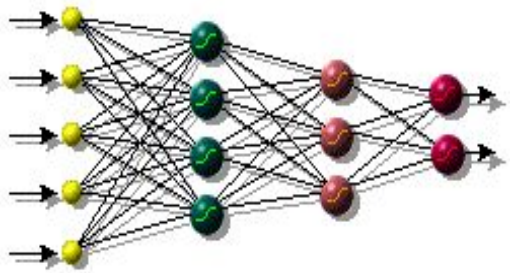
Artificial Neural Networks



Dinesh K. Vishwakarma, Ph.D.

ASSOCIATE PROFESSOR, DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY, DELHI.

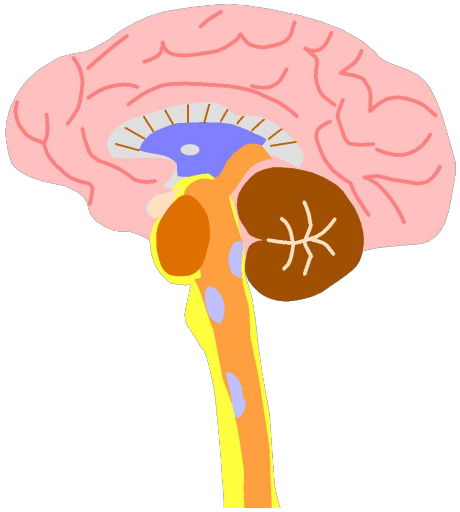
Webpage: http://www.dtu.ac.in/Web/Departments/InformationTechnology/faculty/dkvishwakarma.php

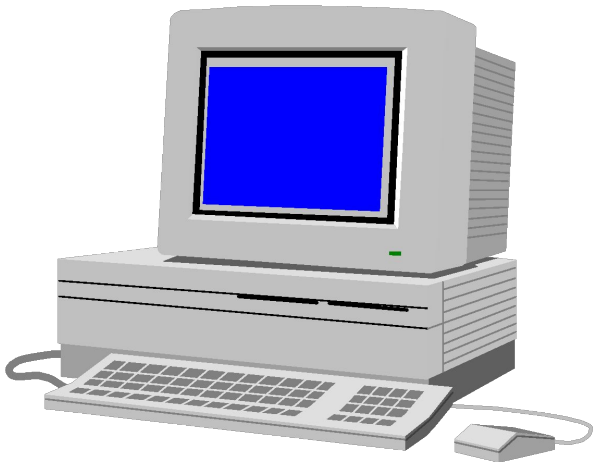
Artificial Neural Networks

- Other terms/names
  - ✔ Connectionist
  - ✔ Parallel distributed processing
  - ✔ Neural computation
  - ✔ Adaptive networks..
- History
  - 1943-McCulloch & Pitts are generally recognised as the designers of the first neural network.
  - 1949-First learning rule
  - 1969-Minsky & Papert - perceptron limitation - Death of ANN
  - 1980's - Re-emergence of ANN - multi-layer networks

Brain and Machine

- **The Brain**
  - ✔ Pattern Recognition
  - ✔ Association
  - ✔ Complexity
  - ✔ Noise Tolerance

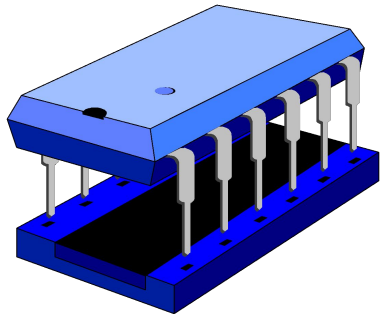


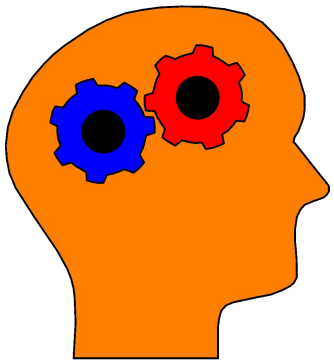


- **The Machine**
  - ✔ Calculation
  - ✔ Precision
  - ✔ Logic

The contrast in architecture

- **The Von Neumann architecture uses a single processing unit;**
  - ✔ Tens of millions of operations per second.
  - ✔ Absolute arithmetic precision.



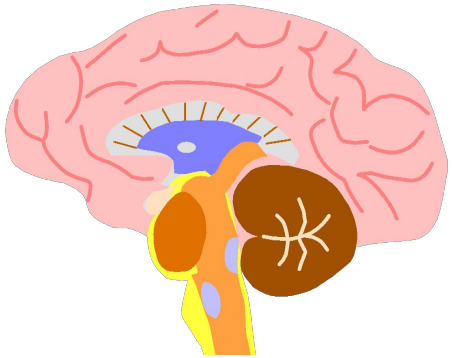


- **The brain uses many slow unreliable processors acting in parallel.**

Features of the Brain

- Ten billion (1010) neurons.
- On average, several thousand connection
- Hundreds of operations per second.
- Die off frequently (never replaced).
- Compensates for problems by massive parallelism.

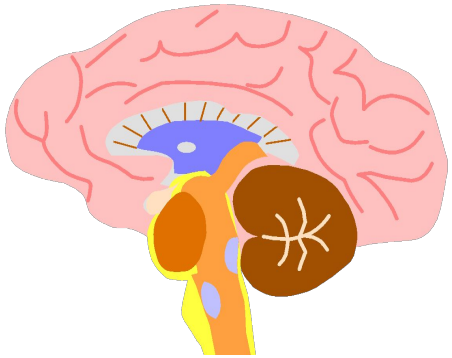
The biological inspiration



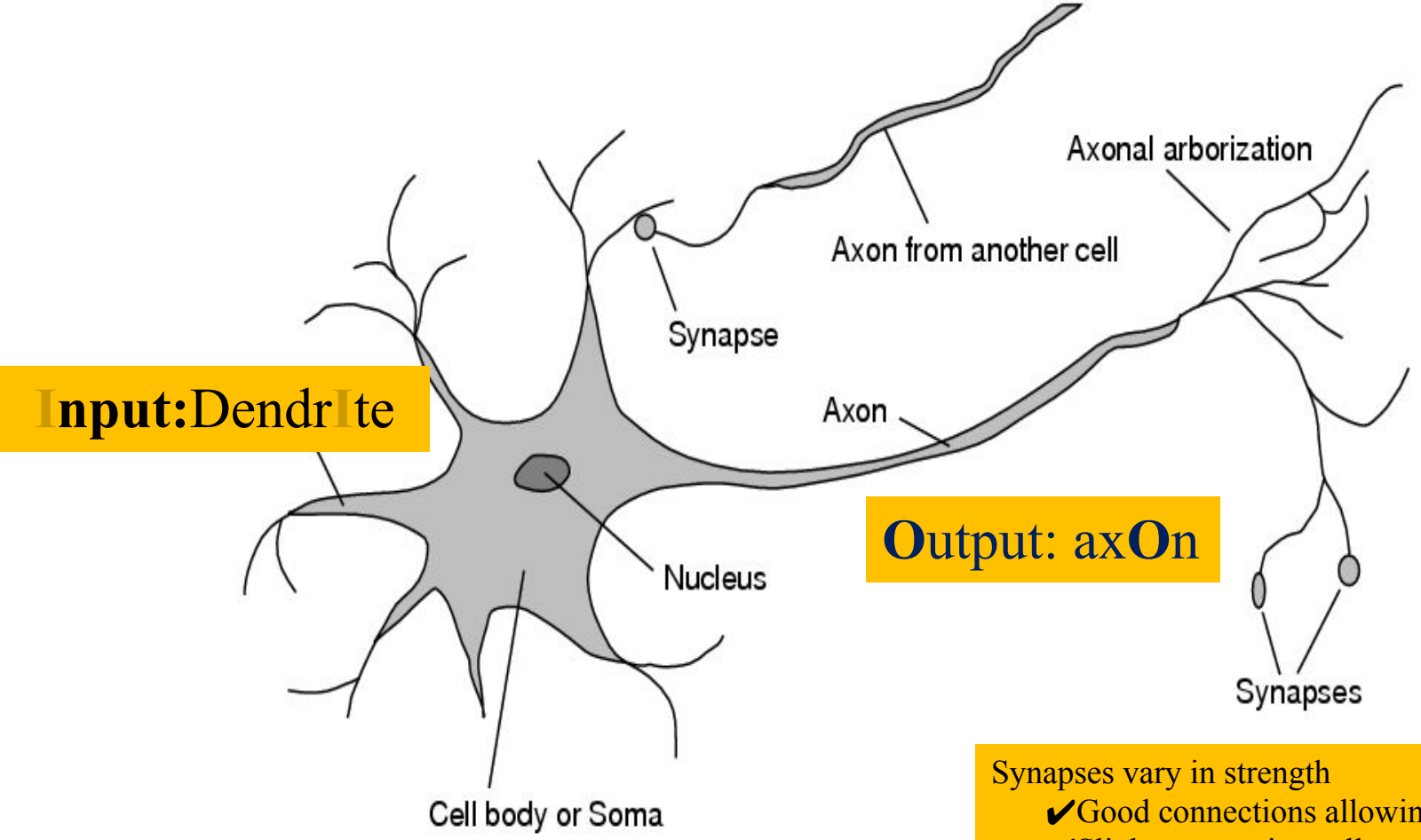
- The brain has been extensively studied by scientists.
- Vast complexity prevents all but rudimentary understanding.
- Even the behaviour of an individual neuron is extremely complex.

The biological inspiration…

- Single “percepts” distributed among many neurons
- Localized parts of the brain are responsible for certain well-defined functions (e.g. vision, motion).

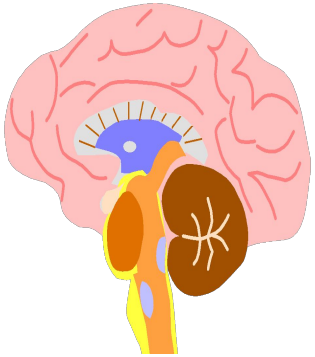


The Model of Neuron



Input: DendrIte

Output: axOn



Synapses vary in strength
- ✔Good connections allowing a large signal
- ✔Slight connections allow only a weak signal.

The Model of Neuron

- A neuron has a cell body, a branching input structure (the dendrIte) and a branching output structure (the axOn).

✔ Axons connect to dendrites via synapses.

✔ Electro-chemical signals are propagated from the dendritic input, through the cell body, and down the axon to other neurons.

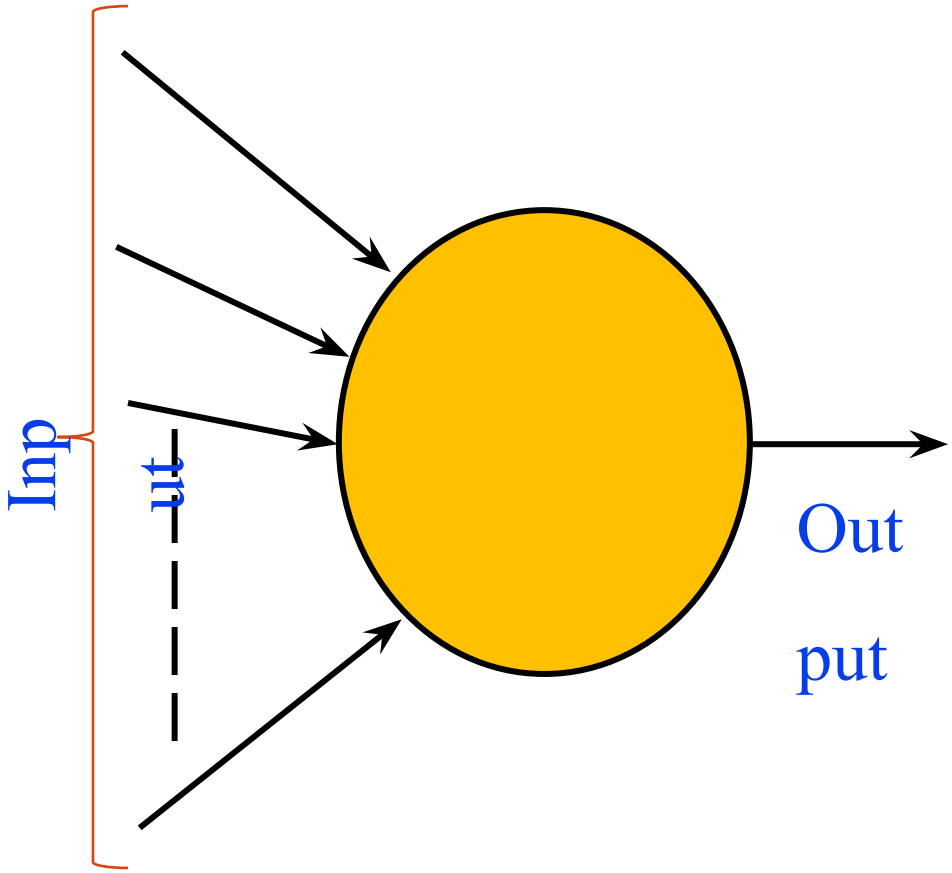
- A neuron only fires if its input signal exceeds a certain amount (the threshold) in a short time period.
- Synapses vary in strength

✔ Good connections allowing a large signal

✔ Slight connections allow only a weak signal.

