Introduction to Deep Neural Networks (DNNs)

Min-sung Koh, Ph. D., Professor

Department of Computer Science and Electrical Engineering (CSEE)
Eastern Washington University, WA
USA

Course Information (1)

- Textbook : Charu C. Aggarwal, “Neural Networks and Deep Learning”, Springer, 2018

- Lab book: Depending on EE or CS, one of the following lab books is suggested.
  - E. Stevens, L. Antiga, and T. Viehmann, “Deep Learning with PyTorch”, Manning, 2020, https://pytorch.org/assets/deep-learning/Deep-Learning-with-PyTorch.pdf
  - Phil Kim, “MATLAB Deep Learning”,Apress, 2017, for EE (Since we will use Python, it is NOT suggested)

Course Information (2)

All my lecture notes are made using the following references. Some of the followings are available in websites.

[1] Charu C. Aggarwal, "Neural Networks and Deep Learning", Springer, 2018, http://www.charuaggarwal.net/neural.htm

[2] I. Goodfellow, Y. Bengio, and A. Courville, " Deep Learning", MIT Press, 2016, https://www.deeplearningbook.org/

[3] Lecture notes for "Neural Networks and Deep Learning" by Prof. Roger Grosse, at http://www.cs.toronto.edu/~rgrosse/courses/csc421_2019/

[4] Lecture notes for "Introduction to Artificial Intelligence", 2019, given by Prof. Ha-Jin Yu in University of Seoul.

[5] Michael Nielsen, "Neural Networks and Deep Learning", at http://neuralnetworksanddeeplearning.com

[6] Lecture notes for "Deep Learning" by Prof. Sargur Srihari, at https://cedar.buffalo.edu/~srihari/CSE676/

[7] Phil Kim, "MATLAB Deep Learning", Apress, 2017

[8] E. Stevens, L. Antiga, and T. Viehmann, "Deep Learning with PyTorch",Manning, 2020, https://pytorch.org/assets/deep-learning/Deep-Learning-with-PyTorch.pdf

Course Information (3)

[9] George. F. Luger, “Artificial Intelligence: Structures and Strategies for Complex Problem Solving”, Pearson, 2008,

[10] Rob Callan, “Artificial Intelligence”, Palgrave, 2003,

[11] Stuart Russell and peter Norvig, “Artificial Intelligence : A Modern Approach”, Prentice hall, 2010

[12] Takayuki Okatani (translated in Korean by Hyo-sup Shim), “Deep Learning”, ISBN: 9784061529021, Jpub, 2016

[13] Nikai Etchuzi (translated in Korean by Myung-Jo Jin), “Deep Learning Getting Started with TensorFlow”, ISBN: 9784839960889, Jpub, 2017

[14] Fei-Fei Li, Ranjay Krishna, and Danfei Xu, “CS231n: Convolutional Neural Networks for Visual Recognition”, Lecture notes, 2019, at http://cs231n.stanford.edu/

Course Information (3)

- Grading policy
  - Homeworks : 10 %
  - Labs : 30 %
  - Midterm exam : 30 %
  - Final exam (class project) : 30 %
- Contact Info.
  - Use the given office hours.
  - Otherwise, please make an appointment.

Labs

- All labs will be done with Python. You may do with MATLAB (w/o given lab guide), but Python is highly recommended to use to handle big data.
- You can run Python using one of the following ways:
  - Use a web browser with "Jupyter notebook" (In this case, you don't need to install Python and other APIs but you have to install the "Jupyter notebook" in your computer). For this, we will use "Colab". Before coming to lab tomorrow, please watch the short (3 min) video given as "Introduction to Colab" at Welcome To Colaboratory - Colaboratory (google.com)
  - Use Jupyter notebook in your own computer (In this case, you have to install both Python and Jupyter notebook in your computer).

Introduction (1)

- In May 2017, AlphaGo, defeated the best player of Go game.
- IBM's Watson computer diagnosed a Leukemia patient correctly in 10 minutes although several doctors were struggling to identify the name of the disease correctly.
- The healthcare market using Artificial Intelligence (AI) is growing rapidly and expected to grow up to 309.6 billion dollars by 2026 (58.3 billion in 2021).

Introduction (2)

- Moreover, the key words, “Alexa”/ “Hey Siri” / “Hey Google”, are gaining regular usage in our daily life to work with an Iphone / Ipad in order to receive information by talking to it.
- A voice command system in various cars is becoming essential in modern vehicles to help drivers receive required information while driving.
- An important, fundamental secret key hidden in aforementioned AI examples is “Deep learning”.

Introduction (3)

- Deep learning has gained its popularity because it can handle many complex tasks, which was not possible before.
- Deep learning has been applied to Deep Neural Networks (DNNs) such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs) etc. to handle complex tasks.
- Since DNNs such as CNNs and RNNs etc. have multiple hidden layers, it is a challenging problem to train them efficiently.

Artificial Intelligence (AI)

- Reasoning
- Learning
- Language communication
- Planning
- Decision making
- Common-sense

Application of AI

- Video processing
  - Autonomous Vehicles (or self-driving vehicles)
  - Video search
- Speech / Natural language processing
  - Smart Speaker
  - “Alexa” / “Hey, Google” / “Hey Siri”
- Big Data
  - Recommendation
- Security
  - Speaker/face/finger print recognitions
- Financing

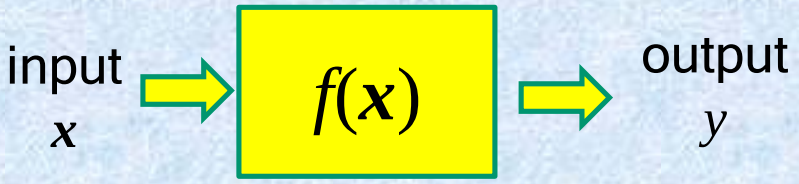
Brief history of AI

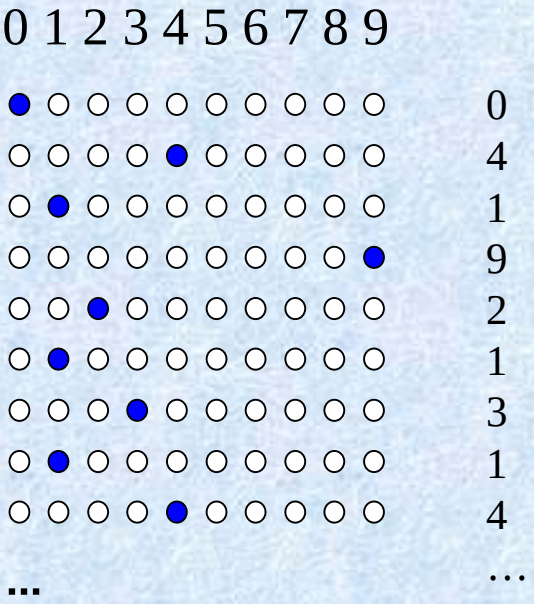
- A conference in 1956
  - John McCarthy (one of the organizers of the conference)
  - “Artificial intelligence” is coined.
- Knowledge-based systems
  - Expert system
    - Relying on hard-coded knowledge could not acquire their own knowledge by extracting pattern from raw data.
- In 1980s through1990s
  - Multi-layer perceptron
- In 2006 through up to present
  - Deep learning

An application of neural networks

(In classifying handwritten digits)

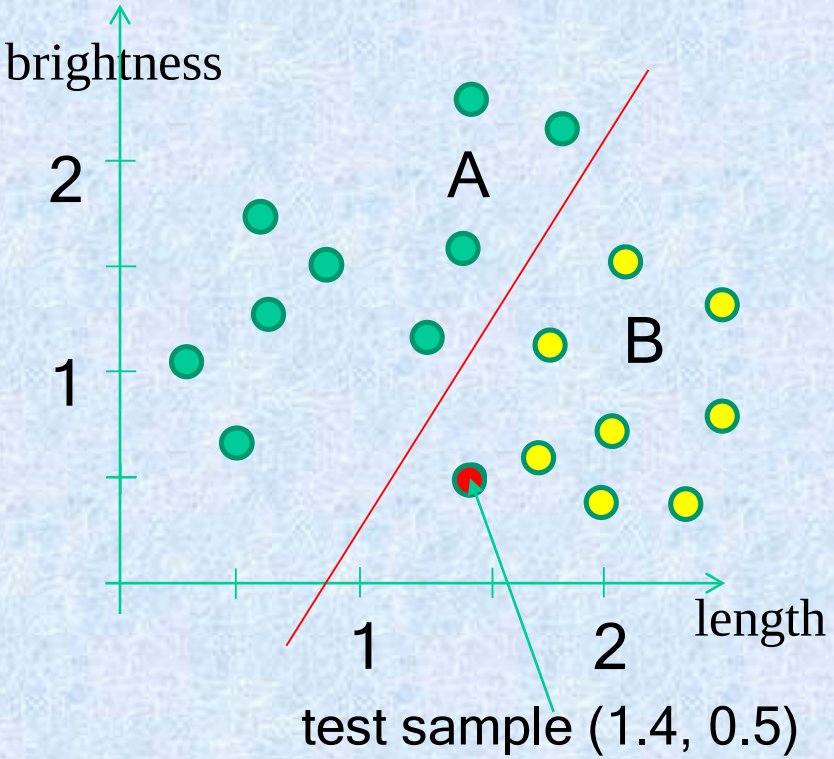
- Speech / speaker
- Image (face, characters, …)
- Language
- Music
- …





An application of neural networks

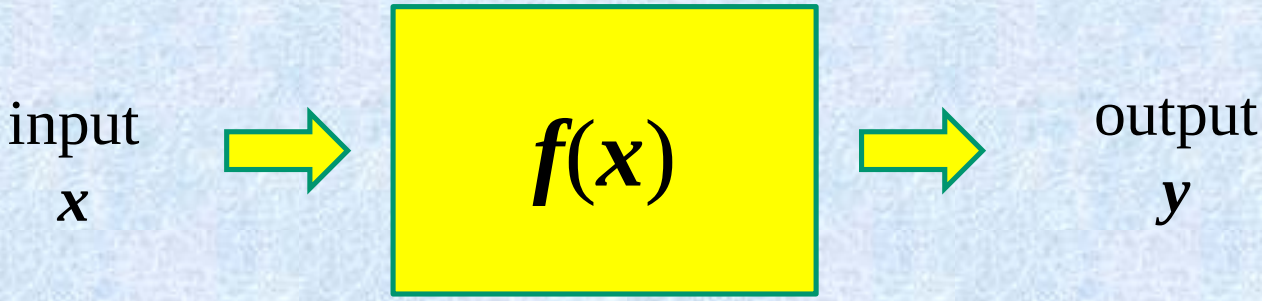
(In classifying fishes)



- Features : length, color, brightness, weight, …

An application of neural networks

(In classifying examples)



