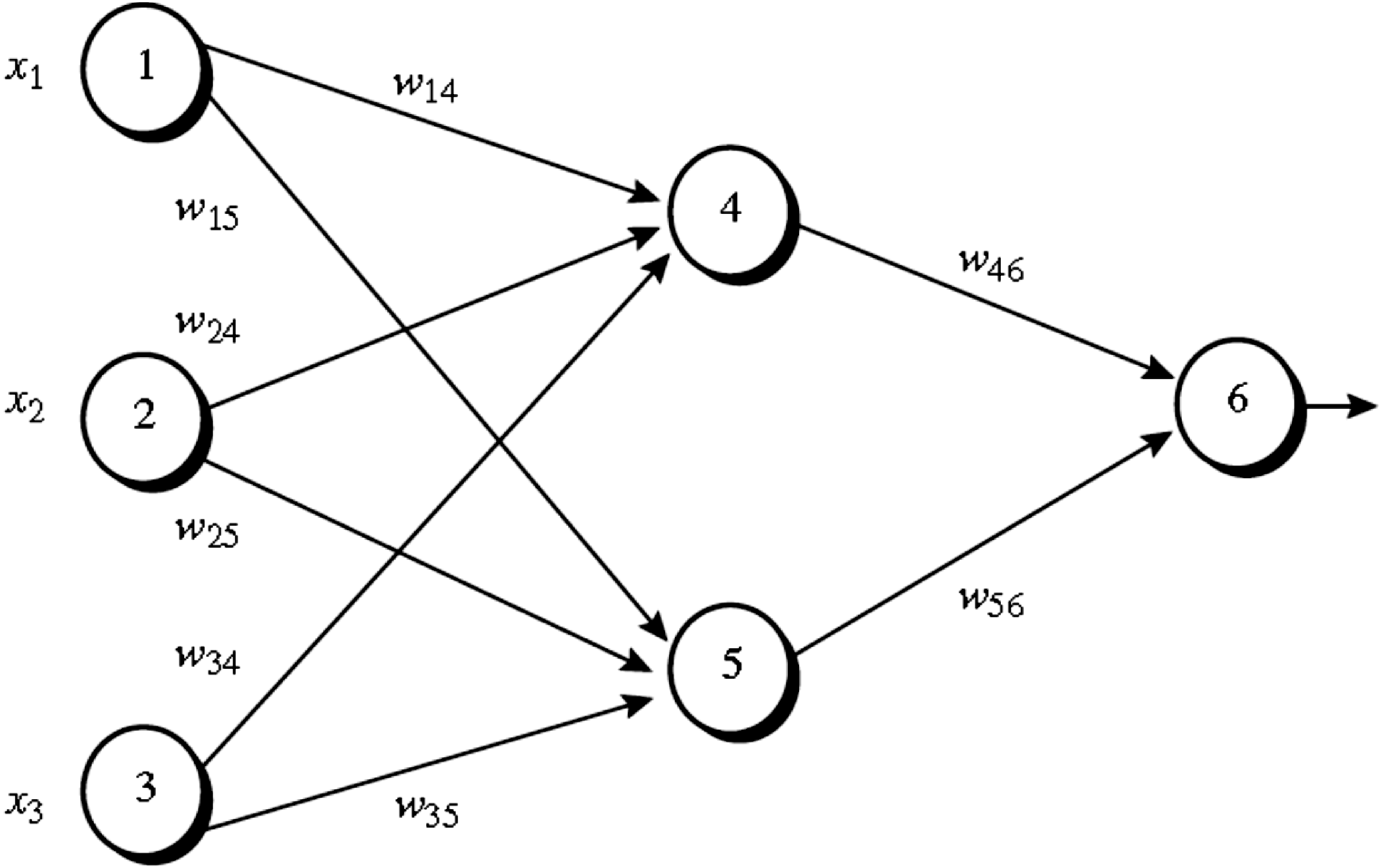
$a = tansig(n)$



$a = purelin(n)$

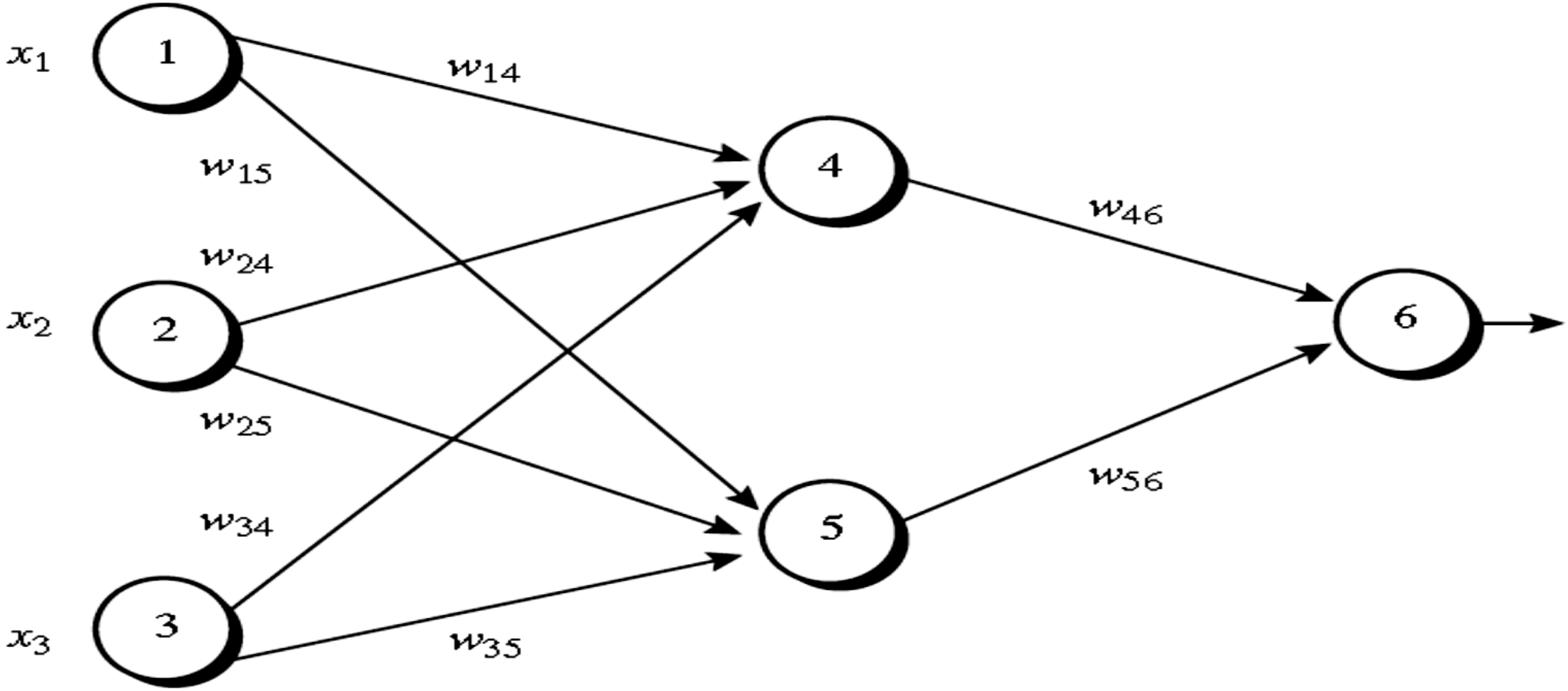
Linear Transfer Function

Example of Backpropagation [4]



An example of a multilayer feed-forward neural network.

Example of Backpropagation



An example of a multilayer feed-forward neural network.

Initial input, weight, and bias values.