The Artificial Neuron

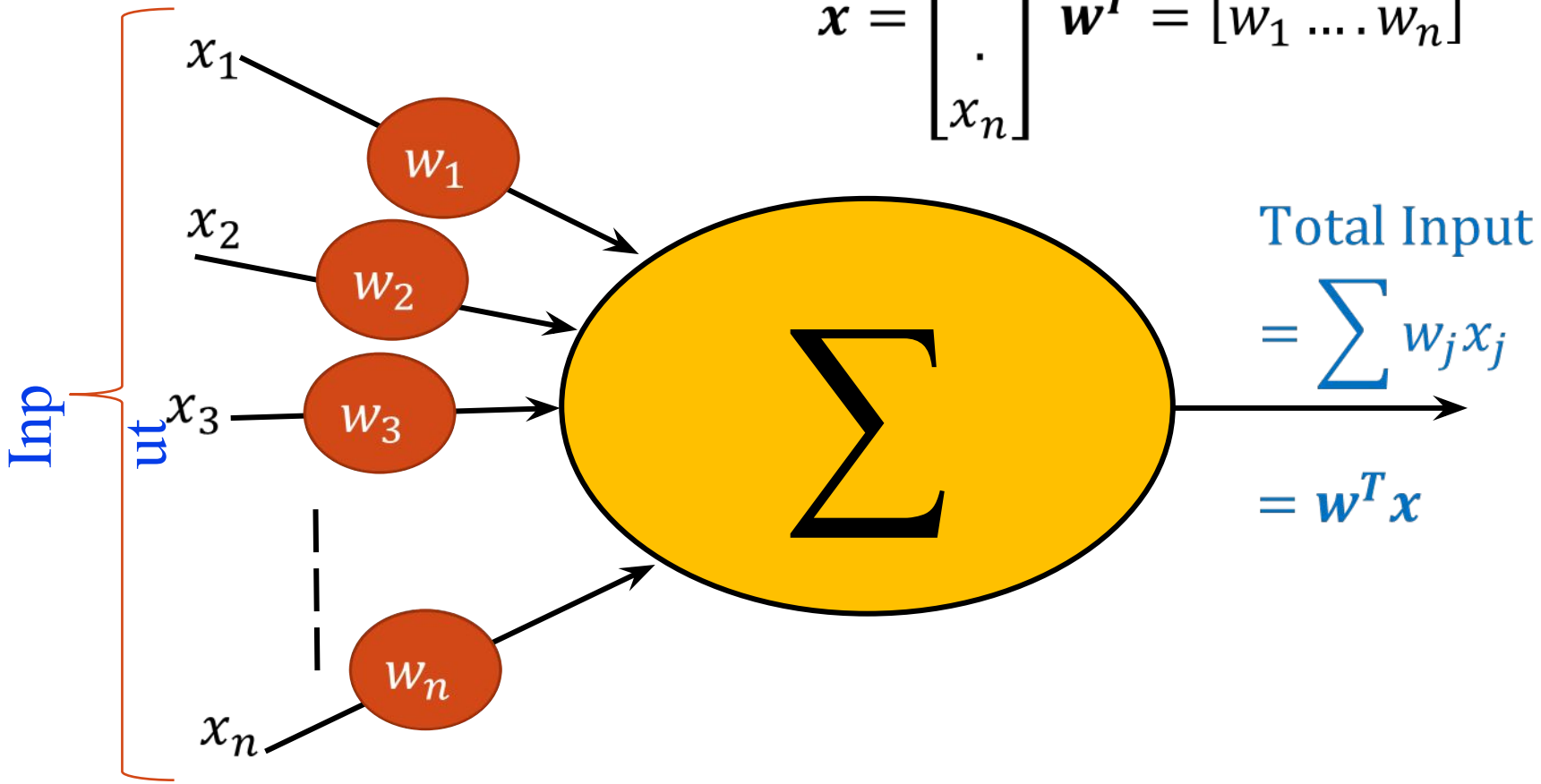
- **Input/output**
  - ✔ Evaluate the Input Data/Signal and determining the strength of each one.
  - ✔ Compute a total for the combined input data/signal and compares that total to some threshold.
  - ✔ Find, what output should be



The Artificial Neuron…

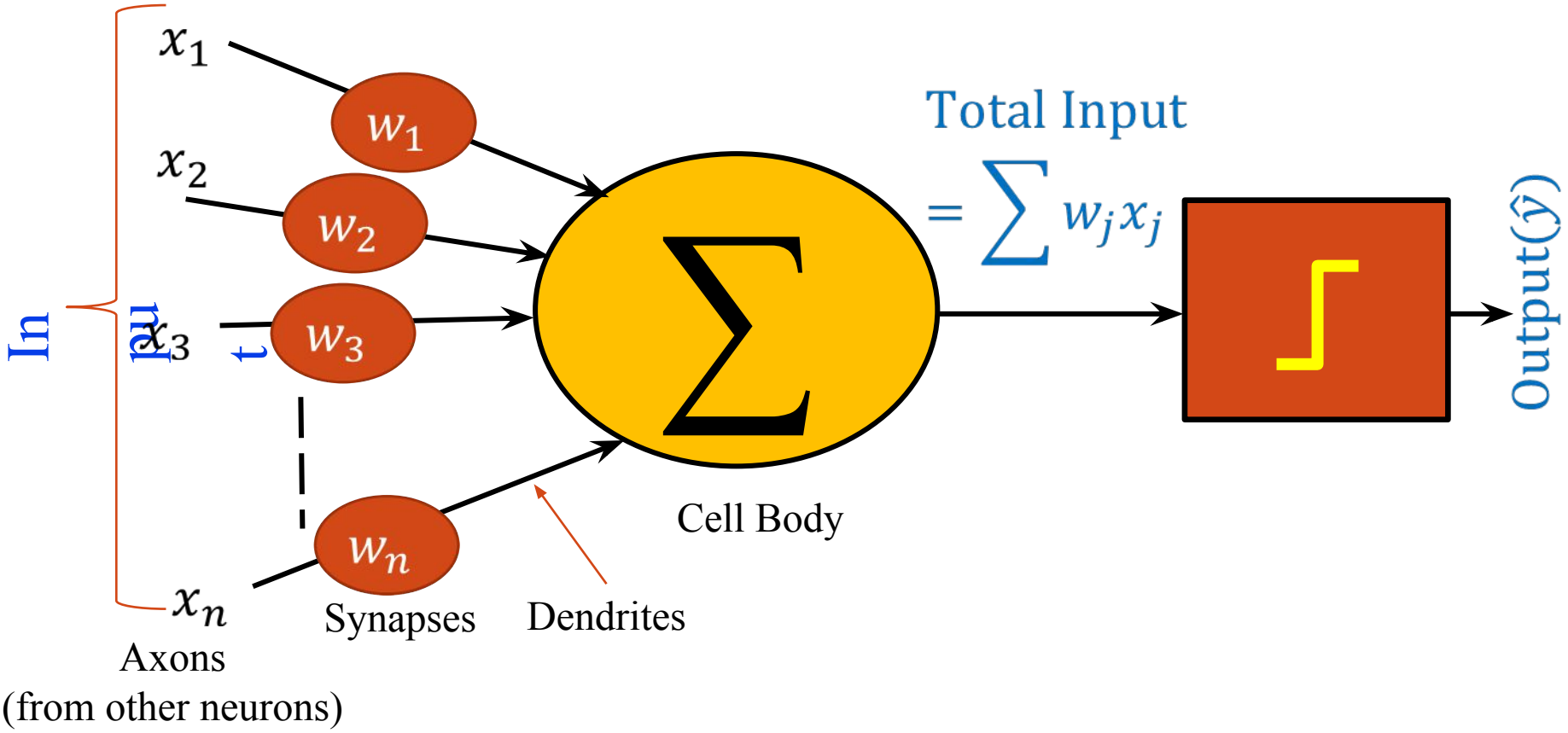
- **Weighting Factors**

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ . \\ . \\ x_n \end{bmatrix} \quad \boldsymbol{w^T} = [w_1 \ldots . w_n]$$



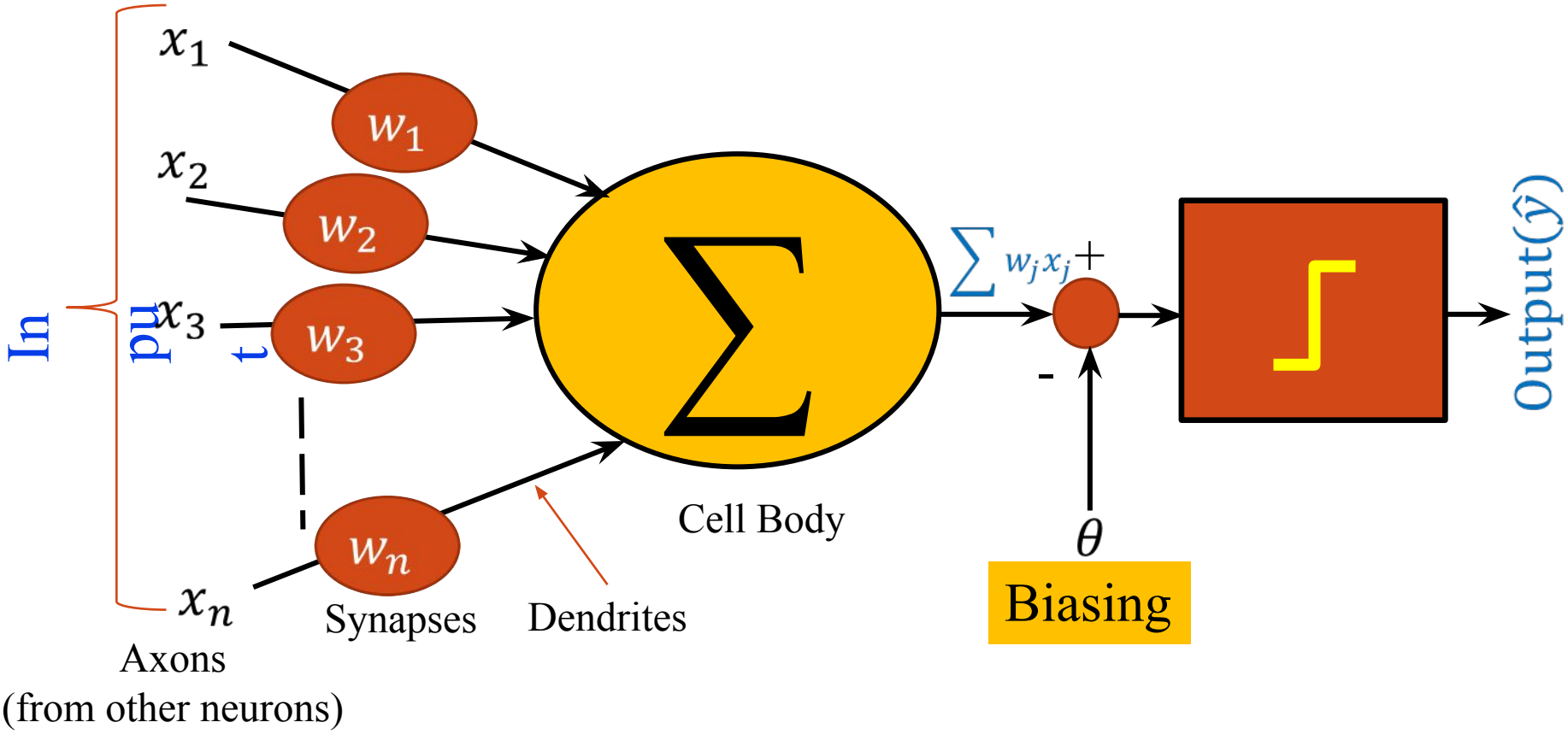
The Artificial Neuron…

- **Activation Functions**



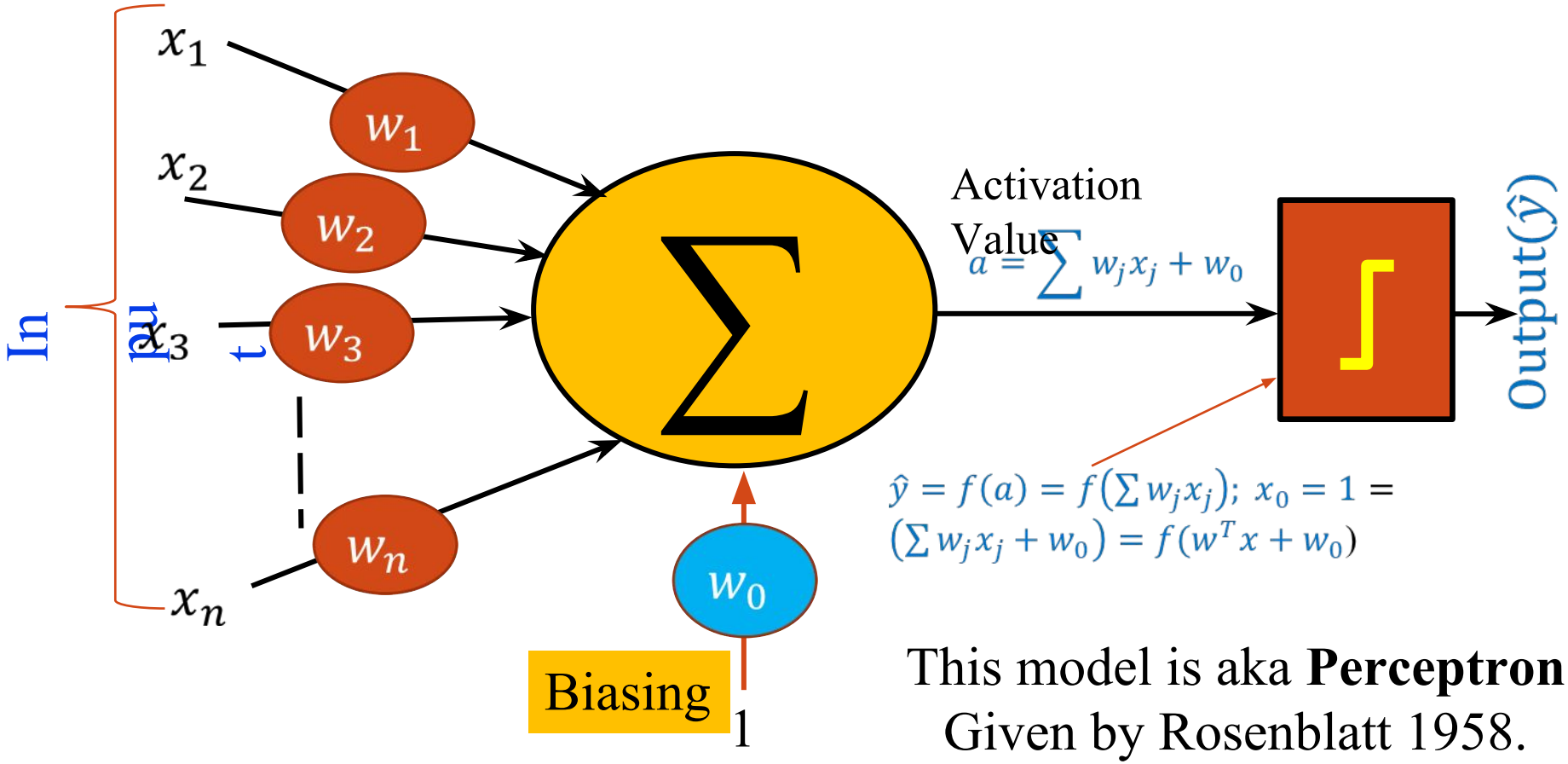
The Artificial Neuron…

- **Activation Functions**



The Artificial Neuron…

- **Activation Functions**



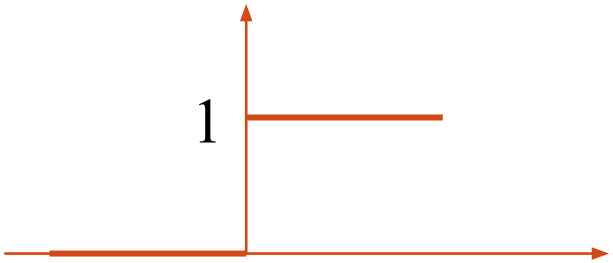
This model is aka **Perceptron**
Given by Rosenblatt 1958.

A Simple Model of a Neuron (Perceptron)…

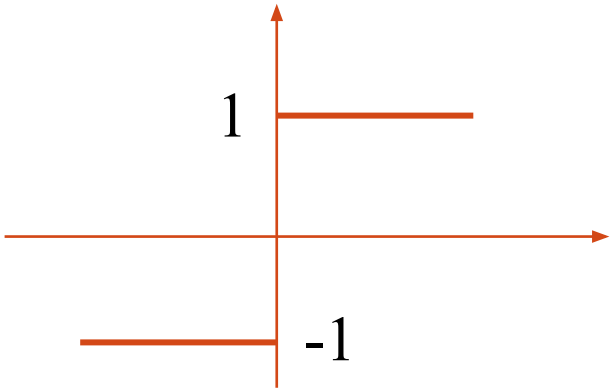
- Each hidden or output neuron has weighted input connections from each of the units in the preceding layer.
- The unit performs a weighted sum of its inputs, and subtracts its threshold value, to give its activation value.
- Activation value is passed through a **activation function** to determine output.