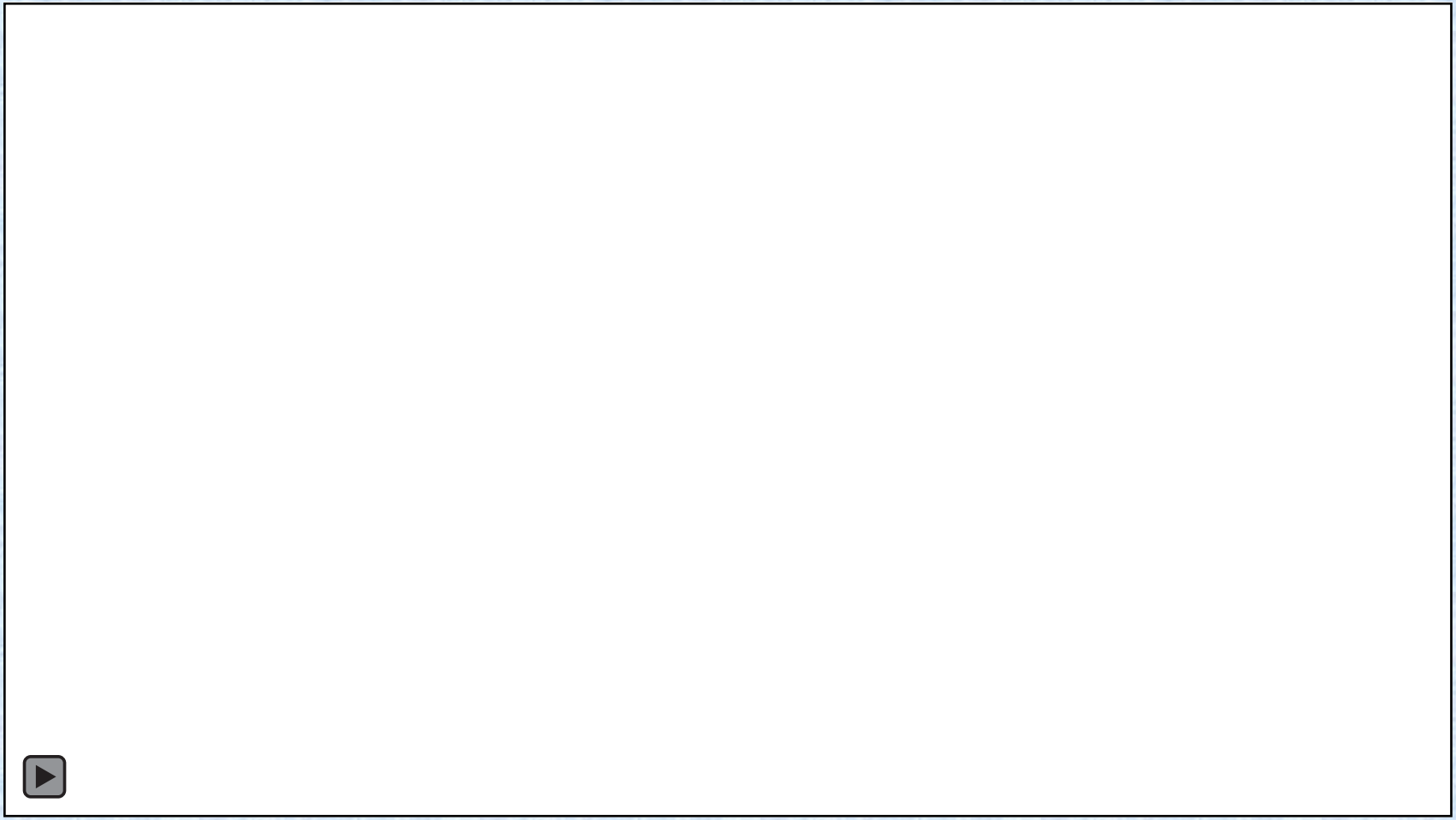
- Goal: Obtaining the unknown mapping f(x) from x to y.

Classifier

- Inputs – feature values
  - e.g, images, color, size, …
- Outputs – different classes.
  - character, word, decision (Y/N), …

Why do we need DNNs ?(1)

(In an application for speaker recognition)

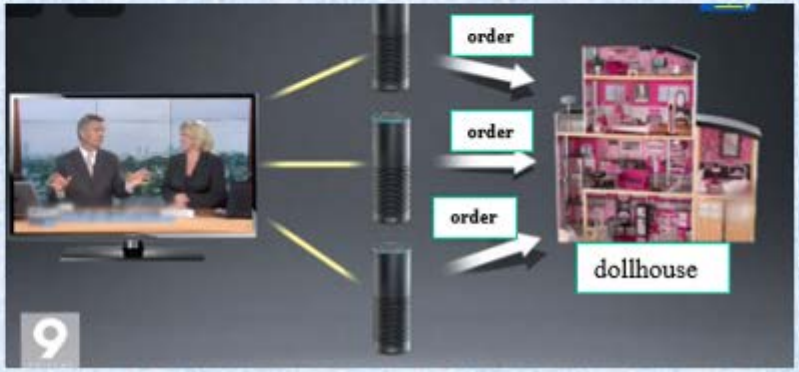


Why do we need DNNs ?(2)

(In an application for speaker recognition)

" Can you play dollhouse with me and get me a dollhouse?"

"Alexa ordered me a dollhouse and cookies"



Why do we need DNNs ?(3)

(In an application for speaker recognition)

- Speaker identification
  - Determines which registered speaker provides a given utterance from amongst a set of known speakers.
- Speaker verification
  - Accepts or rejects the identity claim of a speaker.
  - Is the speaker the person they say they are?

An application of DNNs (1)

(In a generative model)

Input (noise or code) ***x***

p(x) (i.e., f(x)) by a generative model

Output (generated image or speech/music) ***y***

- Goal: Obtaining the unknown distribution, *p(x)*, (i.e., a probability model of training inputs) from x to y.

Generative model ⇨

- Inputs – speech / music / images etc. to be modeled (for training)
- Inputs – noise or code (for testing to generate samples)
- Outputs – speech / music/ images etc. (looks/sounds like real samples)

An application of DNNs (2)
(In a generative model example)

- Which one is a fake face?

Real image is from FFHQ data set at https://drive.google.com/drive/folders/1u2xu7bSrWxrbUxk-dT-UvEJq8IjdmNTP

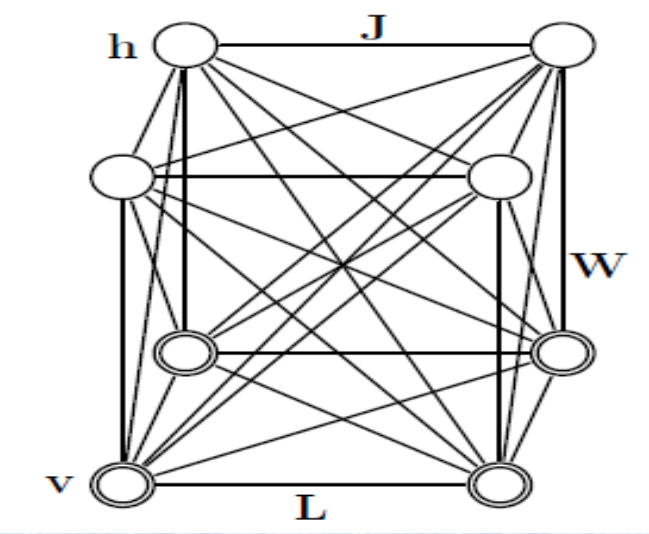
An application of DNNs (3)

(In a generative model example)

- Demonstration of piano music sounds generated by a DNN
- Demonstration of human speech generated by a DNN

An application of DNNs (4)

(In application of RBMs for handwritten digit recognition)

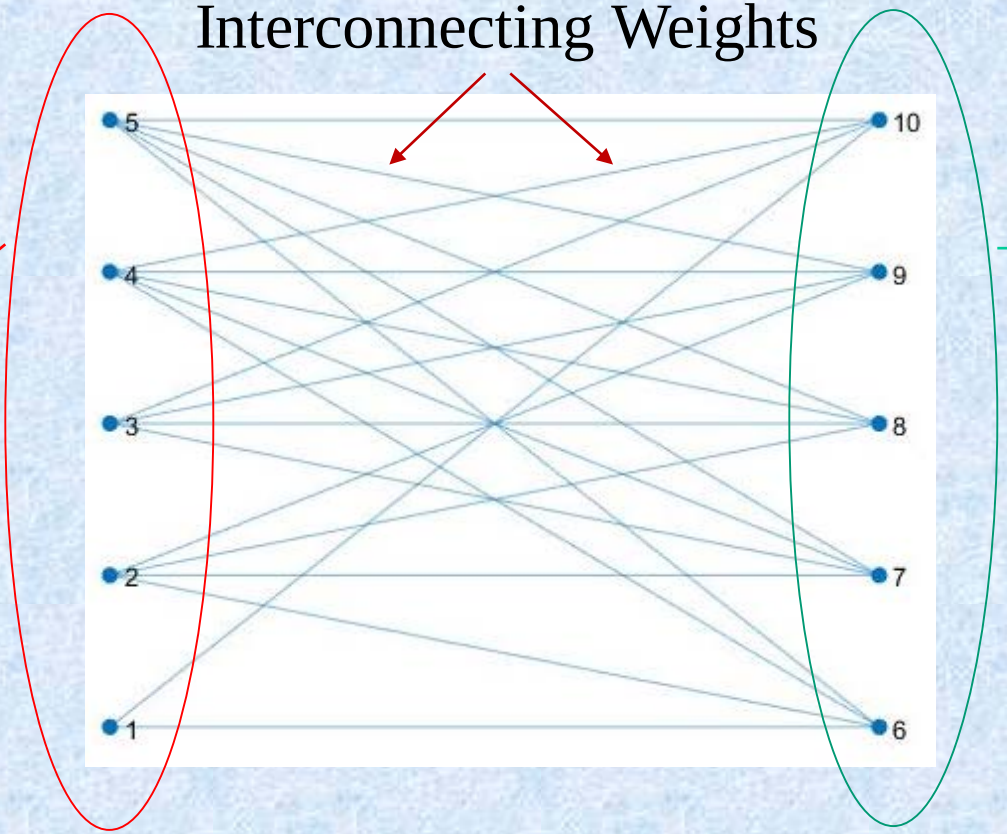


A Boltzmann Machine

Ref : R. Salakhutdinov and G. Hinton, "Deep Boltzmann Machines", 12th International Conference on Artificial Intelligence and Statistics, 2009.

An application of DNNs (5)

(In application of RBMs for handwritten digit recognition)

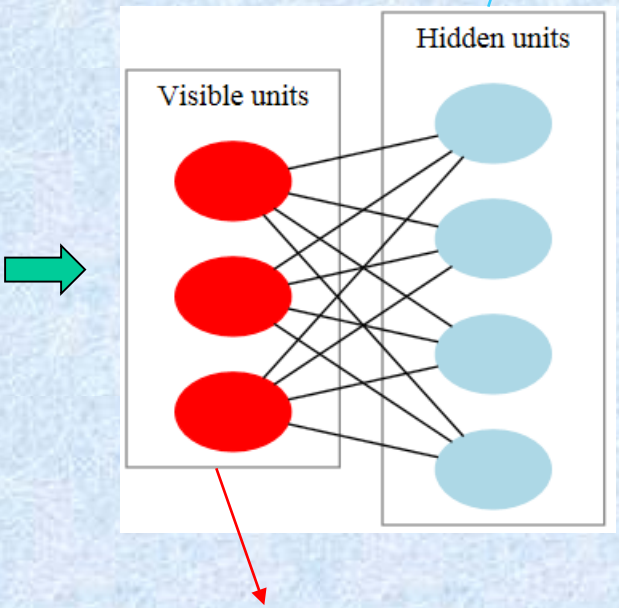


Note
Also “biases” are considered in all nodes

An application of DNNs (6)

(In application of RBMs for handwritten digit recognition)