Class Label : 1

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

Example of Backpropagation

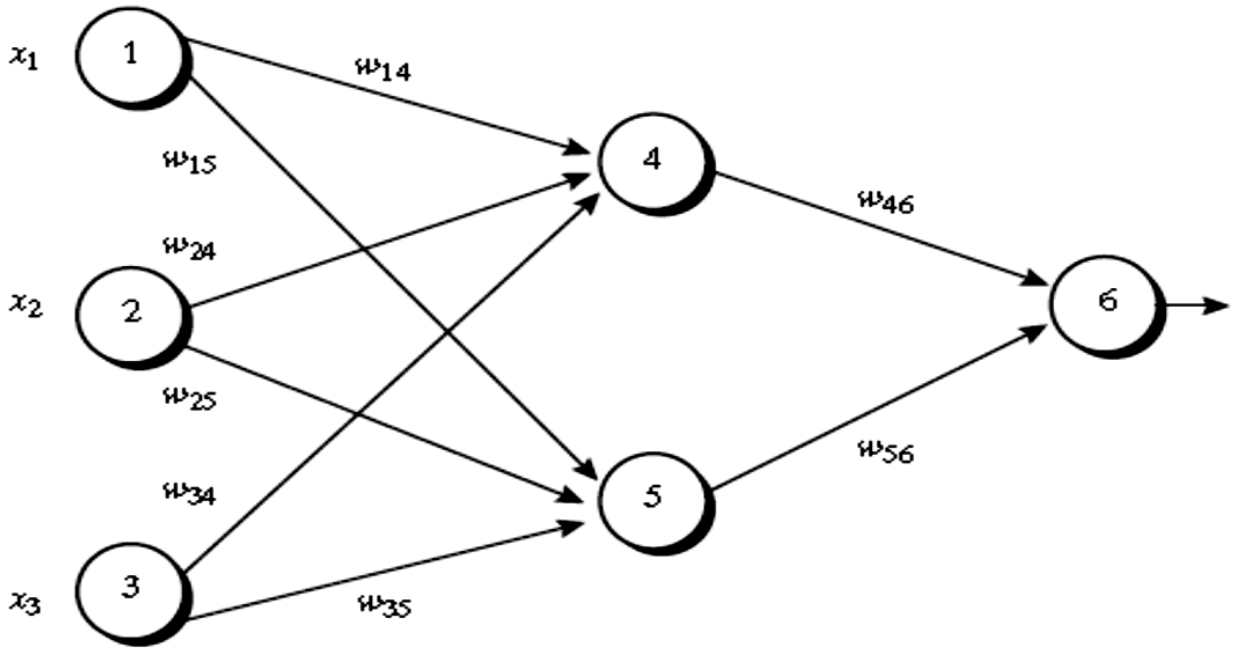


Figure 6.18 An example of a multilayer feed-forward neural network.

Table 6.3 Initial input, weight, and bias values.

Class Label : 1

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

Table 6.4 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	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

Source: Andrew Ng's Lecture on Machine Learning

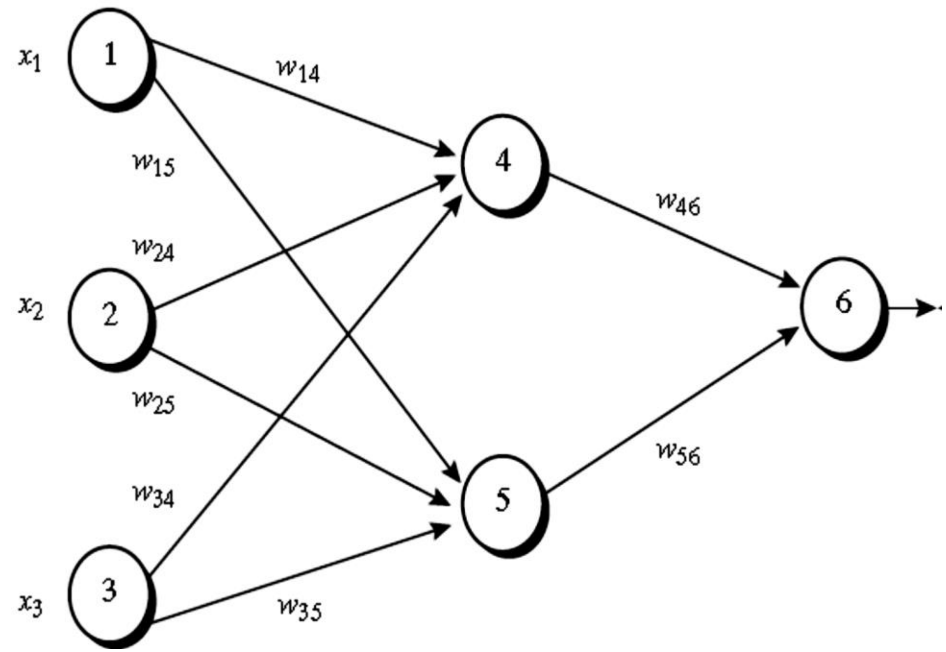
Example of Backpropagation

- Backpropagation

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

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

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



j	O_j
4	0.332
5	0.525
6	0.474

Table 6.5 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	$(0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087$