Activation Functions

- Mapping the activation value to the output.
- It can be done using **squashing function**: unipolar and bipolar.



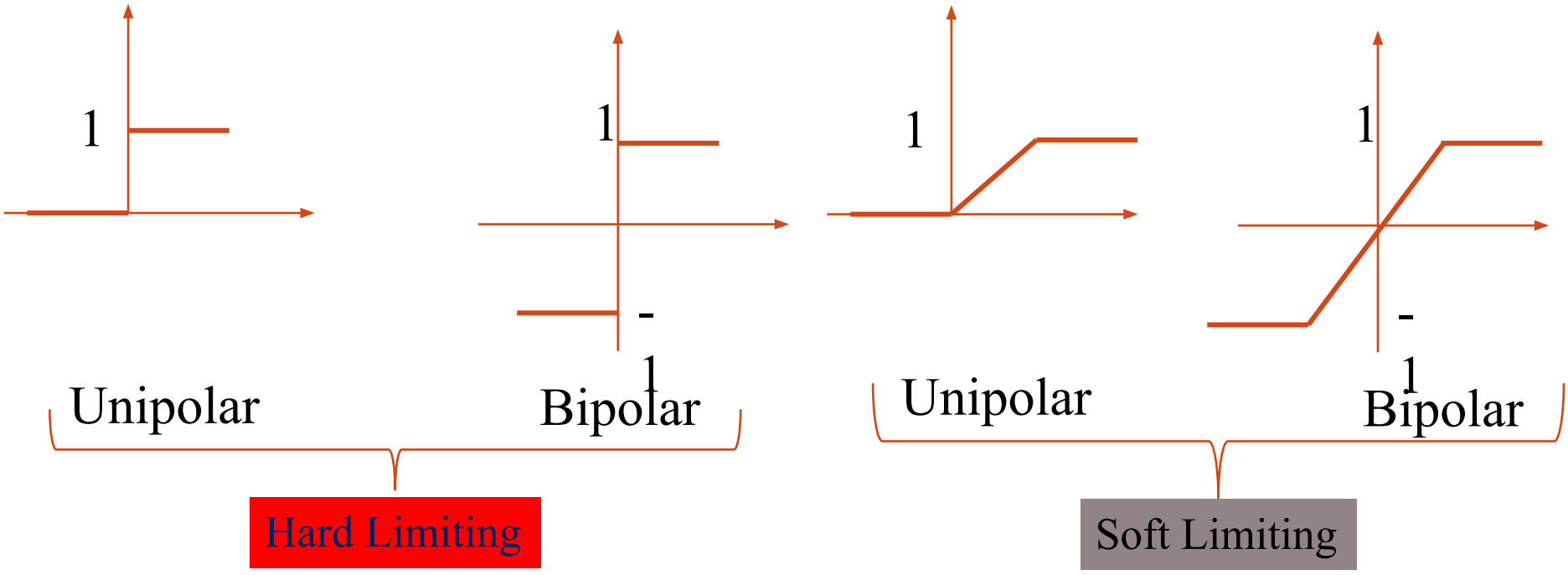
Unipolar



Bipolar

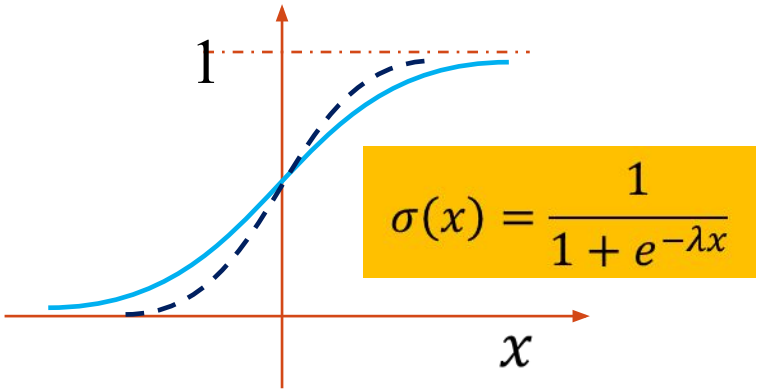
Activation Functions…

- These are also classified as **hard limiting activation function** and **soft limiting activation function.**

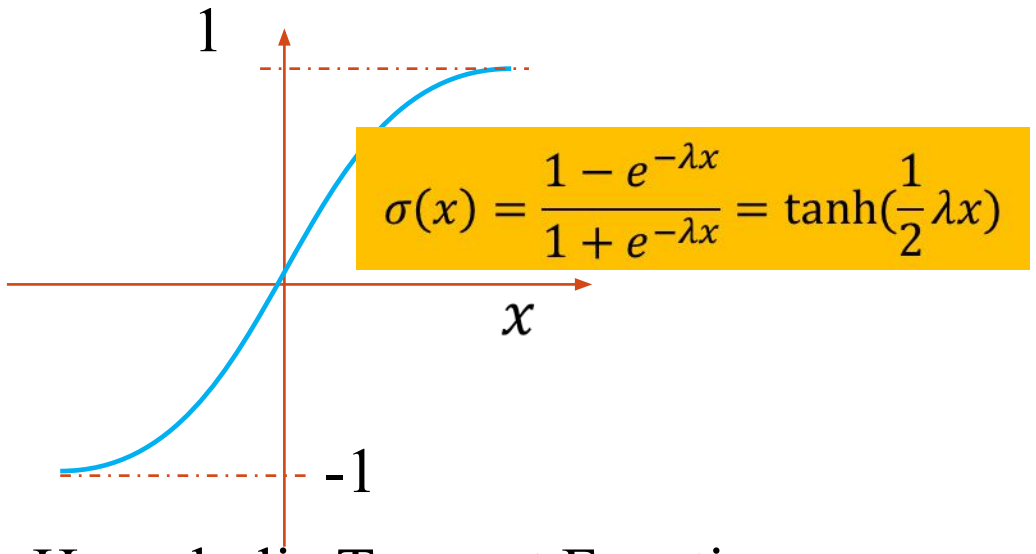


Activation Functions…

- Many training algorithms requires the derivative of activation functions. Hence, activation function must be **differentiable**, E.g. *logistic* and *sigmoid*. Soft limiting function meets the requirement.



Sigmoid Function

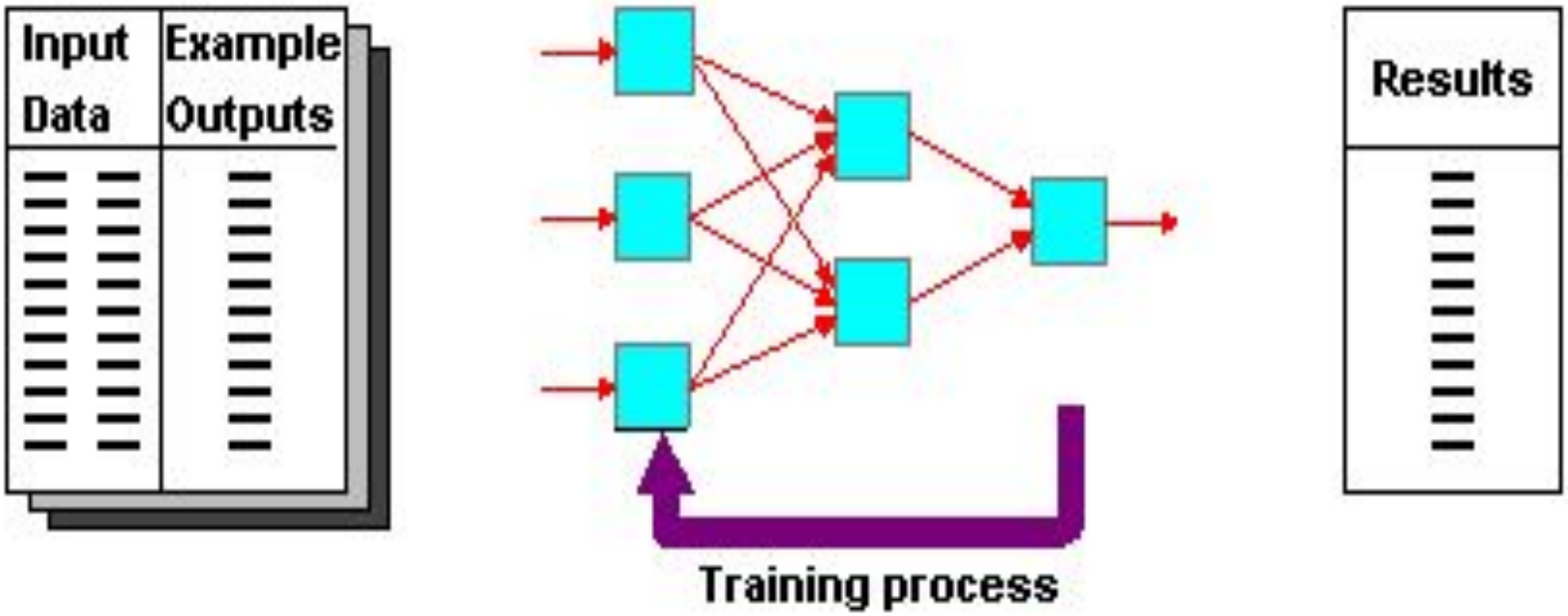


Hyperbolic Tangent Function

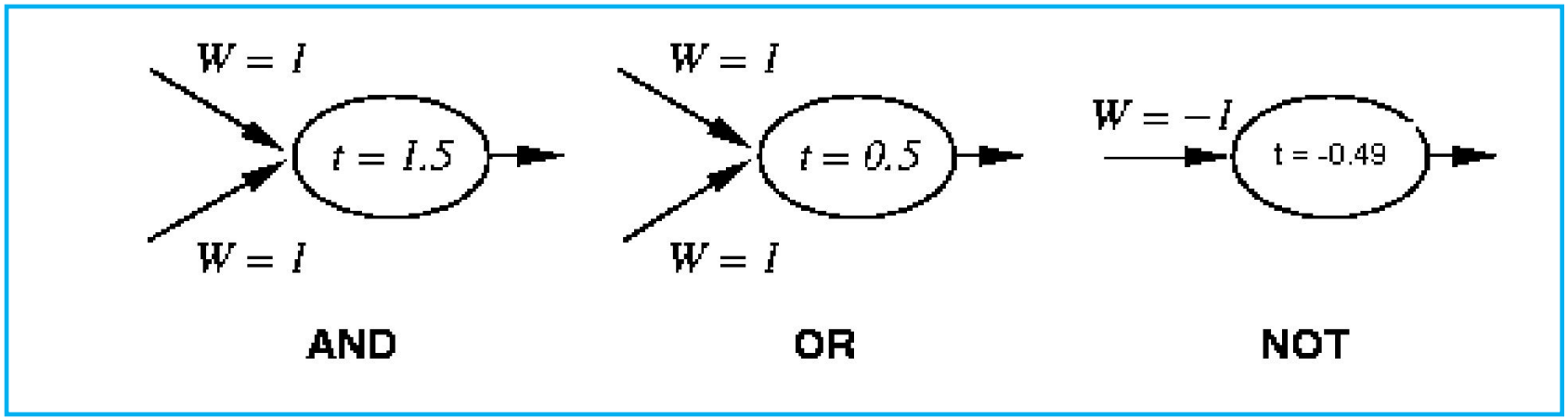
The biological basis of these functions are easily established. Neurons located in different parts of the nervous system have different characteristics. **Ocular motor: sigmoid, Visual Cortex: Gaussian**

Learning of Artificial Neurons

- Supervised Learning
  - ✔ Training and test data sets
  - ✔ Training set; input & target



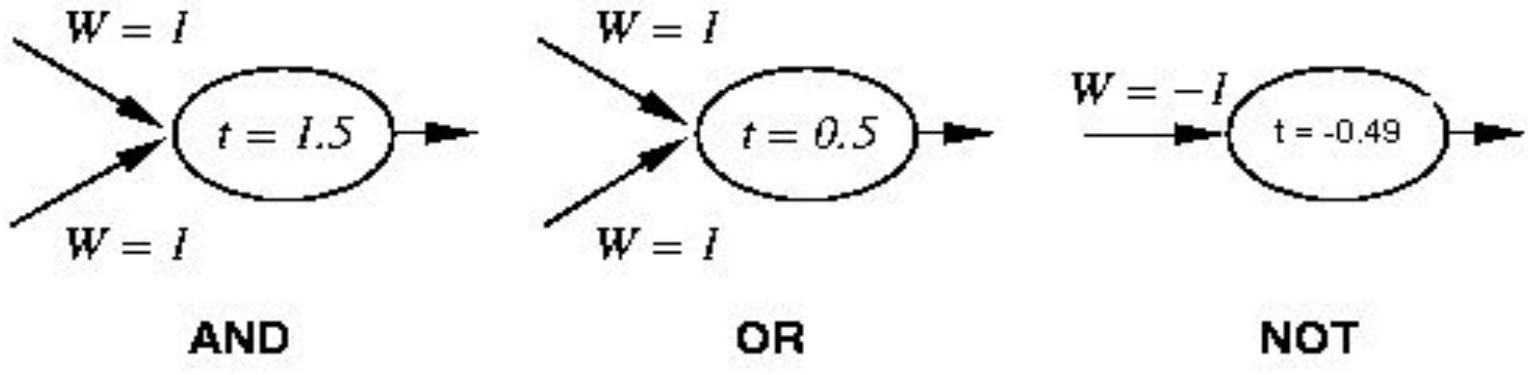
Perceptron Training



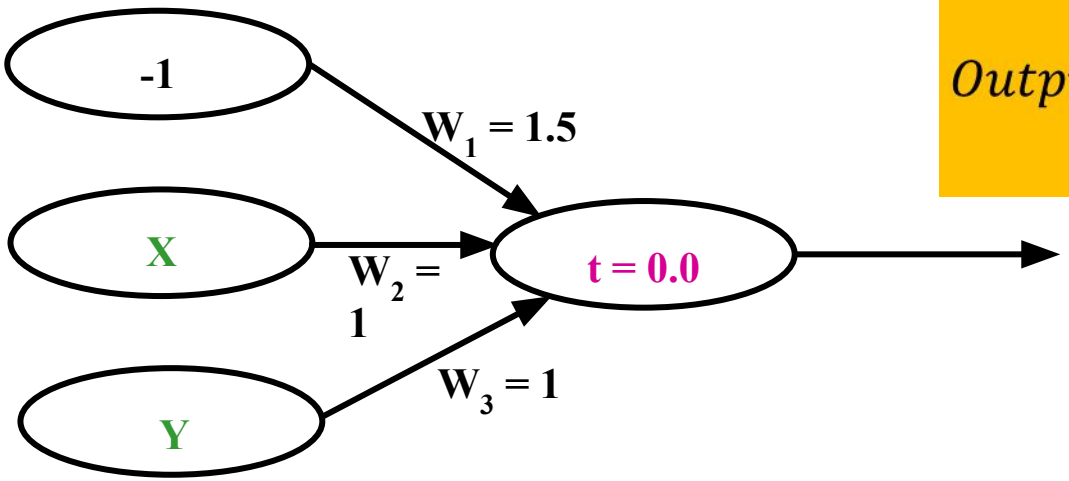
$$Output = \begin{cases} 1 \ if \sum_{i=0} w_i x_i > t \\ 0 \quad otherwise \end{cases}$$

- Linear threshold is used.
- W - weight value
- t - threshold value

Simple network



AND with a Biased input



$$Output = \begin{cases} 1 \ if \ \sum_{i=0} w_i x_i > t \\ 0 \quad otherwise \end{cases}$$

Learning algorithm

While epoch produces an error

Present network with next inputs from epoch

Error = T – O

If Error <> 0 then

$W_j = W_j + LR * I_j * Error$

End If

End While

Learning algorithm

Epoch : Presentation of the entire training set to the neural network. In the case of the AND function an epoch consists of four sets of inputs being presented to the network (i.e. [0,0], [0,1], [1,0], [1,1])

Error: The error value is the amount by which the value output by the network differs from the target value. For example, if we required the network to output 0 and it output a 1, then Error = -1

Learning algorithm

Target Value, T : When we are training a network we not only present it with the input but also with a value that we require the network to produce. For example, if we present the network with [1,1] for the AND function the target value will be 1.