500 hidden nodes



28 x 28 = 784 visible nodes

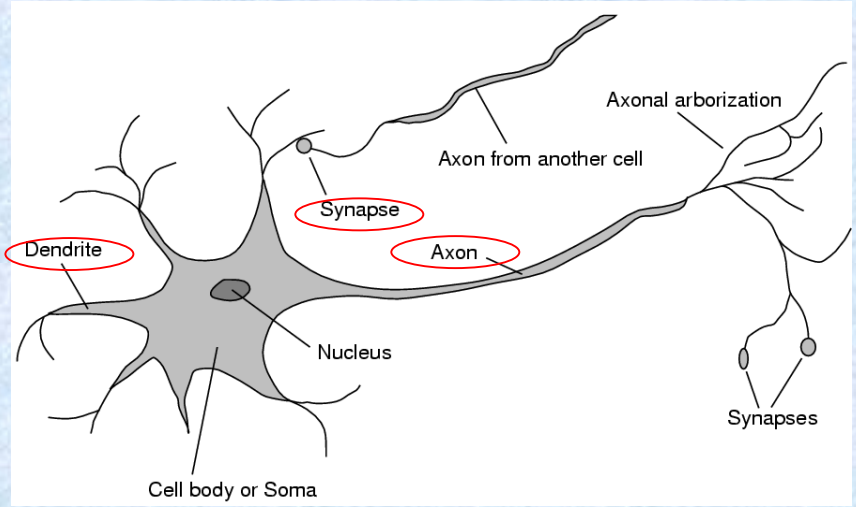
An application of DNNs (7)

(In application of RBMs for handwritten digit recognition)

- Demonstration to generate the learned handwritten digits using RBM

A neuron

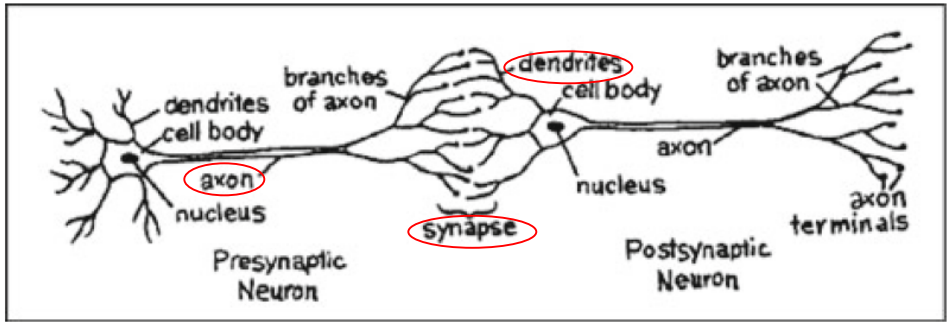
- A neuron is a nerve cell that is the basic building block of the biological nerve system.
- Processing (i.e., activation)
- Transmit information in both chemical and electrical forms.



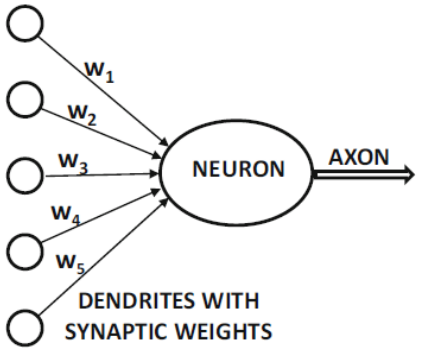
A neuron

Artificial Neural Network

- Artificial Neural Network is motivated by the biological neural network.
- Neurons are connected to one another through axions and dendrites.
- The connecting regions between axons and dendrites are referred to as “synapses”.
- Learning occurs by changing the connections with synaptic strength (i.e., weights in artificial NNs)

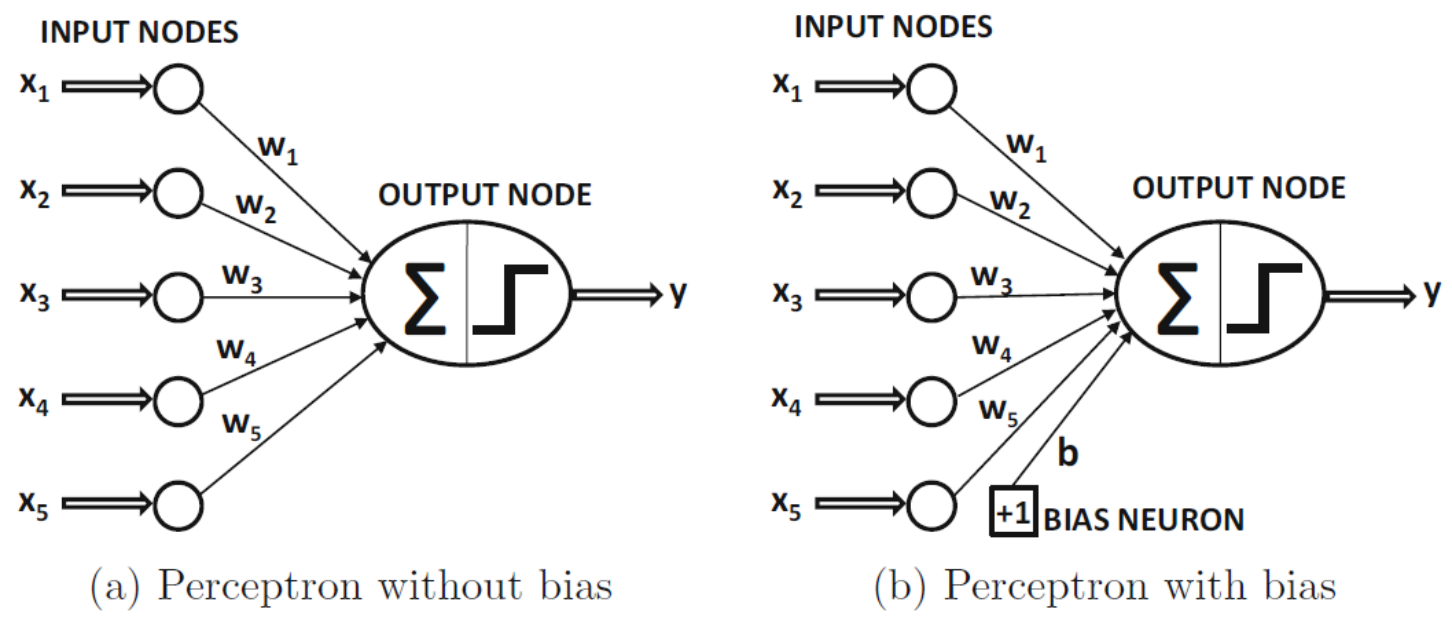


(a) Biological neural network

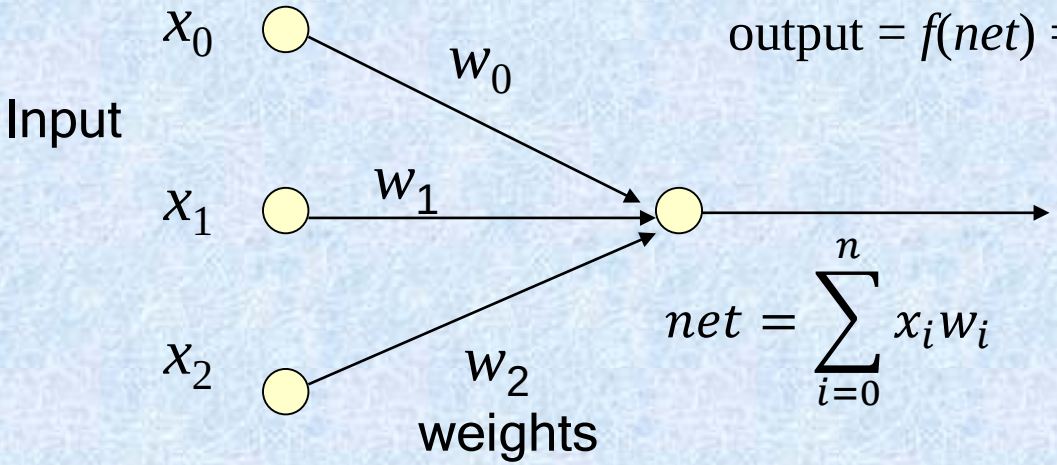


(b) Artificial neural network

Perceptron
(single layer neural network)

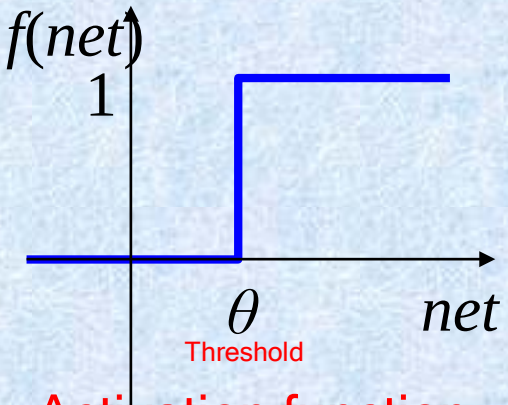


Perceptron



output = $f(net) = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$

"perceptron"



Activation function

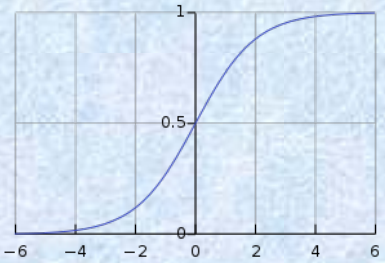
In vector forms,

$\boldsymbol{w} = [\, w_0, w_1, \ldots, w_N]$

$\boldsymbol{x}^{(i)} = [\, x_0, x_1, \ldots, x_N]$

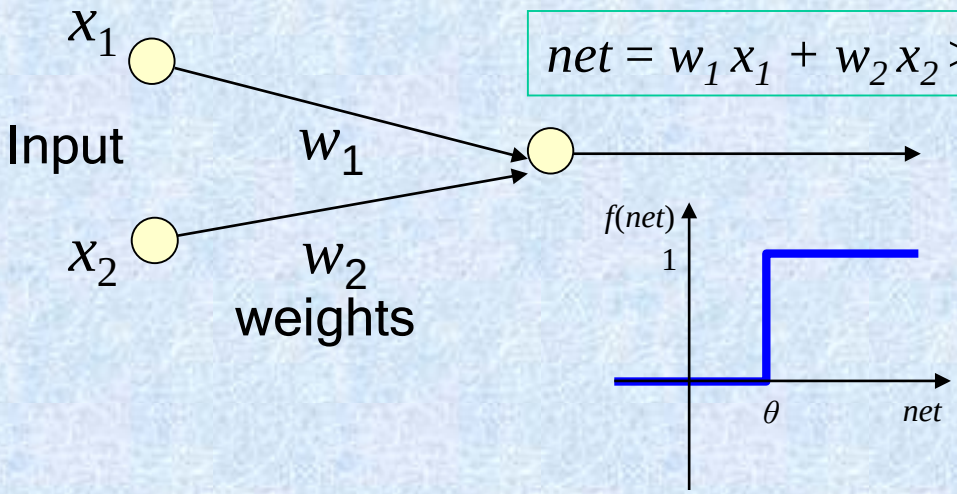
$net = \boldsymbol{w}^T \boldsymbol{x} + \theta$

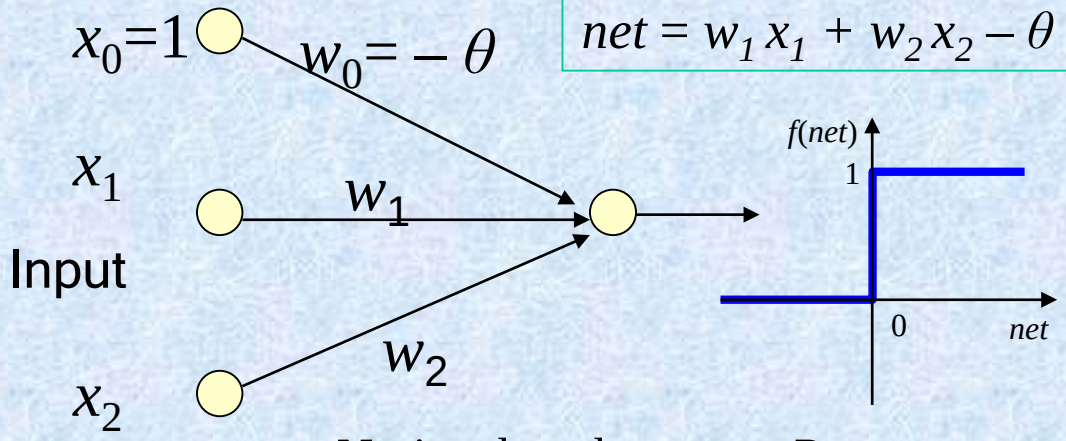
$f(\boldsymbol{x}^{(i)}; \boldsymbol{w}, \theta)$ = network output