Example of Backpropagation

- Backpropagation

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

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

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

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

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

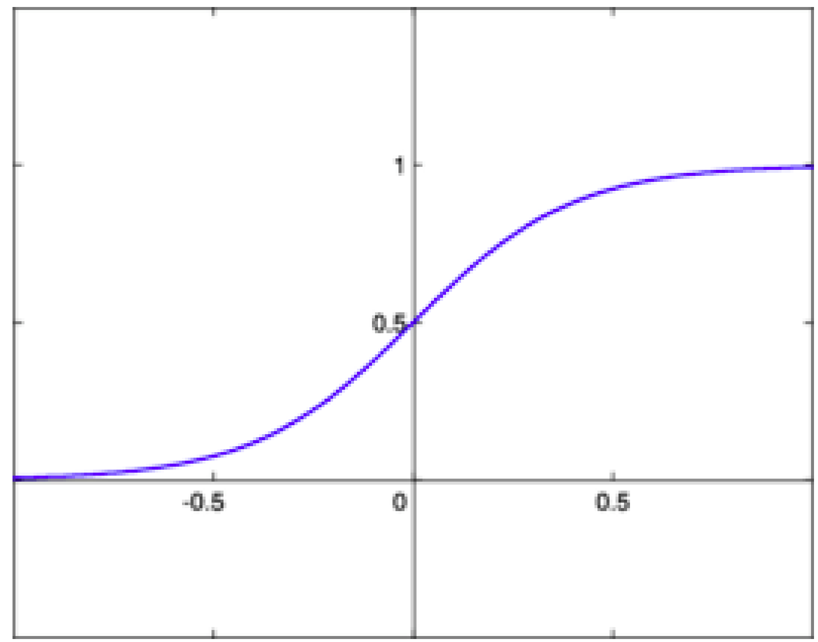
j	O_j	Err_j
4	0.332	-0.0087
5	0.525	-0.0065
6	0.474	0.1311

Table 6.6 Calculations for weight and bias updating.

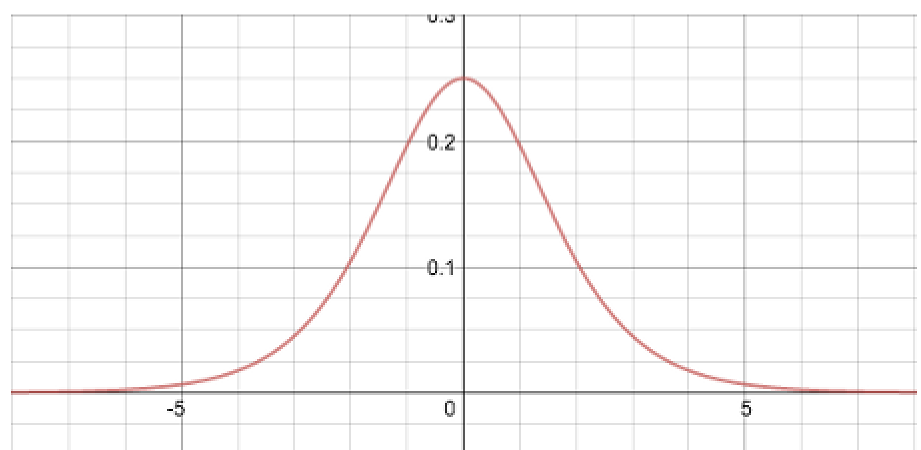
Weight or bias	*New value*
w_{46}	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
w_{56}	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
w_{14}	$0.2 + (0.9)(-0.0087)(1) = 0.192$
w_{15}	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
w_{24}	$0.4 + (0.9)(-0.0087)(0) = 0.4$
w_{25}	$0.1 + (0.9)(-0.0065)(0) = 0.1$
w_{34}	$-0.5 + (0.9)(-0.0087)(1)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

Vanishing Gradient Problem

$$Sigmoid = S(\alpha) = \frac{1}{1+e^{-\alpha}}$$



$$\frac{1}{1+e^{-\alpha}}\left[1-\frac{1}{1+e^{-\alpha}}\right]$$



Simply: S(1-S)