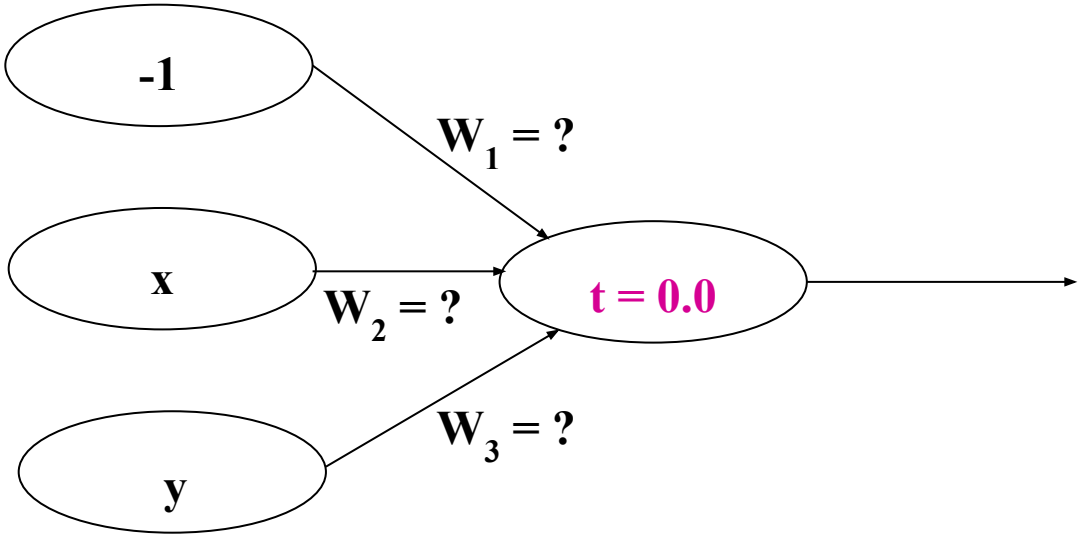
Output , O : The output value from the neuron

Ij : Inputs being presented to the neuron

Wj : Weight from input neuron (I_j) to the output neuron

LR: The learning rate. This dictates how quickly the network converges. It is set by a matter of experimentation. It is typically 0.1

Training Perceptrons

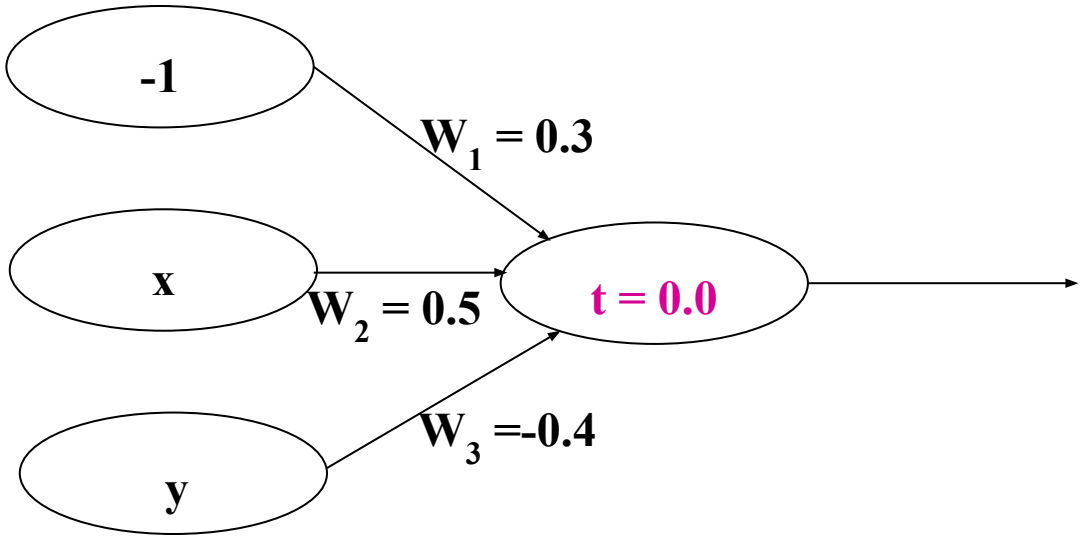


For AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

- What are the weight values?
- Initialize with random weight values

Training Perceptron's



For AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

I_1	I_2	I_3	Summation	Output
-1	0	0	(-1*0.3) + (0*0.5) + (0*-0.4) = -0.3	0
-1	0	1	(-1*0.3) + (0*0.5) + (1*-0.4) = -0.7	0
-1	1	0	(-1*0.3) + (1*0.5) + (0*-0.4) = 0.2	1
-1	1	1	(-1*0.3) + (1*0.5) + (1*-0.4) = -0.2	0

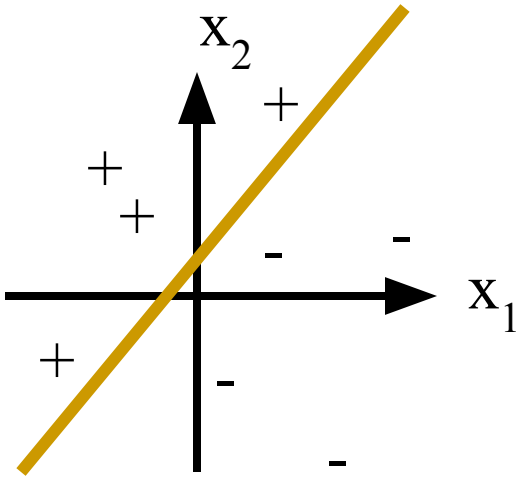
Learning in Neural Networks

- Learn values of weights from I/O pairs
- Start with random weights
- Load training example's input
- Observe computed input
- Modify weights to reduce difference
- Iterate over all training examples
- Terminate when weights stop changing OR when error is very small

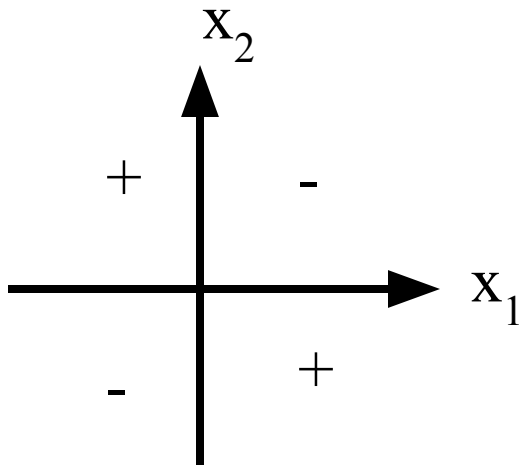
Decision Boundaries

- In simple cases, divide feature space by drawing a hyperplane across it.
- Known as a **decision boundary**.
- **Discriminant function:** returns different values on opposite sides. (straight line)
- Problems which can be thus classified are **linearly separable**.

Decision Surface of a Perceptron



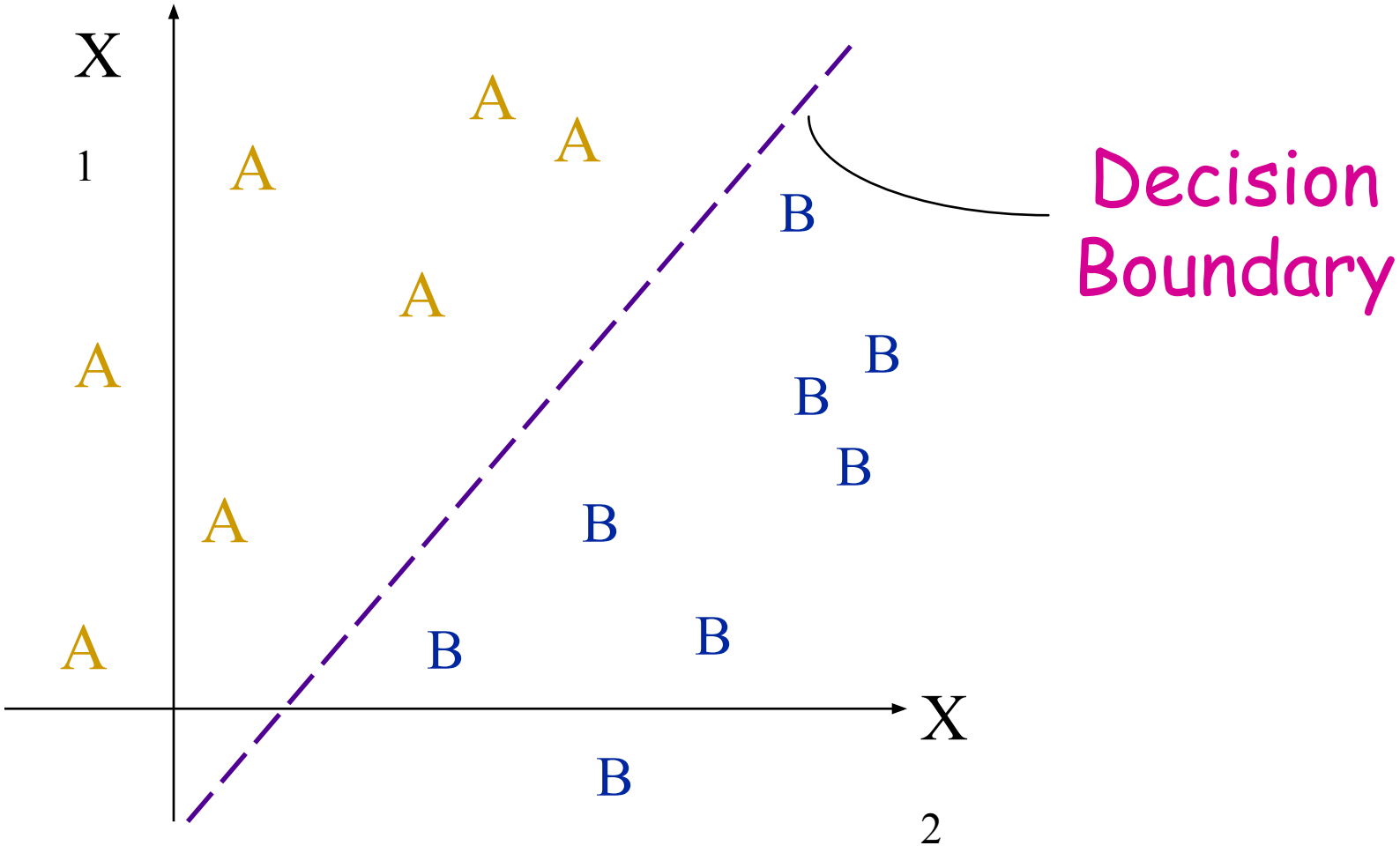
Linearly separable



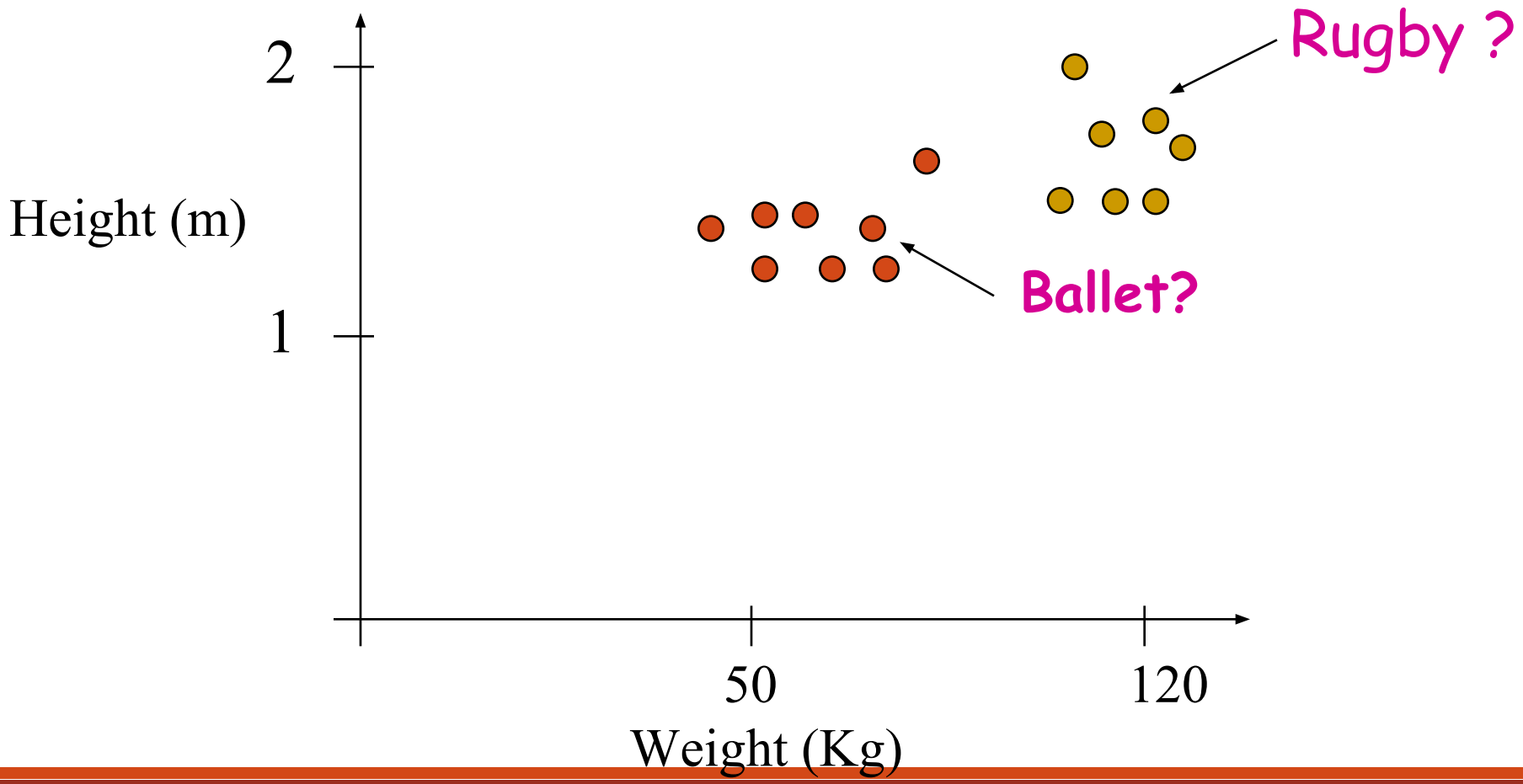
Non-Linearly separable

- Perceptron is able to represent some useful functions
- AND(x_1,x_2) choose weights w_0=-1.5, w_1=1, w_2=1
- But functions that are not linearly separable (e.g. XOR) are not representable