Sigmoid function

Perceptron





if $net > \theta$ then $output = 1$
else output = 0

if $net - \theta > 0$ then $output = 1$
else output = 0

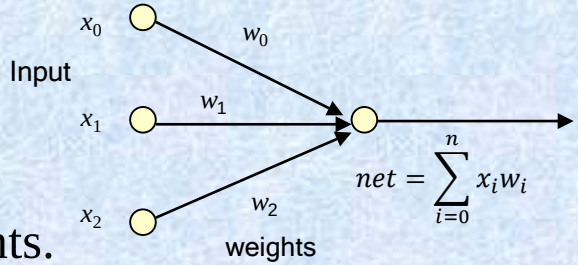
if $net + w_0 x_0 > 0$ then $output = 1$
else output = 0

if $net + w_0 > 0$ then $output = 1$
else output = 0

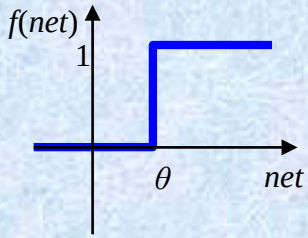
- Notice that above two Perceptrons work same.

Basic structure of Neural networks

- Perceptron (or single layer) in a NN
  - Input signals x_i
  - Weights w_i
    - Learning is achieved by changing these weights.
  - An activation level *net*
  - An activation function (threshold function)
    - If the activation level is higher than a threshold, then the perceptron is activated.



- (Deep) Neural Networks consist of many connecting Perceptrons, where followings must be considered:
  - Network topology
    - Connecting structure of many Perceptrons.
    - Feed forward, feed backward
  - Learning algorithm to get the weights.
  - Activation schemes.



Classification using a NN

input

Dog

Cat

Elephant

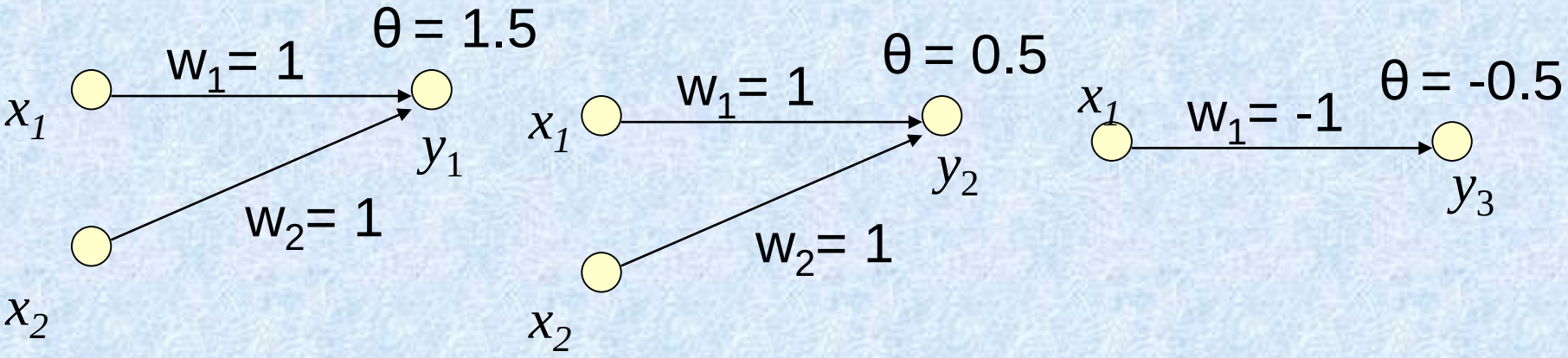
input

Human

Bus

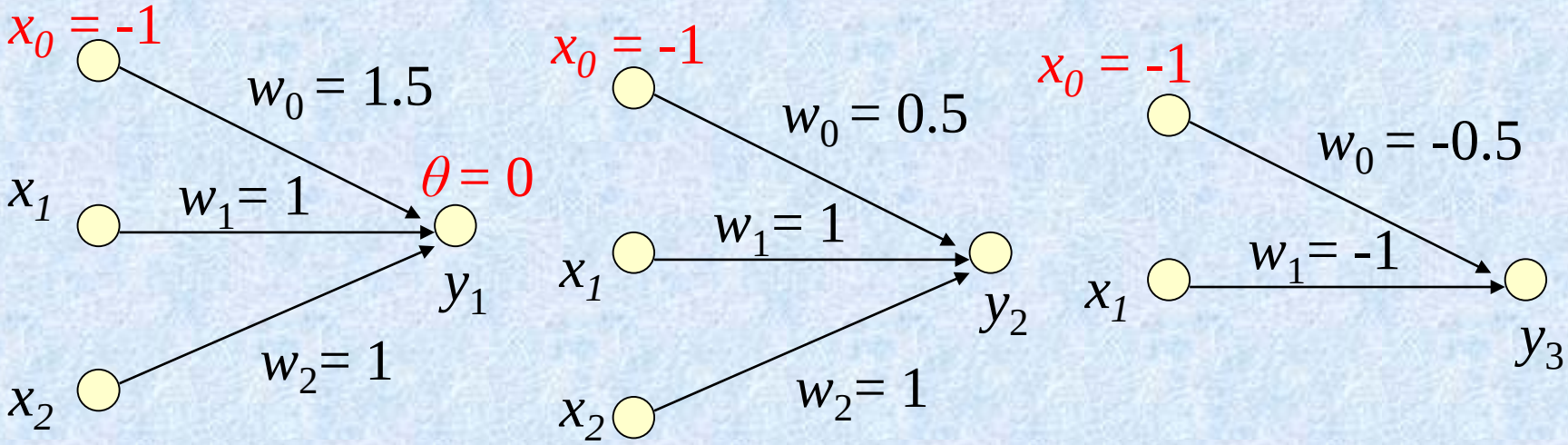
Animal

Example: Logic Gates



input		y_1		y_2		y_3	
x_1	x_2	net	out	net	out	net	out
0	0	0	0	0	0	0	1
0	1	1	0	1	1	0	1
1	0	1	0	1	1	-1	0
1	1	2	1	2	1	-1	0

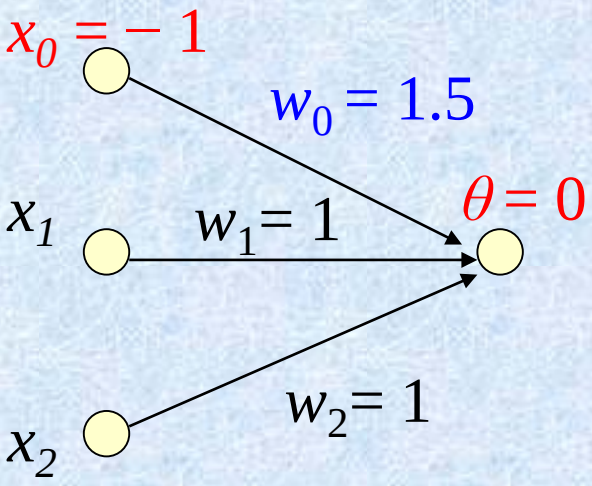
Logic Gates (Fix $\theta = 0$, then include w_0)



input		?		?		?	
x_1	x_2	net	out	net	out	net	out
0	0	-1.5	0	-0.5	0	0.5	1
0	1	-0.5	0	0.5	1	0.5	1
1	0	-0.5	0	0.5	1	-0.5	0
1	1	0.5	1	1.5	1	-0.5	0

What is perceptron?

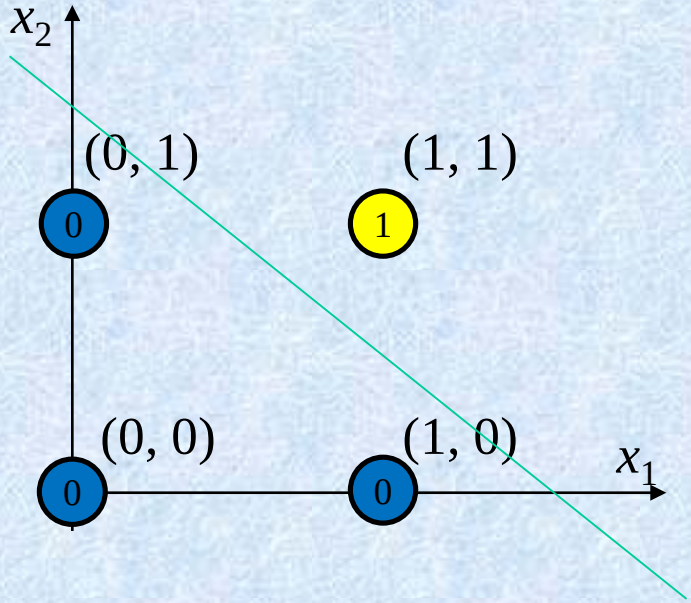
- AND gate



input		output
x_1	x_2	
0	0	0
0	1	0
1	0	0
1	1	1

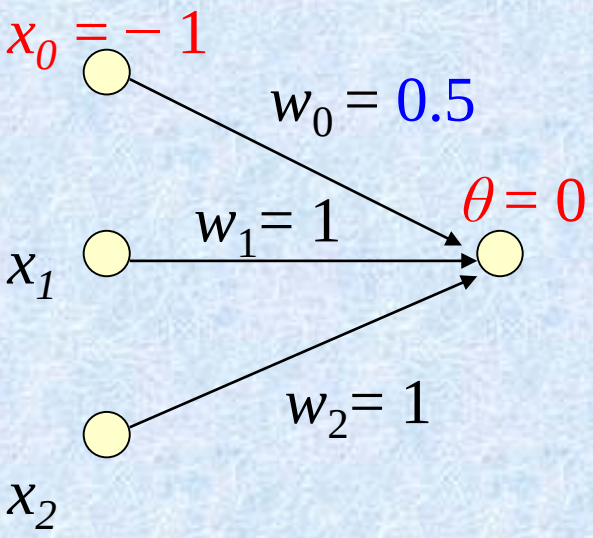
$$net = w_0 x_0 + w_1 x_1 + w_2 x_2$$
$$= -1.5 + x_1 + x_2 = 0$$

$$x_2 = -x_1 + 1.5$$



What is perceptron?