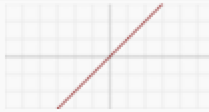
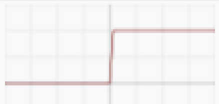
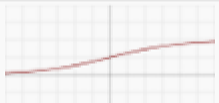
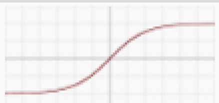
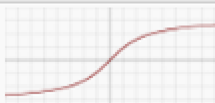
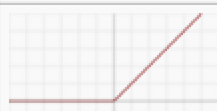
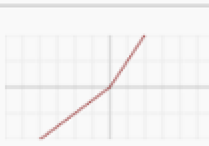
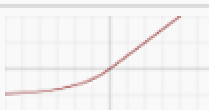
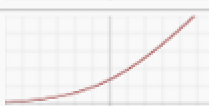
These slides are not original and have been prepared from various sources for teaching purpose - Priyank Thakkar

Note: Images are not original

Vanishing Gradient Problem

- How does ReLU solve (delay) the problem?
- Dead Neuron in case of RELU and its implication
- Leaky/Parameterized ReLU

Vanishing Gradient Problem

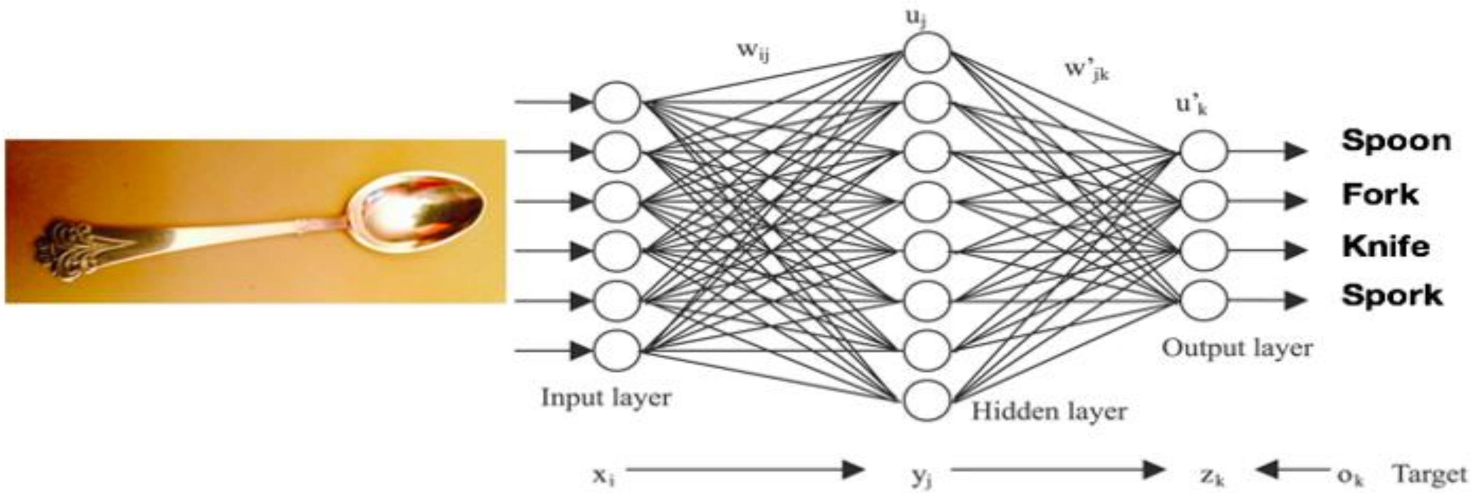
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Computer Vision & Vanilla Neural Networks

- Feature Engineering
- Loss of Structural Information
- Difference in Indented Part, Orientation, Backdrop, Size, Location
- Noise
- Scalability