Linear Separability



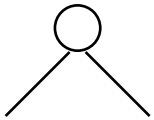
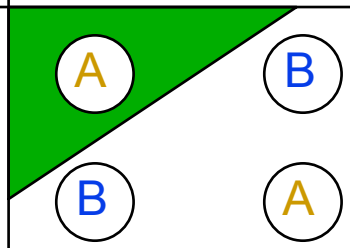
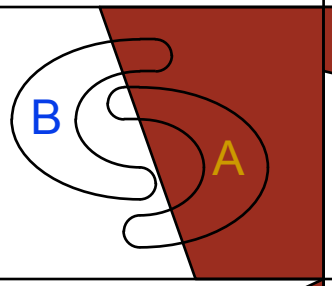
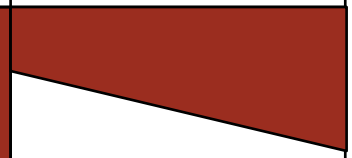
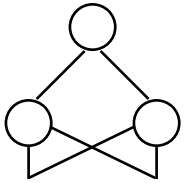
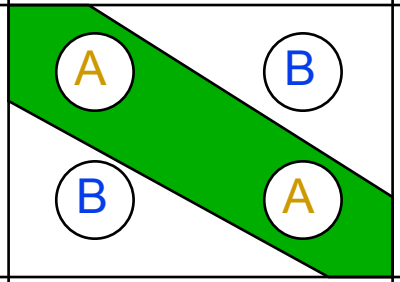
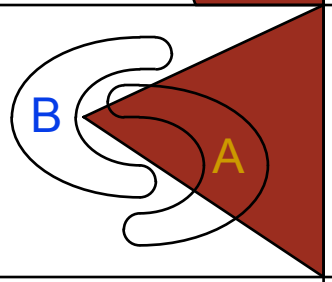
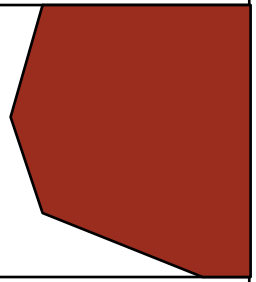
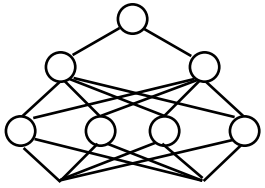
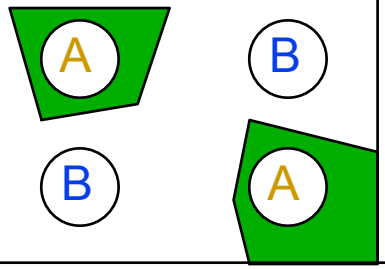
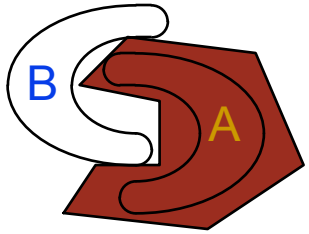
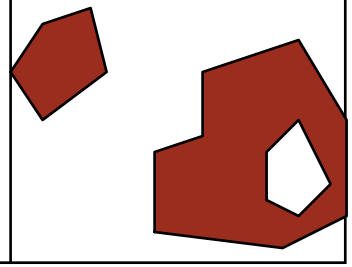
Rugby players & Ballet dancers



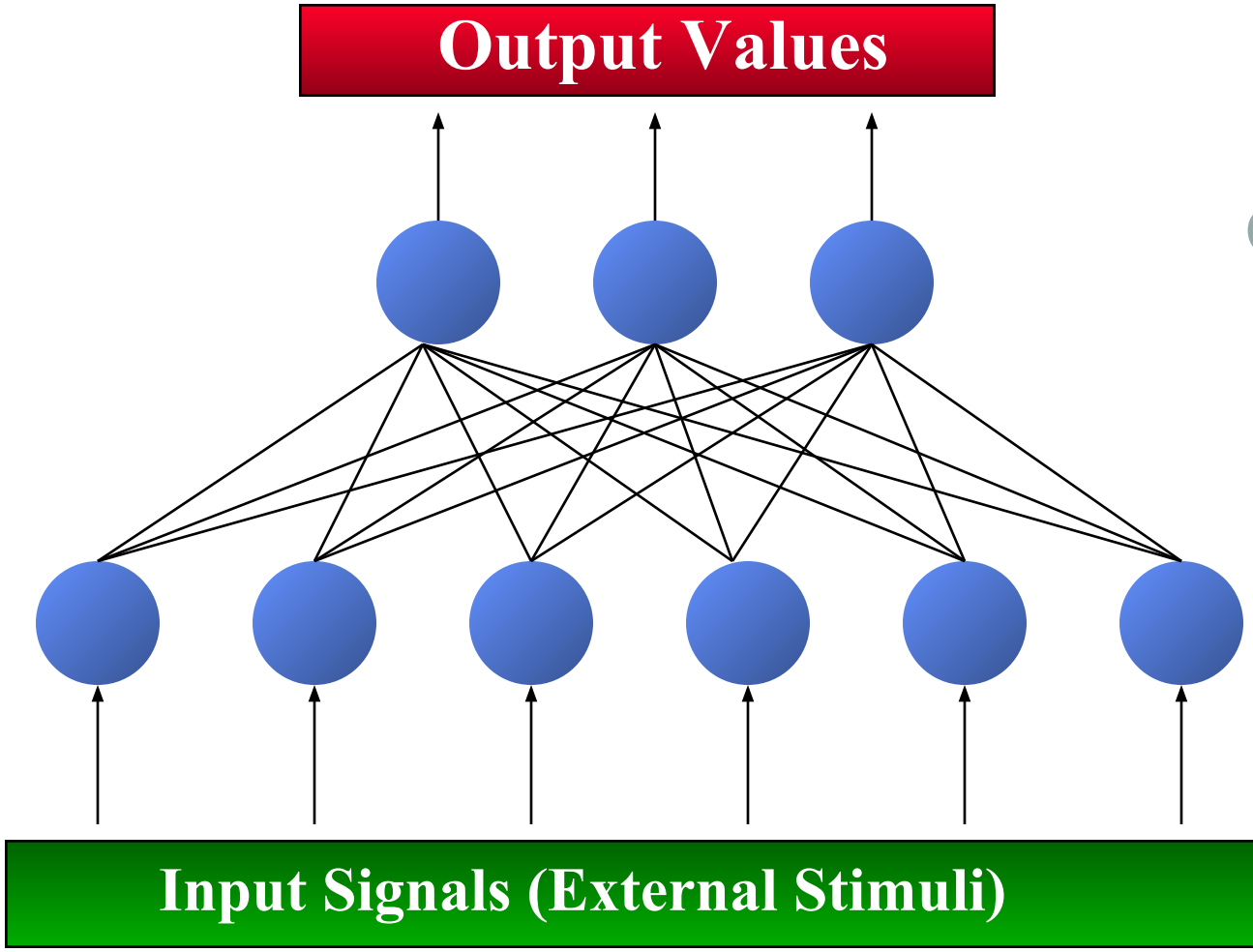
Hyperplane Partitions

- A single Perceptron (i.e. output unit) with connections from each input can perform, and learn, a linear separation.
- Perceptrons have a step function activation.
- An extra layer models a convex hull
  - Perceptron models, but can't learn
  - Sigmoid function learning of convex hulls
  - Two layers add convex hulls together
  - Sufficient to classify anything "sane".
- In theory, further layers add nothing
- In practice, extra layers may be better

Different Non-Linearly Separable Problems

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

Multilayer Perceptron (MLP)



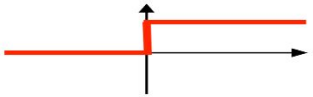
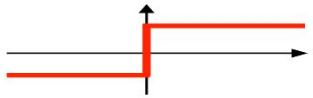
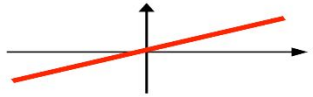
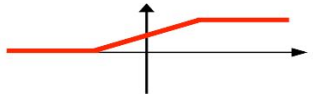
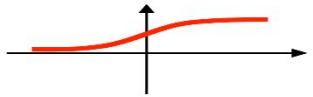
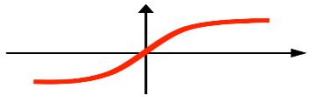
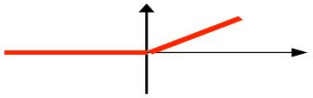
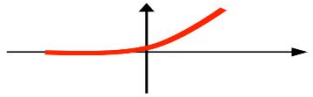
Types of Layers

- The input layer.
  - Introduces input values into the network.
  - No activation function or other processing.
- The hidden layer(s).
  - Perform classification of features
  - Two hidden layers are sufficient to solve any problem
  - Features imply more layers may be better
- The output layer.
  - Functionally just like the hidden layers
  - Outputs are passed on to the world outside the neural network.

Activation functions

- Transforms neuron's input into output.
- Features of activation functions:
  - A squashing effect is required
    - Prevents accelerating growth of activation levels through the network.
  - Simple and easy to calculate

Activation functions…

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016 (http://sebastianraschka.com)

Standard activation functions

- The hard-limiting threshold function
  - Corresponds to the biological paradigm
    - either fires or not
- Sigmoid functions ('S'-shaped curves)
  - The logistic function

$$\varphi(x) = \frac{1}{1 + e^{-ax}}$$

  - The hyperbolic tangent (symmetrical)
  - Both functions have a simple differential
  - Only the shape is important

Training Algorithms

- Adjust neural network weights to map inputs to outputs.
- Use a set of sample patterns where the desired output (given the inputs presented) is known.
- The purpose is to learn to generalize
  - Recognize features which are common to good and bad exemplars

Back-Propagation

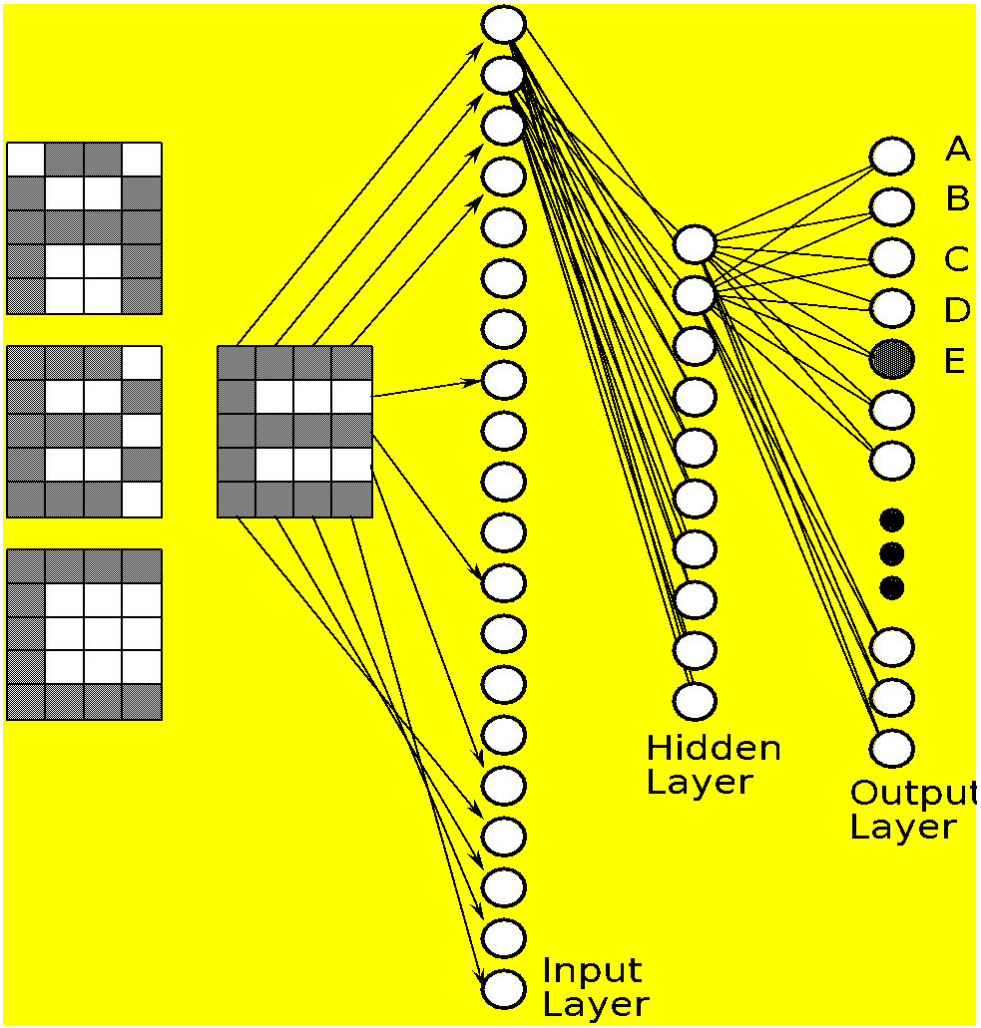
- A training procedure which allows multi-layer feedforward Neural Networks to be trained;
- Can theoretically perform "any" input-output mapping;
- Can learn to solve linearly inseparable problems.

Applications

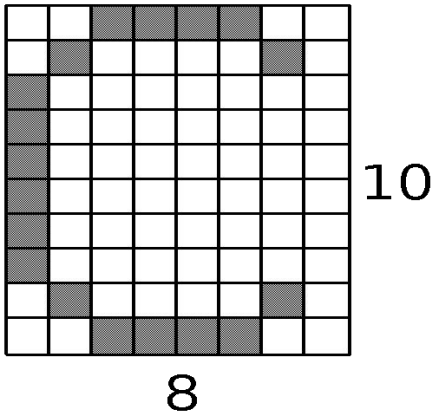
- The properties of neural networks define where they are useful.
  - Can learn complex mappings from inputs to outputs, based solely on samples
  - Difficult to analyse: firm predictions about neural network behaviour difficult;
    - Unsuitable for safety-critical applications.
  - Require limited understanding from trainer, who can be guided by heuristics.

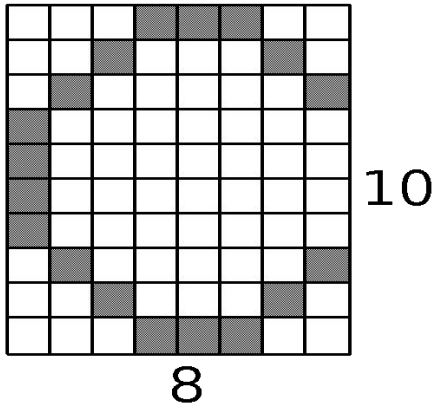
Neural network for OCR

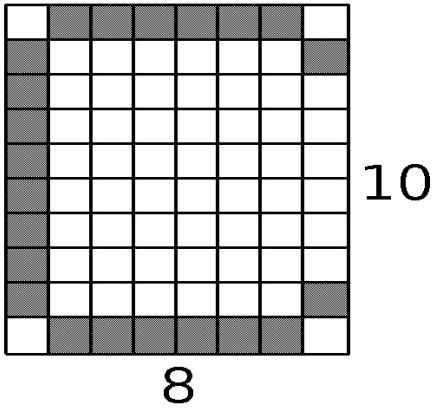
- Feedforward network
- Trained using Back-propagation



OCR for 8x10 characters

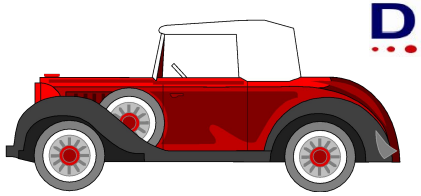






- NN are able to generalise
- learning involves generating a partitioning of the input space
- for single layer network input space must be linearly separable
- what is the dimension of this input space?
- how many points in the input space?
- this network is binary(uses binary values)
- networks may also be continuous

Engine management

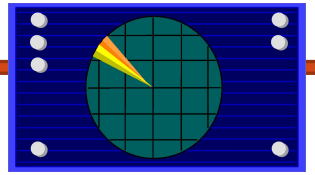


- The behaviour of a car engine is influenced by a large number of parameters
  - temperature at various points
  - fuel/air mixture
  - lubricant viscosity.
- Major companies have used neural networks to dynamically tune an engine depending on current settings.