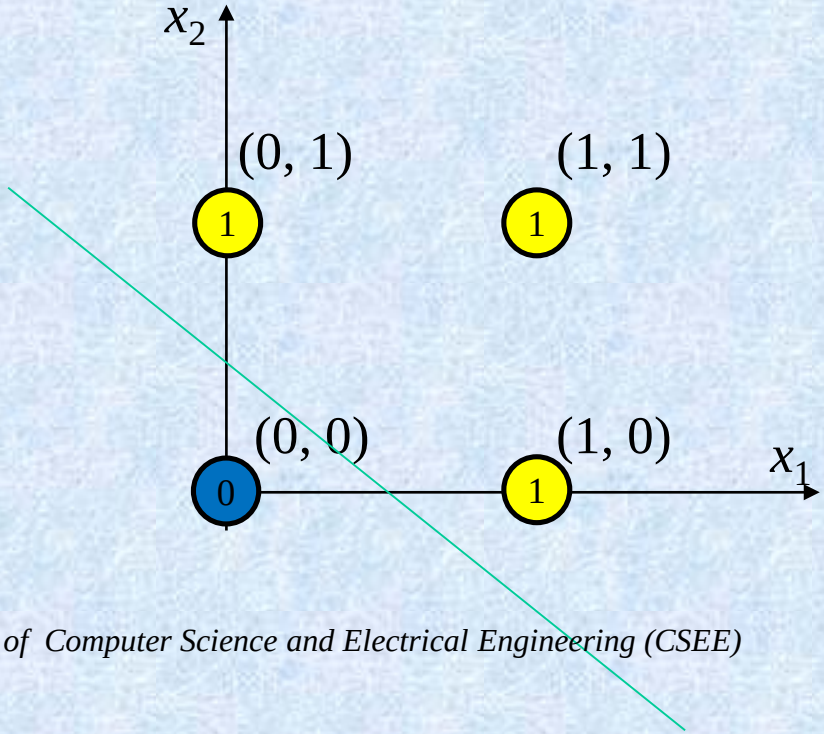
- OR gate



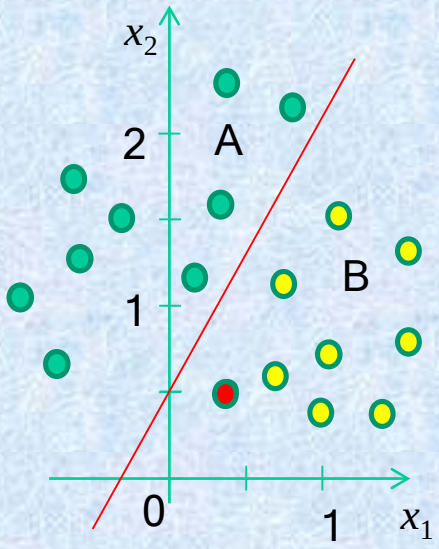
input		output
x_1	x_2	
0	0	0
0	1	1
1	0	1
1	1	1

$$net = w_0 x_0 + w_1 x_1 + w_2 x_2$$
$$= -0.5 + x_1 + x_2 = 0$$

$$x_2 = -x_1 + 0.5$$



Decision function



- A line that separates the two classes

  $x_2 = 1.5x_1 + 0.5$

- Decision function:

  $f(x_1, x_2) = -x_2 + 1.5x_1 + 0.5$

  class B if $f(x_1, x_2) \geq 0$

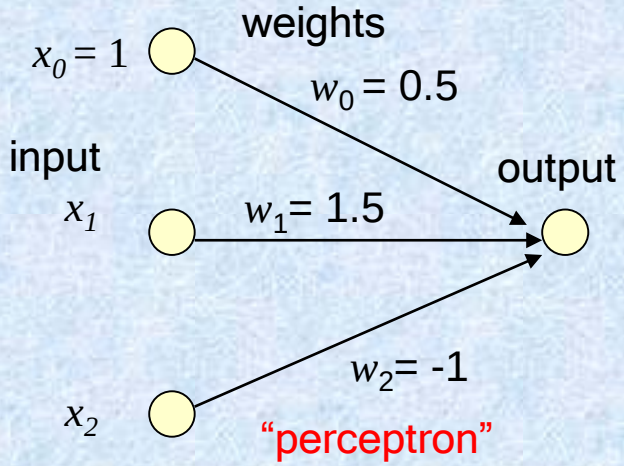
  class A if $f(x_1, x_2) < 0$

Example) a new sample (0.4, 0.5)

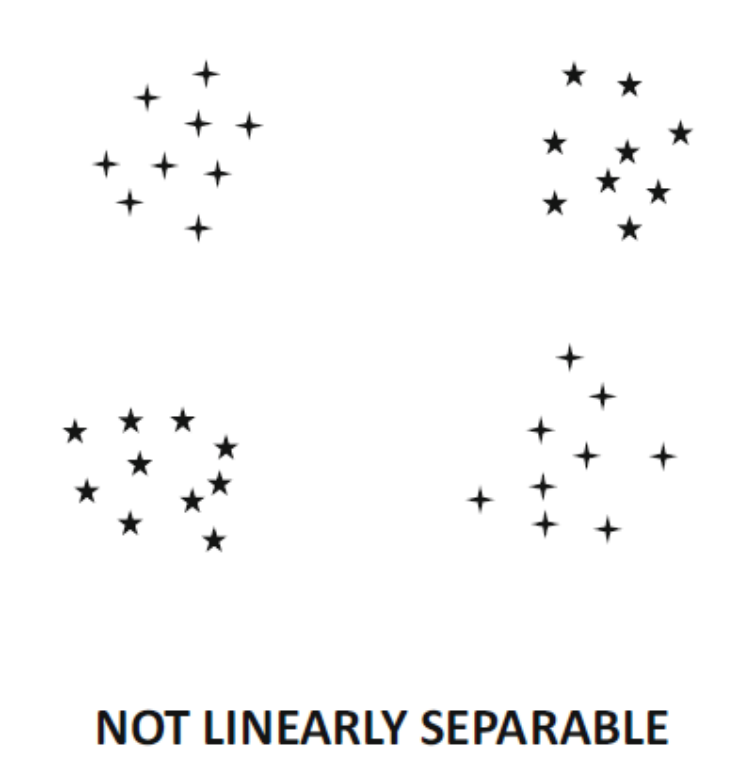
- $f(0.4, 0.5) = .0.5 + (1.5 \times 0.4) + 0.5 = 0.6$

  → class B

- Decision function → neuron



How about this data?

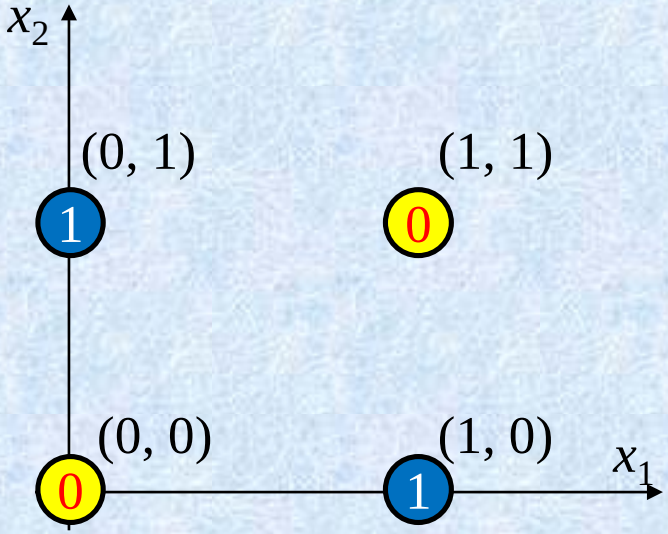


It shows the limitation of a perceptron, which necessitates the use of more complex neural architectures.

Limitations of the perceptron model

- Not linearly separable.

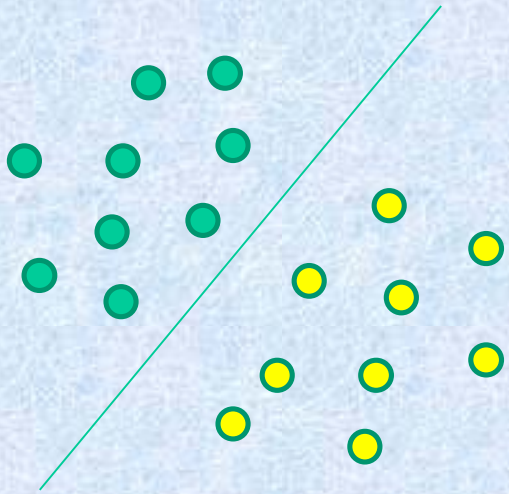
Input (x)	Output (y)
(0, 0)	0
(0, 1)	1
(1, 0)	1
(1, 1)	0



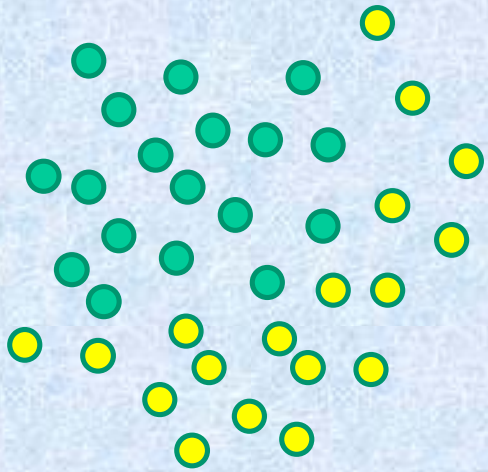
The truth table for exclusive-or (i.e., XOR).

Linear and non-linear problems

- Classification task
  - **Linear** problem : Samples or classes are separable with a line, plane, or hyperplane. ↔ **Non-linear** problem



linear problem

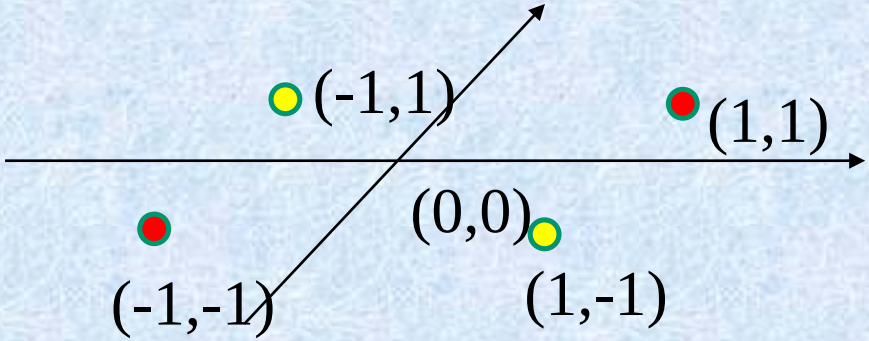


non-linear problem

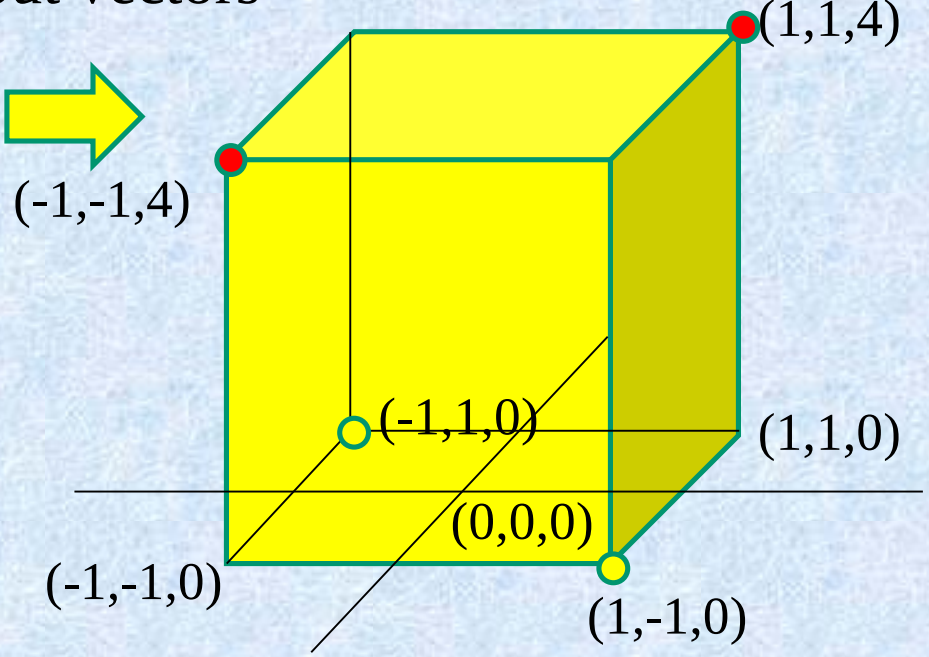
Solving non-linear problems with a neural network