Computer Vision & Vanilla Neural Networks

- Loss of Structural Information

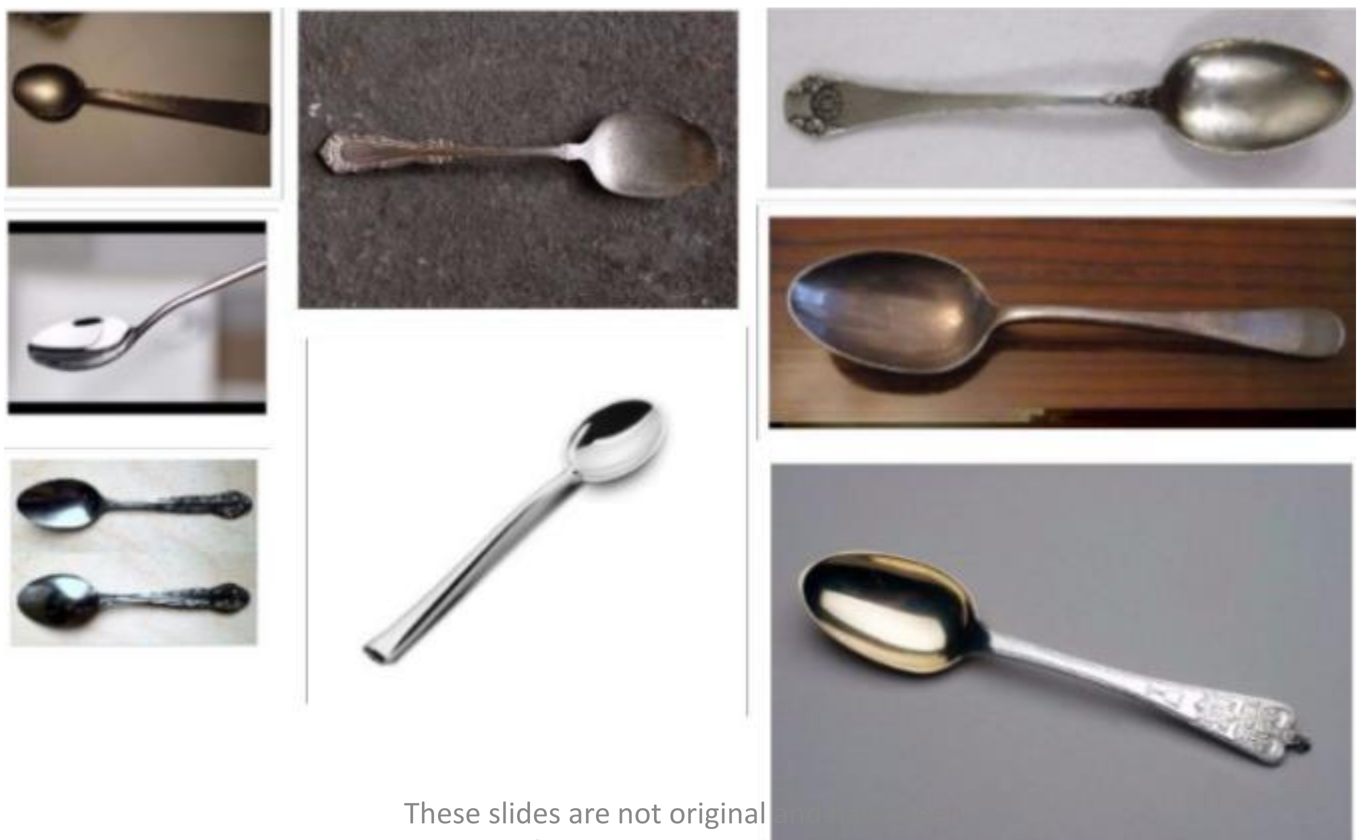


$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Pixel Vector

Computer Vision & Vanilla Neural Networks

- Difference in Indented Part, Orientation, Backdrop, Size, Location
- Noise



Computer Vision & Vanilla Neural Networks

- Scalability

Disclaimer

- These slides are not original and have been prepared from various sources for teaching purpose.

References

1. https://towardsdatascience.com/the-10-deep-learning-methods-ai-practitioners-need-to-apply-885259f402c1
2. https://semiengineering.com/deep-learning-spreads/
3. https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/ch04.html

References

4. Data mining: concepts and techniques, J. Han, and M. Kamber. Morgan Kaufmann, (2006)

5. Elements of Artificial Neural Networks, Kishan Mehrotra, Chilukuri K. Mohan, Sanjay Ranka. MIT Press, (1997)

6. Matlab Neural Network Tollbox Documentation

7. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

8. Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

References

9. https://ayearofai.com/rohan-lenny-2-convolutional-neural-networks-5f4cd480a60b

10. http://cs231n.github.io/convolutional-networks/

11. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

12. Xu, Bing, et al. "Empirical evaluation of rectified activations in convolutional network." arXiv preprint arXiv:1505.00853 (2015).

Disclaimer

- These slides are not original and have been prepared from various sources for teaching purpose.