Signature recognition

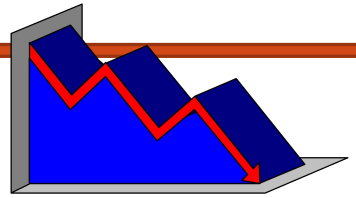
- Each person's signature is different.
- There are structural similarities which are difficult to quantify.
- One company has manufactured a machine which recognizes signatures to within a high level of accuracy.
  - Considers speed in addition to gross shape.
  - Makes forgery even more difficult.

Sonar target recognition



- Distinguish mines from rocks on sea-bed
- The neural network is provided with a large number of parameters which are extracted from the sonar signal.
- The training set consists of sets of signals from rocks and mines.

Stock market prediction



- "Technical trading" refers to trading based solely on known statistical parameters; e.g. previous price
- Neural networks have been used to attempt to predict changes in prices.
- Difficult to assess success since companies using these techniques are reluctant to disclose information.

Mortgage assessment

- Assess risk of lending to an individual.
- Difficult to decide on marginal cases.
- Neural networks have been trained to make decisions, based upon the opinions of expert underwriters.
- Neural network produced a 12% reduction in delinquencies compared with human experts.

Neural Network Problems

- Many Parameters to be set
- Overfitting
- long training times
- ...

Parameter setting

- Number of layers
- Number of neurons
  - too many neurons, require more training time
- Learning rate
  - from experience, value should be small ~0.1
- Momentum term
- ..

Hyperparameters

- In statistics, hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed.
- In any ML algorithm, these parameters need to be initialized before training a model.
- ***Model parameters*** are the properties of training data that will learn on its own during training by the classifier or model.

✔ Weights and Biases.

✔ Split points in Decision Tree

Hyperparameters…

- ***Model Hyperparameters*** are the properties that govern the entire training process. The below are the variables usually configure before training a model.

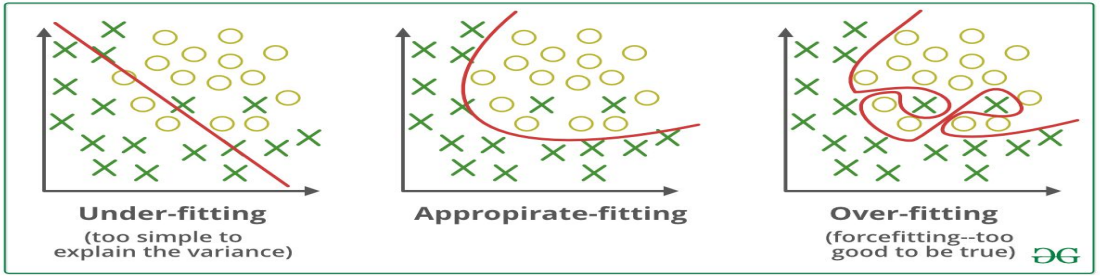
✔ Learning Rate
✔ Number of Epochs
✔ Hidden Layers
✔ Hidden Units
✔ Activations Functions

Hyperparameters are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model is being trained.

Hyperparameters…

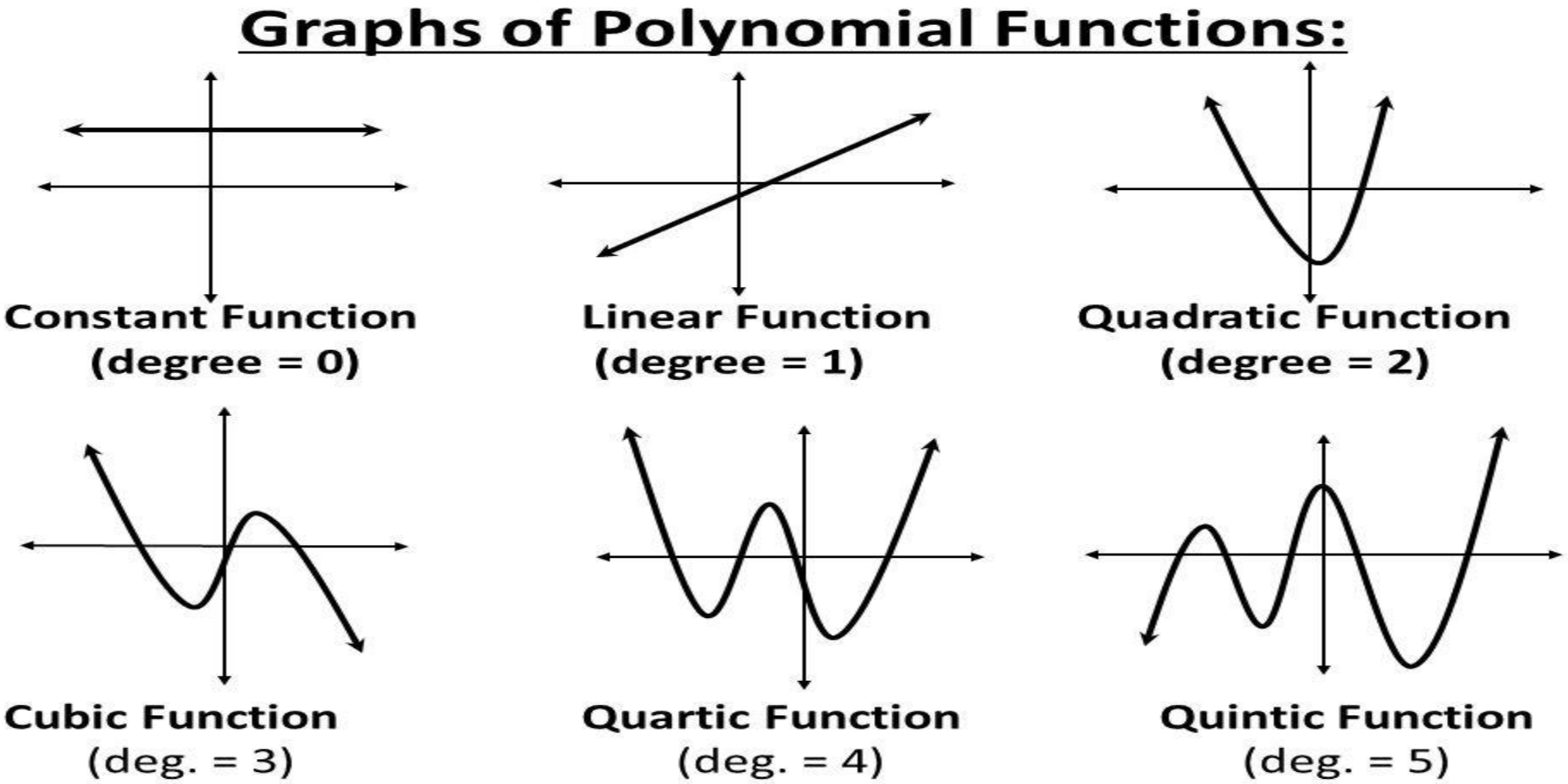
- **Choosing good hyperparameters gives two benefits:**
  - ✔ Efficiently search the space of possible hyperparameters
  - ✔ Easy to manage a large set of experiments for hyperparameter tuning.
- **Hyperparameters Optimisation Techniques**
  - ✔ *The process of finding most optimal hyperparameters in machine learning is called hyperparameter optimization.*
- **Common algorithms used for optimization is**
  - ✔ Grid Search
  - ✔ Random Search
  - ✔ Bayesian Optimisation

Overfittings

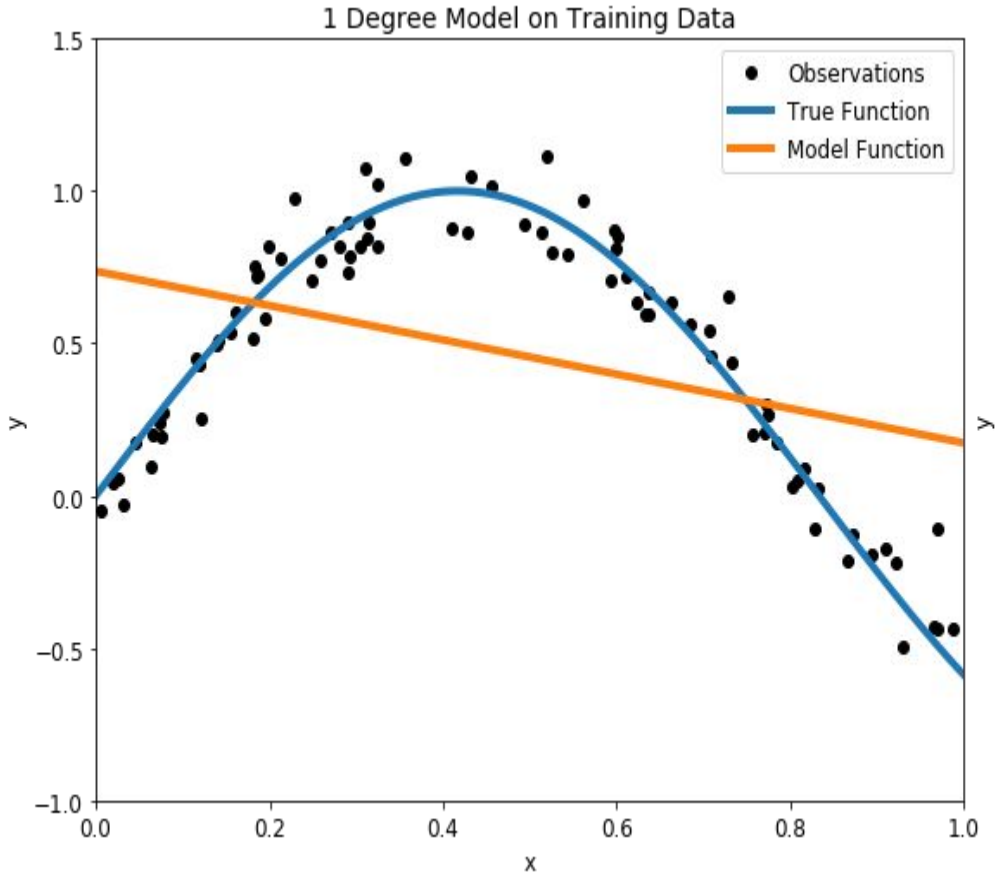


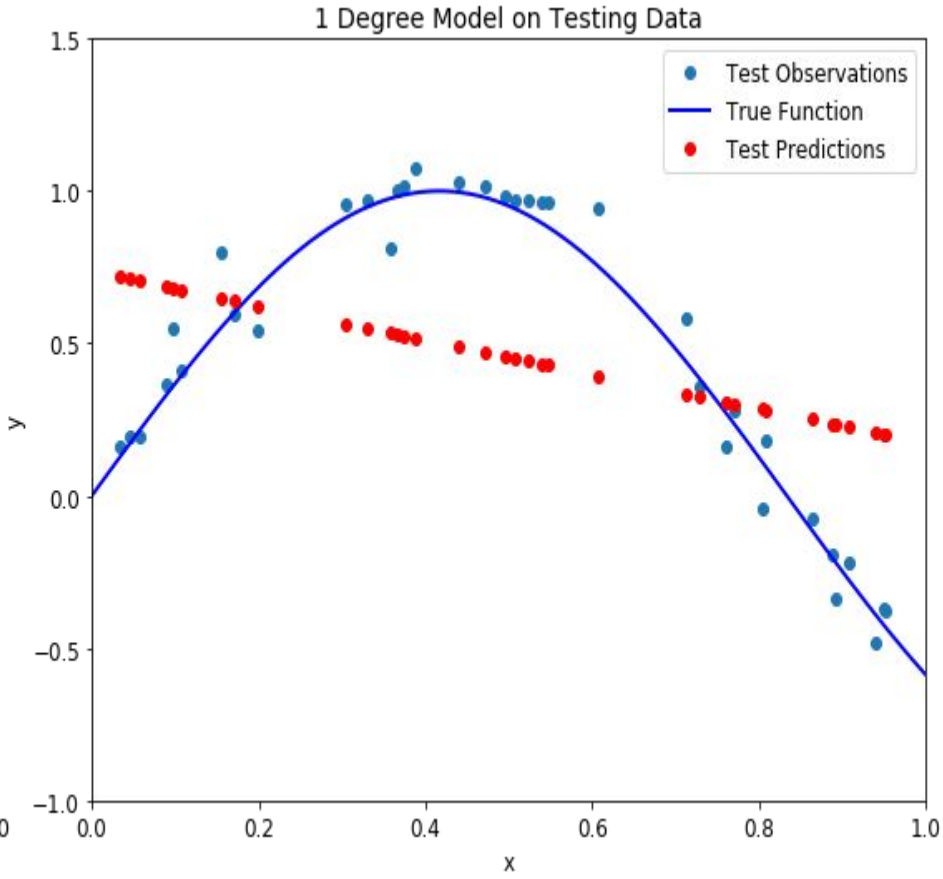
- The problem of Overfitting vs Underfitting, appears when we talk about the polynomial degree.
- The degree represents how much flexibility is in the model, with a higher power allowing the model freedom to hit as many data points as possible.
- An underfit model will be less flexible and cannot account for the data.
- May have poor generalisation ability.
- Cross-validation with some patterns
  - Typically 30% of training patterns
  - Validation set error is checked each epoch
  - Stop training if validation error goes up.