1. Increase number of lines.
2. Increase dimension of input vectors

Augmenting input $(x_1+x_2)^2$



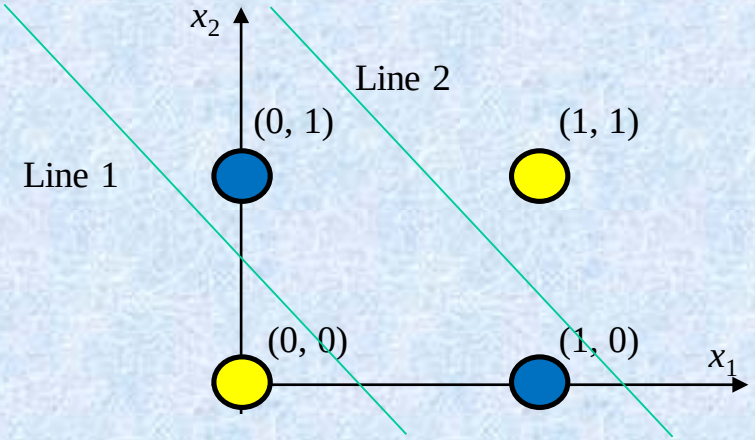
Not separable with one line (non-linear)



Separable with one plane (linear)

Solving non-linear problem with a neural network

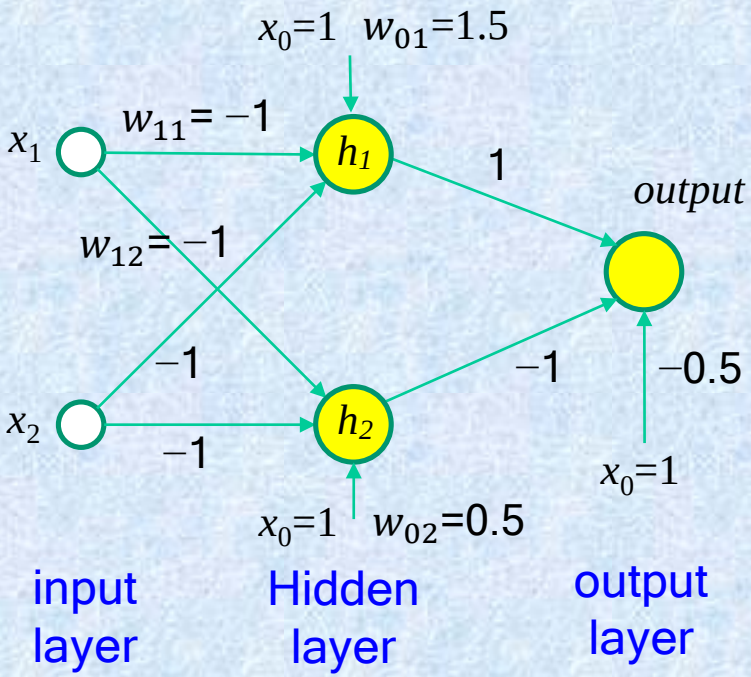
- Solution : uses two lines to separate the data
- Two lines → two nodes → multilayered perceptron
- Require
  - Two hidden nodes, both connected to two input nodes to represent the two lines
  - A third node to combine the outputs from the two hidden (middle) nodes.



The XOR problem solved with two separating lines.

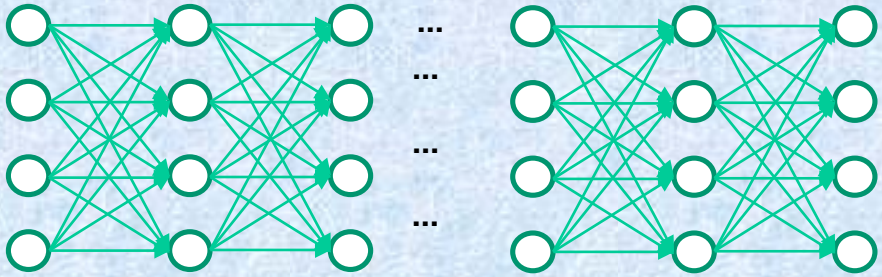
Network used to model the XOR relation

Two-layered perceptron



x_1	x_2	net_1	h_1	net_2	h_2	net_3	output
0	0						
1	0						
0	1						
1	1						

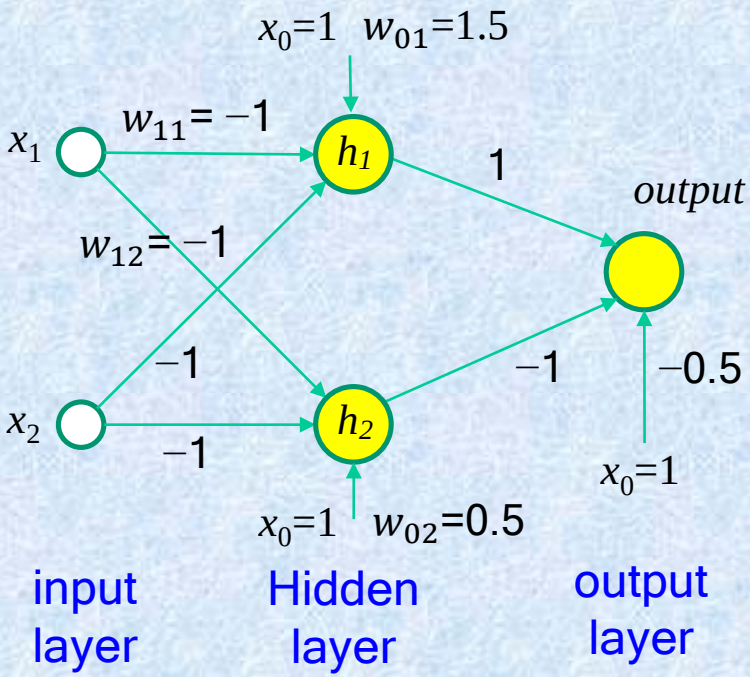
$$net_j = \sum_{i=0}^{n} x_i w_i \qquad h_1 = f(\text{net}_1)$$



Deep neural networks

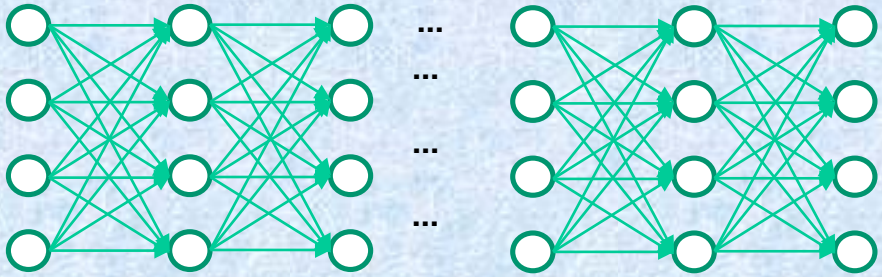
Network used to model the XOR relation

Two-layered perceptron



x_1	x_2	net_1	h_1	net_2	h_2	net_3	output
0	0	1.5	1	0.5	1	-0.5	0
1	0	0.5	1	-0.5	0	0.5	1
0	1	0.5	1	-0.5	0	0.5	1
1	1	-0.5	0	-1.5	0	-0.5	0

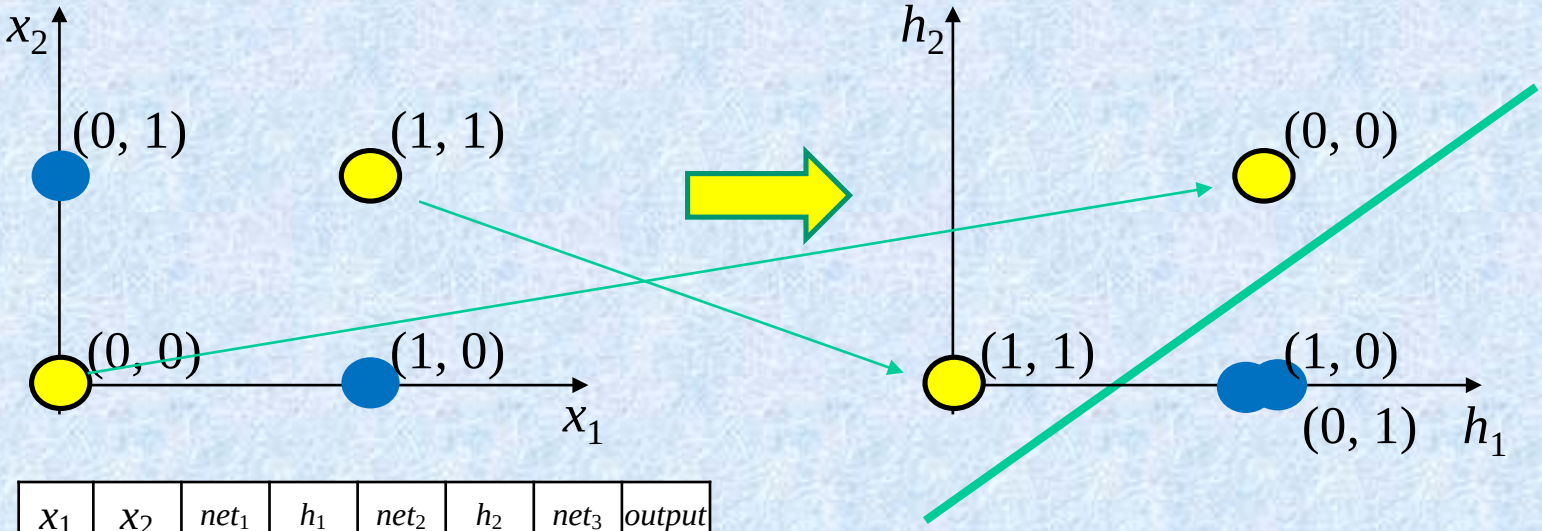
$$net_j = \sum_{i=0}^{n} x_i w_i \qquad h_1 = f(\text{net}_1)$$



Deep neural networks

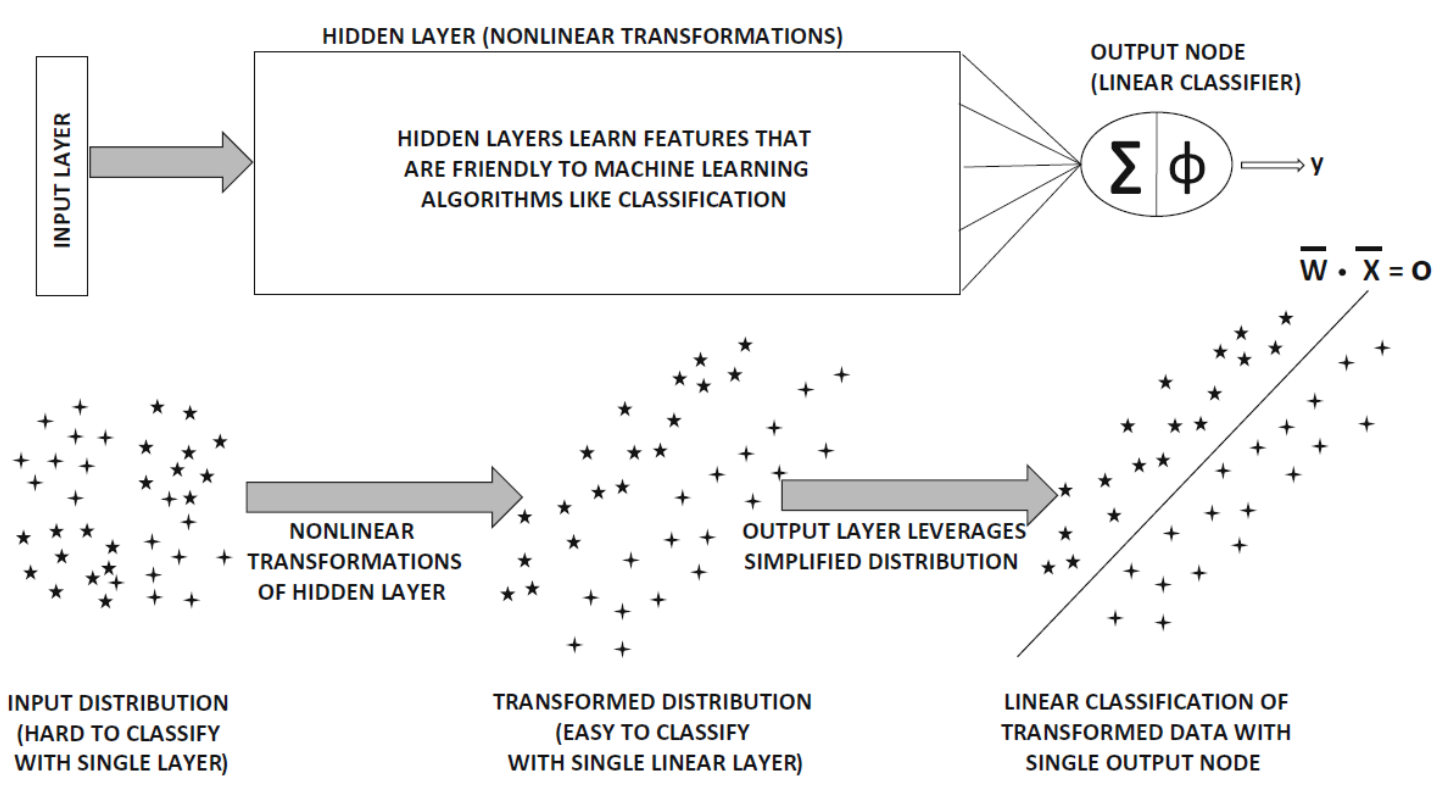
The role of the first layer ?

- Input data are mapped to different input data, which are linearly separable.



x_1	x_2	net_1	h_1	net_2	h_2	net_3	*output*
0	0	1.5	1	0.5	1	-0.5	0
1	0	0.5	1	-0.5	0	0.5	1
0	1	0.5	1	-0.5	0	0.5	1
1	1	-0.5	0	-1.5	0	-0.5	0

The general role of hidden layers



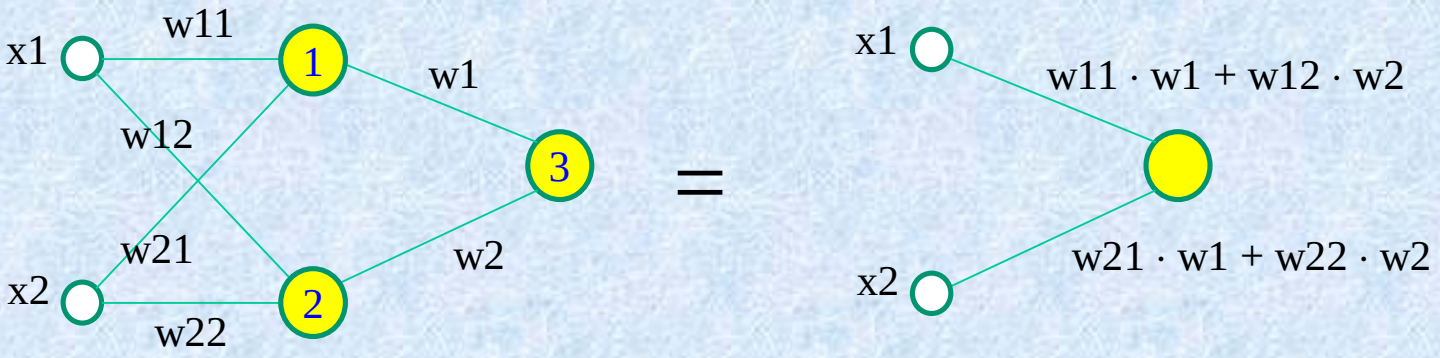
The output of each hidden layer is a transformed feature representation of the data.

Necessity of non-linear activation functions

- What if there is no non-linear activation function in a NN?
- Linear activation functions → a single layer of weights

∴ In a multilayered network, non-linear activation functions are required.

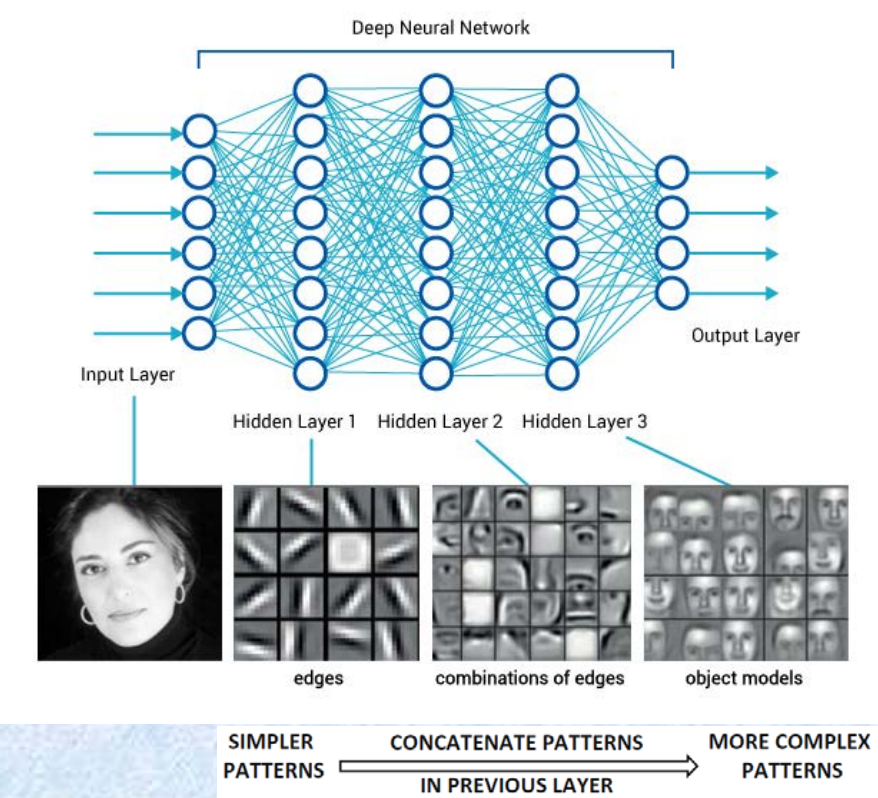
net3 = net1 w1 + net2 w2
= (x1 w11 + x2 w21) w1 + (x1 w12 + x2 w22) w2 → single layer !!
= x1 (w11 w1 + w12 w2) + x2 (w21 w1 + w22 w2)



Necessity of non-linear activation functions

- **Theorem:** *A multi-layer network that uses only the identity activation function in all its layers reduces to a single-layer network performing linear regression.* (HW: Prove this theorem. Don't need a submission)
- As we saw it in an example, these nonlinear activation functions play an important role in increasing the modeling power of a network. If the linear activation is only used, then it would not provide better modeling power than a single layer linear network.
- The main issue is that the number of hidden units required to do so is rather large, which increases the number of parameters to be learned. This results in practical problems in training the network with a limited amount of data. In fact, deeper networks are often preferred because they reduce the number of hidden units in each layer as well as the overall number of parameters.

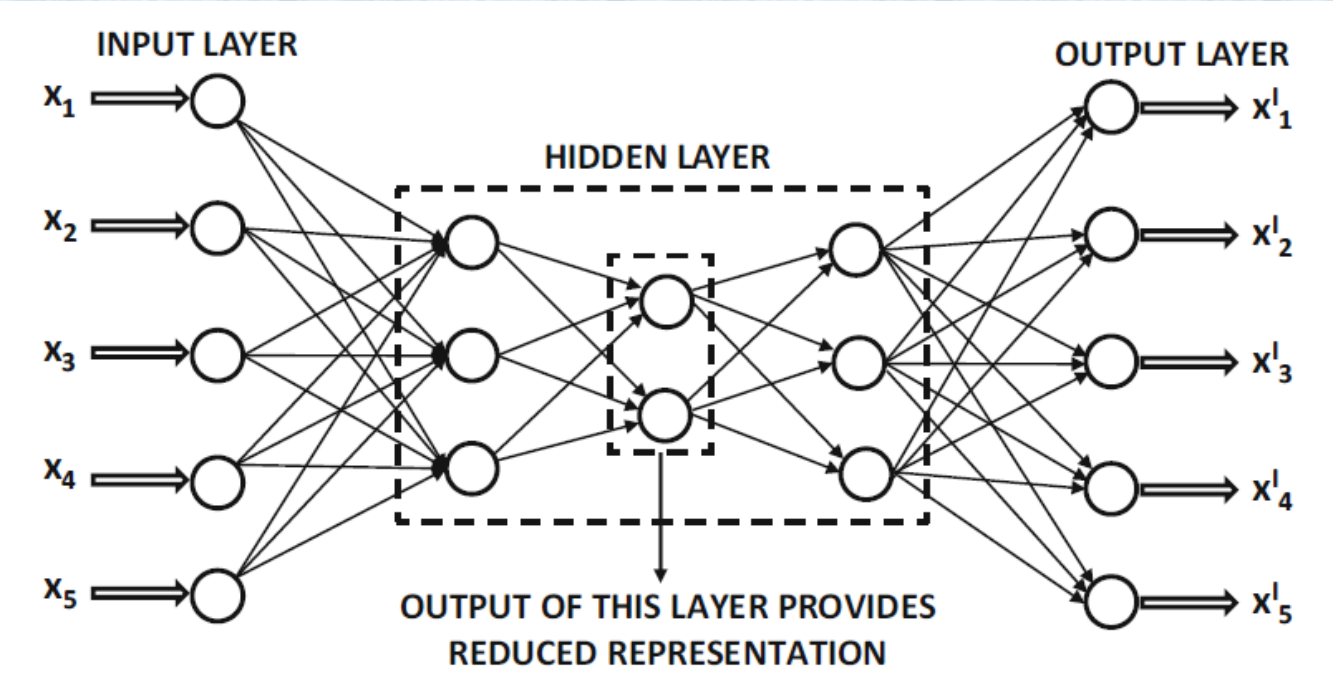
Layers and their abstraction power in a deep neural network



1. The features in earlier layers are used repeatedly as building blocks to create more complex features.

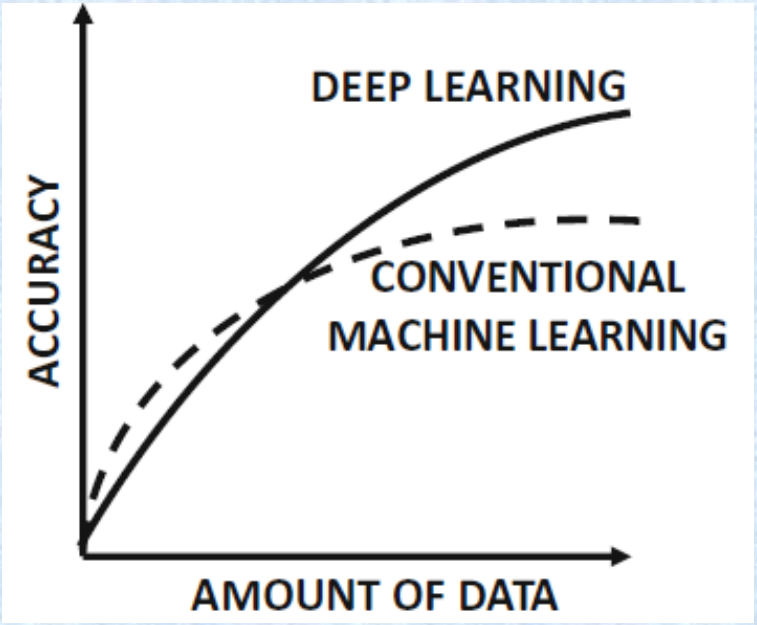
2. This general principle of "putting together" simple features to create more complex features lies at the core of the successes achieved with deep neural networks.