Overfittings…



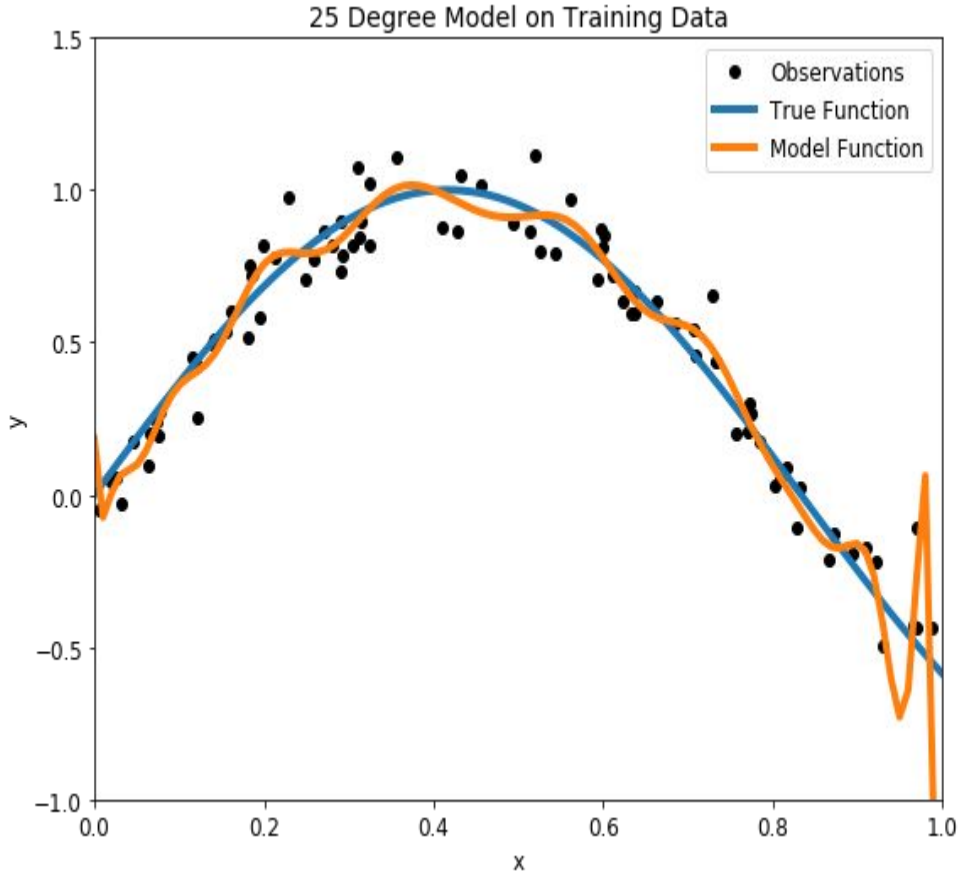
Overfittings…_Underfit

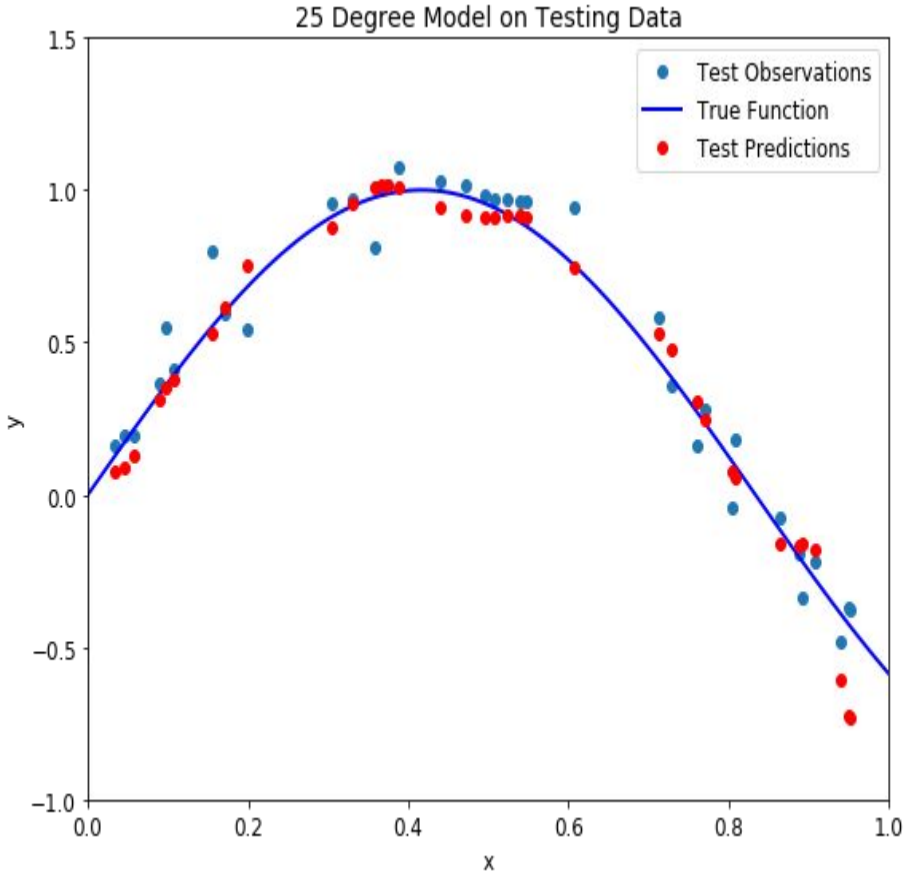




Underfit 1 degree polynomial model on training (left) and testing (right) datasets

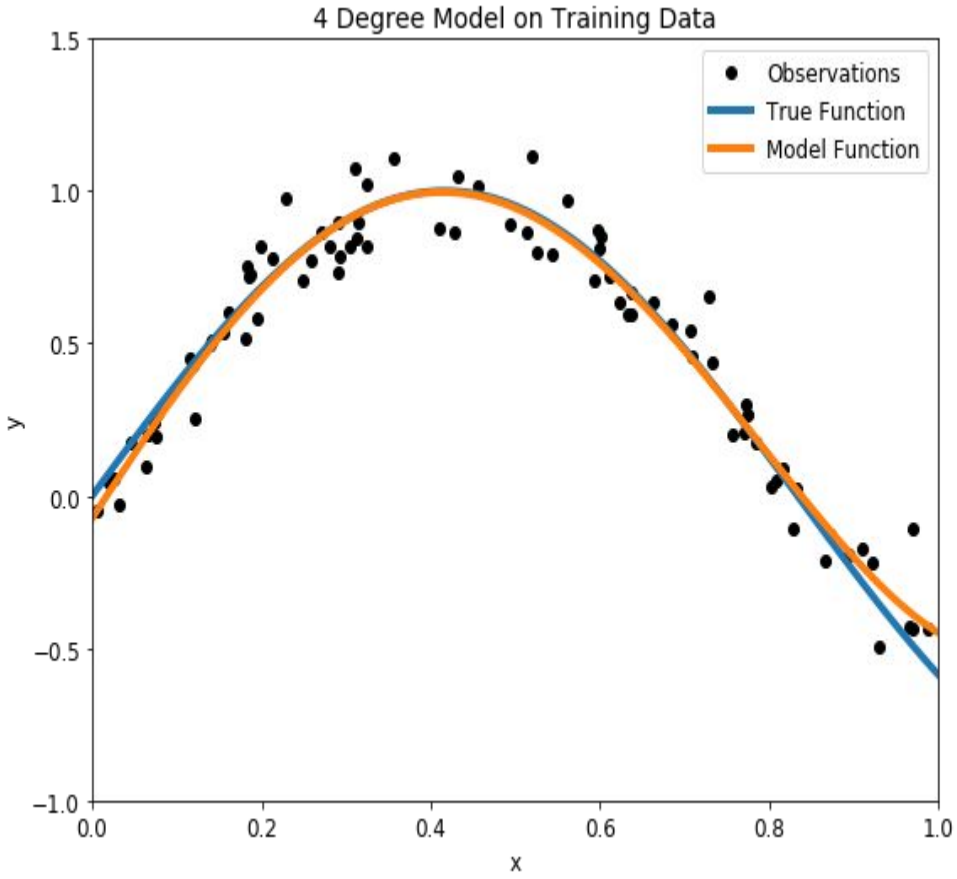
Overfittings…_Overfit

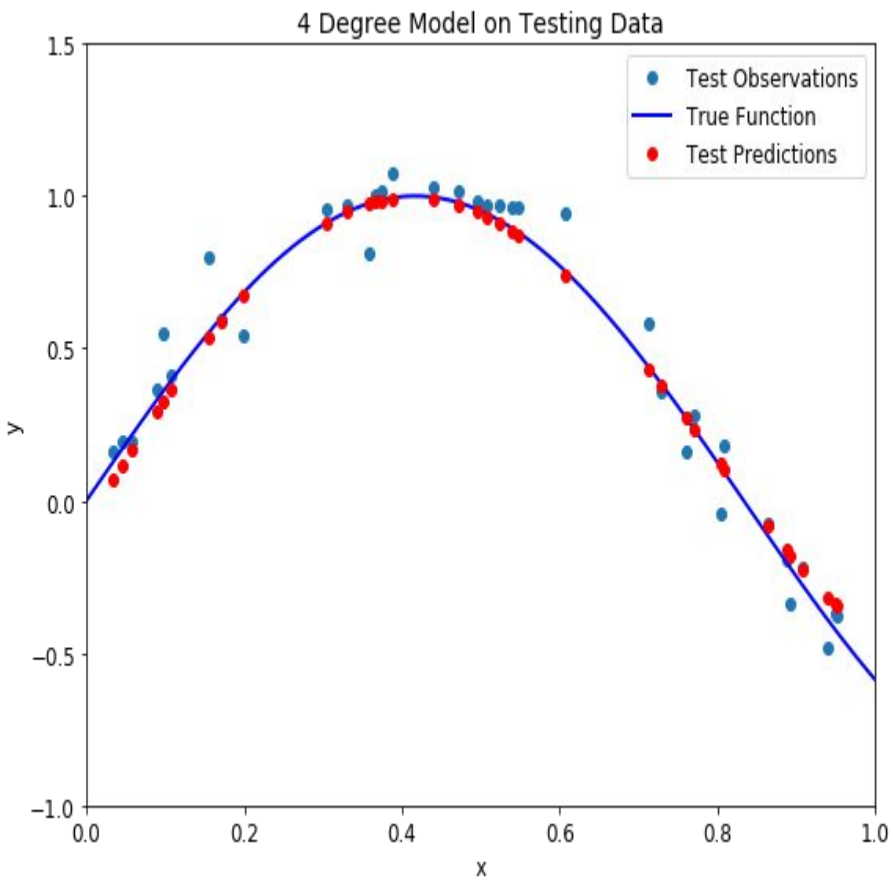




Overfit 25 degree polynomial model on training (left) and testing (right) datasets

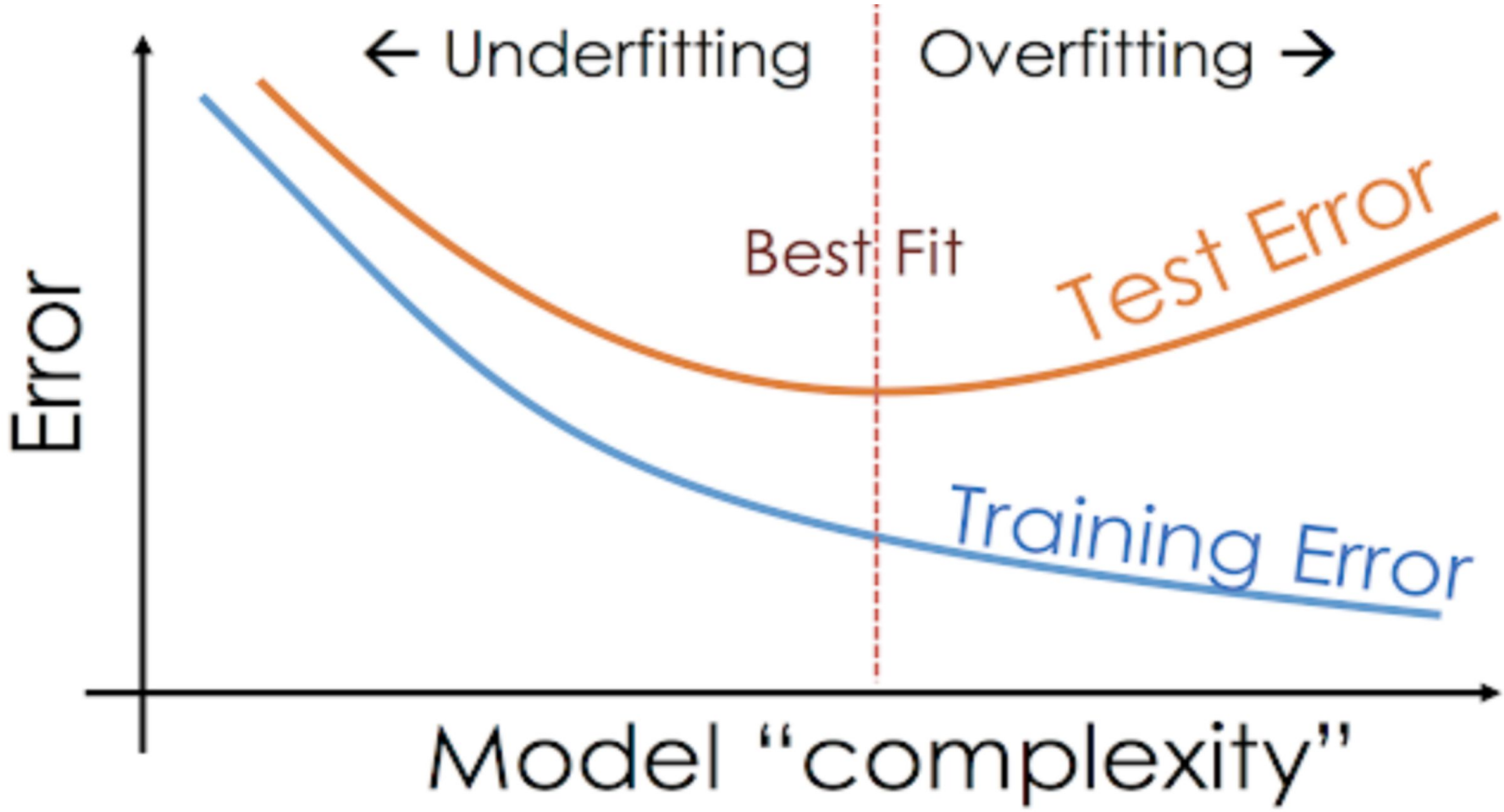
Overfittings…_Balanced Fit





Balanced Four degree polynomial model on training (left) and testing (right) datasets

Underfit_Overfit_Bestfit



Training Time

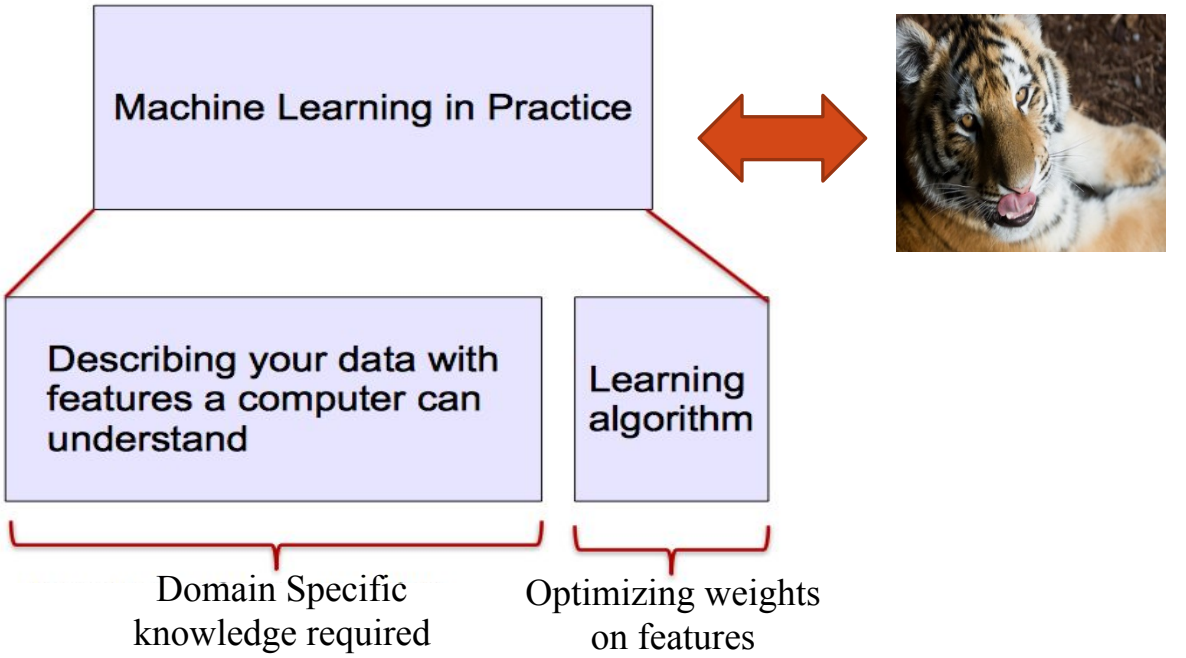
- How many epochs of training?
  - Stop if the error fails to improve (has reached a minimum)
  - Stop if the rate of improvement drops below a certain level
  - Stop if the error reaches an acceptable level
  - Stop when a certain number of epochs have passed

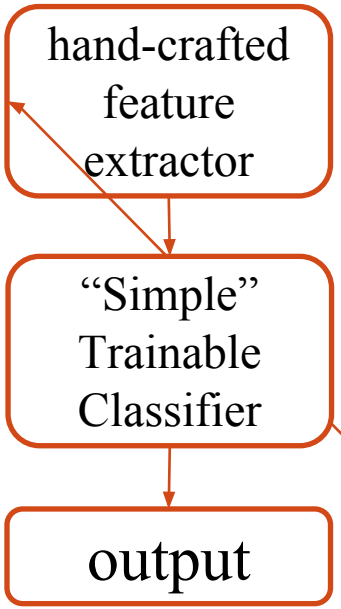
ML vs DL

- Can computers take decisions just like humans?
- Can computers help humans in doing their tasks of daily living?
- Can we build a smart eco-system where users get feedback and systems can update their actions?
- Can we develop technology to learn from human behaviour?