Dimensionality reduction with a deep neural network



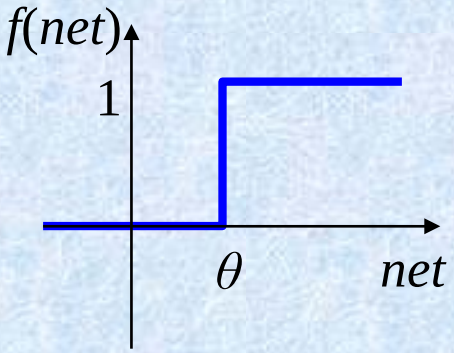
Autoencoder

Deep vs simple NNs

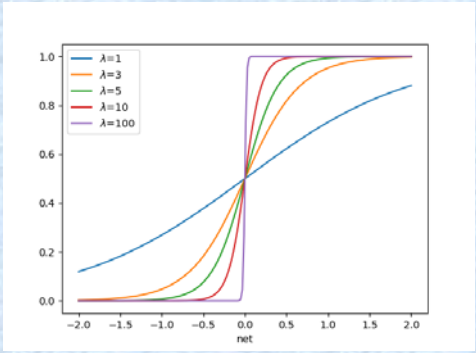


- Conventional Machine Learning (e.g., Supporting Vector Machine, linear regression etc.) implies simple neural network architectures (i.e., small nodes and shallow NNs)

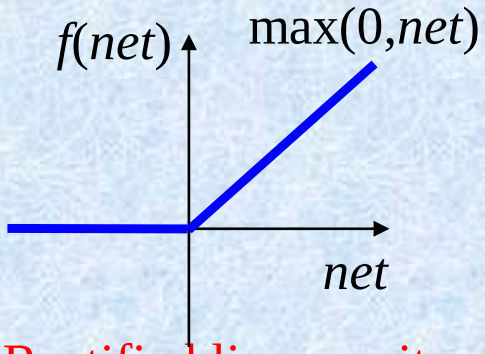
Activation (Thresholding) functions (1)



Hard limiting



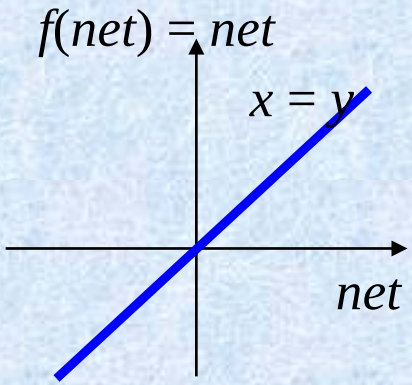
Sigmoid



Rectified linear unit (ReLU)

$$\sigma(net) = \frac{1}{1 + e^{-\lambda \cdot net}}$$

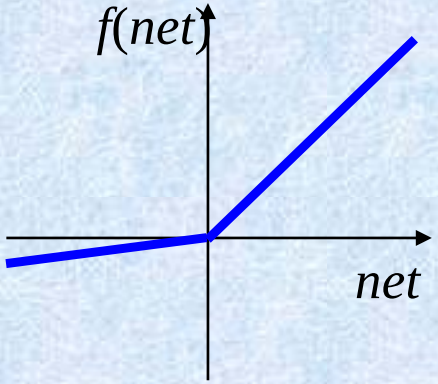
$$net = \sum_{i=0}^{n} x_i w_i$$



Linear

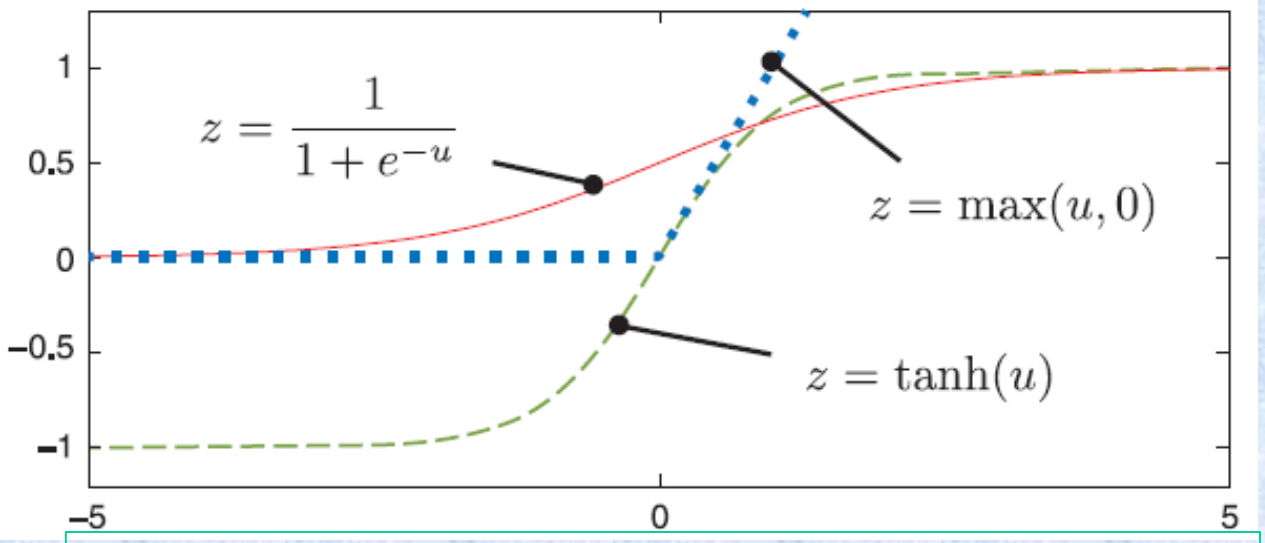
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

Maxout



Leaky Rectified linear unit

Activation (Thresholding) functions (2)



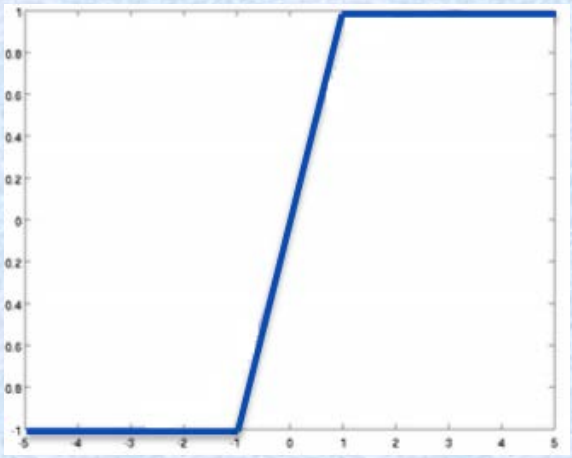
Nikai Etchuzi (translated in Korean by Myung-Jo Jin), "Deep Learning Getting Started with TensorFlow", ISBN: 9784839960889, Jpub, 2017

$$\Phi(v) = \frac{e^{2v} - 1}{e^{2v} + 1} \text{ (tanh function)}$$

$$\tanh(v) = 2 \cdot \text{sigmoid}(2v) - 1$$

Activation (Thresholding) functions (3)

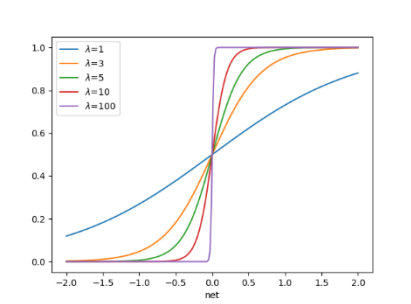
$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$



Instead of sigmoid and tanh, the ReLU and hard tanh are good substitutes respectively for the ease in training in Multilayer NN with those activation functions.

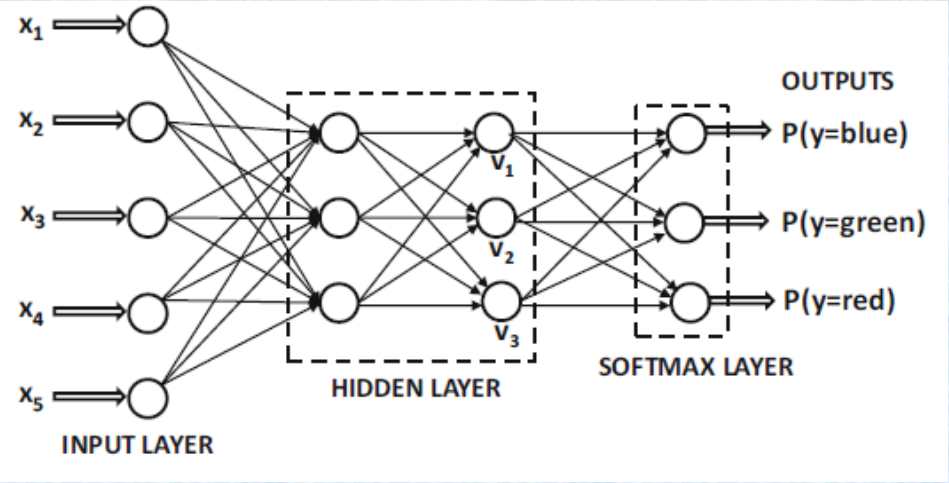
Sigmoid function

- Also known as *Logistic* function
- λ : "squashing parameter" used to fine-tune the sigmoidal curve.
  - Controls the slope of the function
  - Small value → linear threshold function over {0,1};
- Continuous thresholding function.



Softmax activation function

- For a k-way classification task, k output values can be used with a softmax activation function at the nodes in a given layer as follows:



$$\Phi(\overline{v})_i = \frac{\exp(v_i)}{\sum_{j=1}^{k} \exp(v_j)} \quad \forall i \in \{1, \ldots, k\}$$

Softmax function

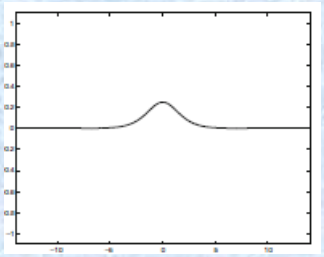
Derivatives of activation functions (HW)

- Sigmoid function, $\Phi(v) = \dfrac{1}{1+e^{-v}}$

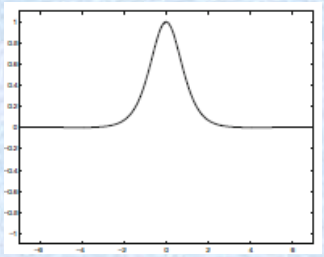
$$\frac{\partial \Phi}{\partial v} = \Phi'(v) = \frac{e^{-v}}{(1+e^{-v})^2} = \Phi(v)\big(1-\Phi(v)\big)$$

- Tanh, $\Phi(v) = \dfrac{e^{2v}-1}{e^{2v}+1}$

$$\frac{\partial \Phi}{\partial v} = \Phi'(v) = \frac{4e^{2v}}{(e^{2v}+1)^2} = 1 - \Phi^2(v)$$



Derivative of sigmoid



Derivative of tanh

Charu C. Aggarwal, "Neural Networks and Deep Learning", 2018