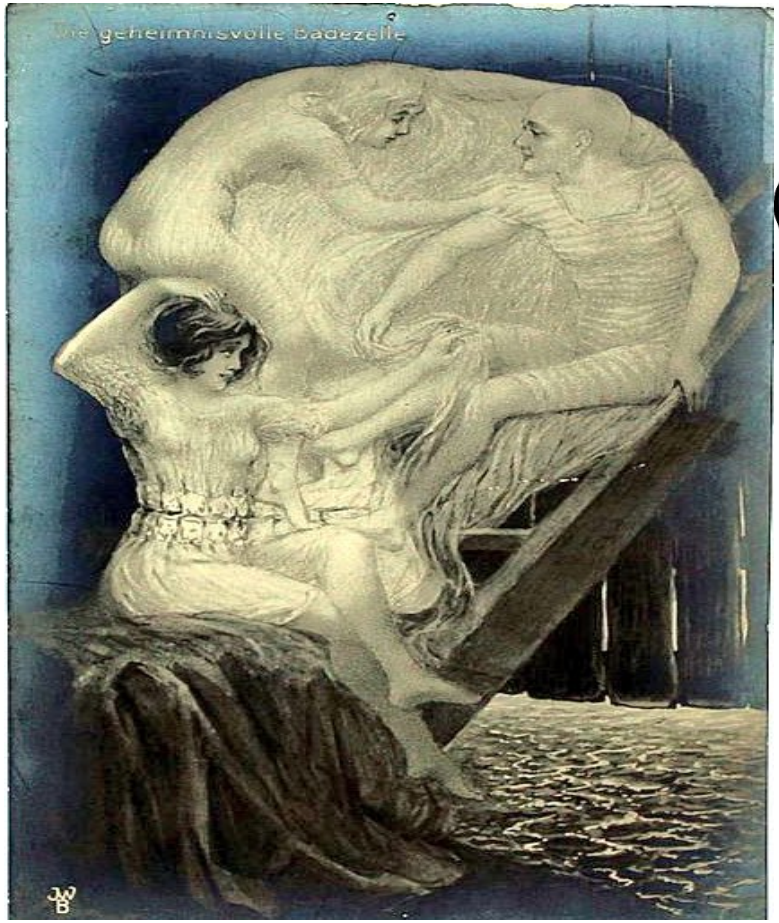
ML Operations

- Most of the **ML approaches** work well because of **human-designed representations** and ***input features***.
- ML becomes just **optimizing weights** to make a final prediction best.





Features Extraction



Really hard to understand

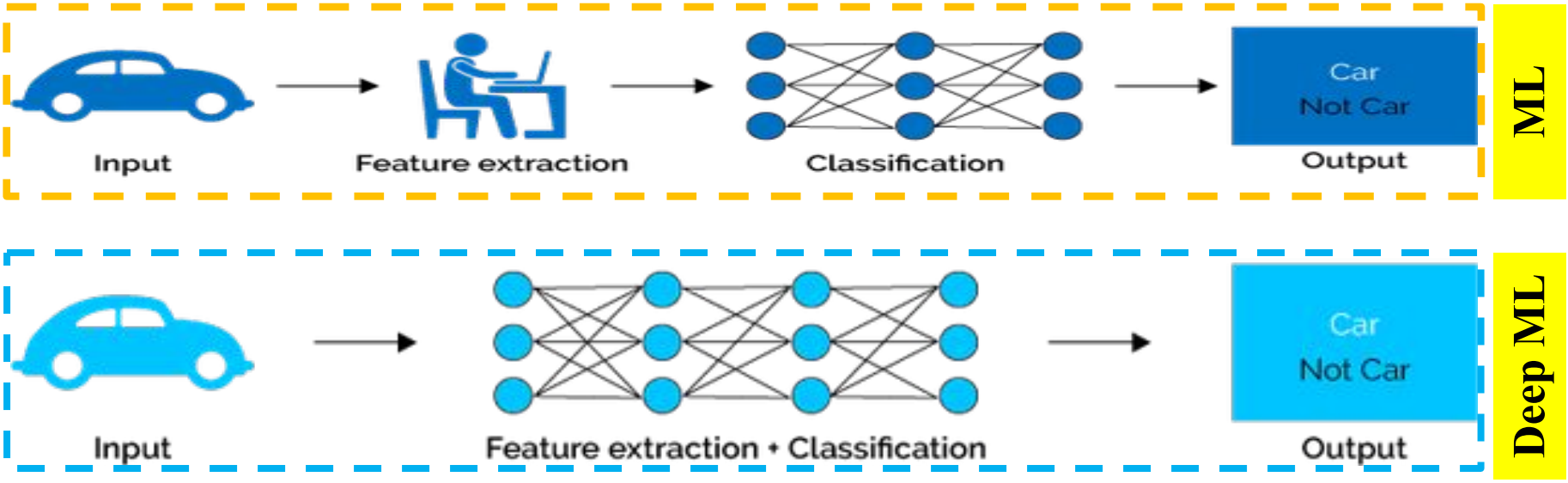
All is Vanity**,** by C. Allan Gilbert, 1873-1929 American Illustrator

Limitations of ML

- It is **very tedious** and **costly** to develop hand-crafted features
- The hand-crafted features are usually highly **dependents on one application**, and cannot be transferred easily to other applications.
- Typically, hand-crafted features are extracted from images for further processing tasks.
- These features are then passed to a ML algorithm to learn specific models. These features are generally difficult to design and are **poorly adaptable** from one data set to the other.

ML v.s. Deep Learning

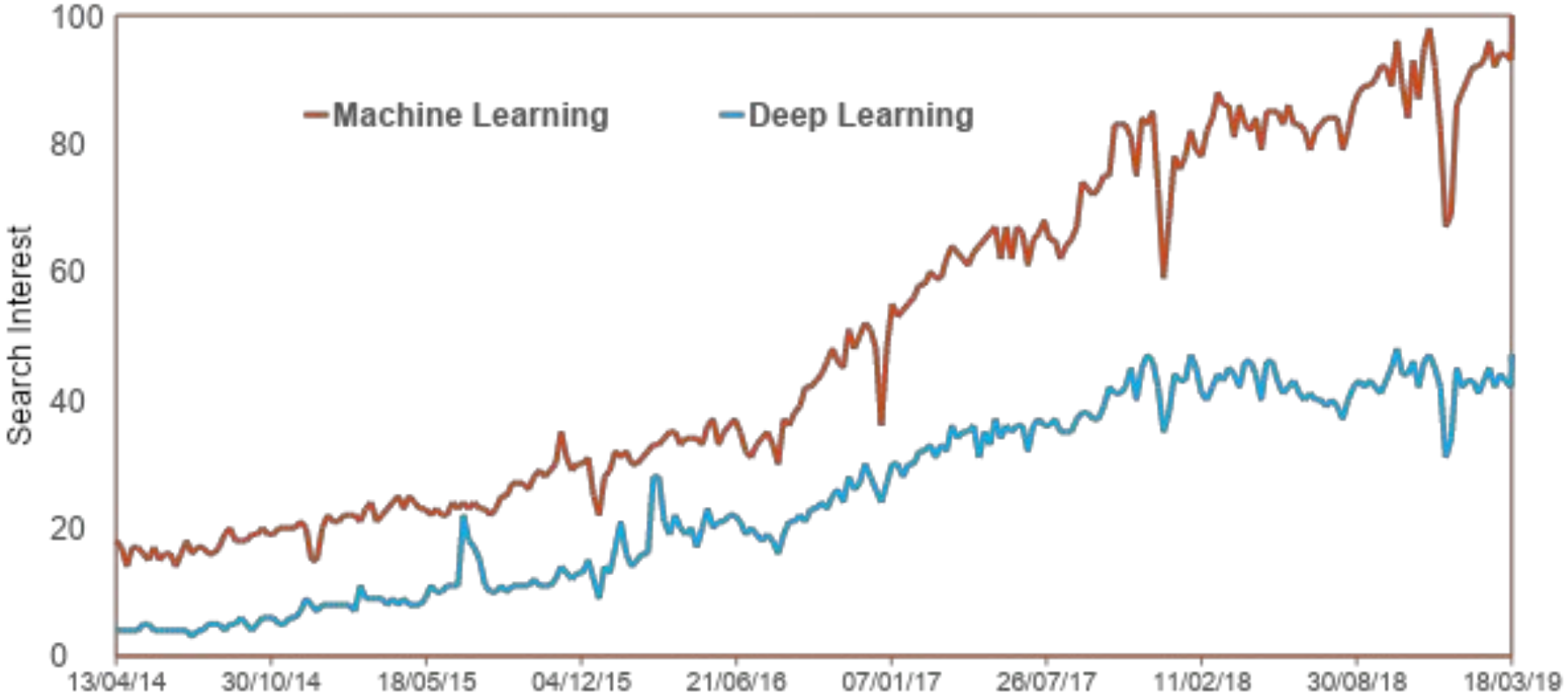
- A machine learning subfield of learning **representations** of data exceptional effective at **learning patterns**.
- Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**
- If you provide the system **tons of information**, it begins to understand it and respond in useful ways.



Why Deep Learning?

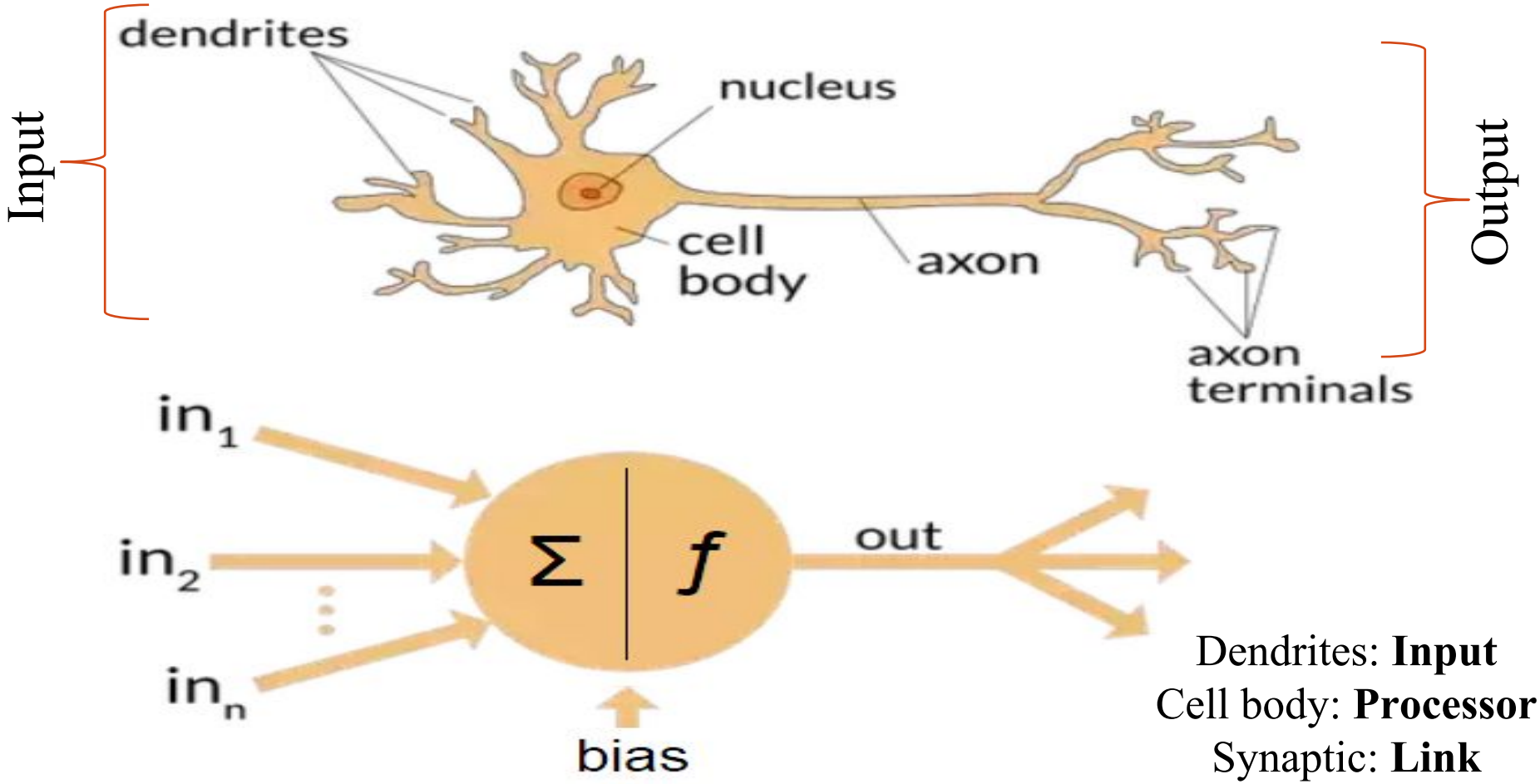
- **Manually designed** or **Hand Crafted** features are often over-specified, incomplete and take a long time to design and validate.
- **Learned Features** are **easy to adapt, fast** to learn
- **Deep learning** provides a **very flexible, universal**, learnable framework for representing world, visual and linguistic information.
- Can learn both unsupervised and supervised.
- Utilize large amounts of training data

ML vs Deep Learning Trends



Data Collected from Google Trends!

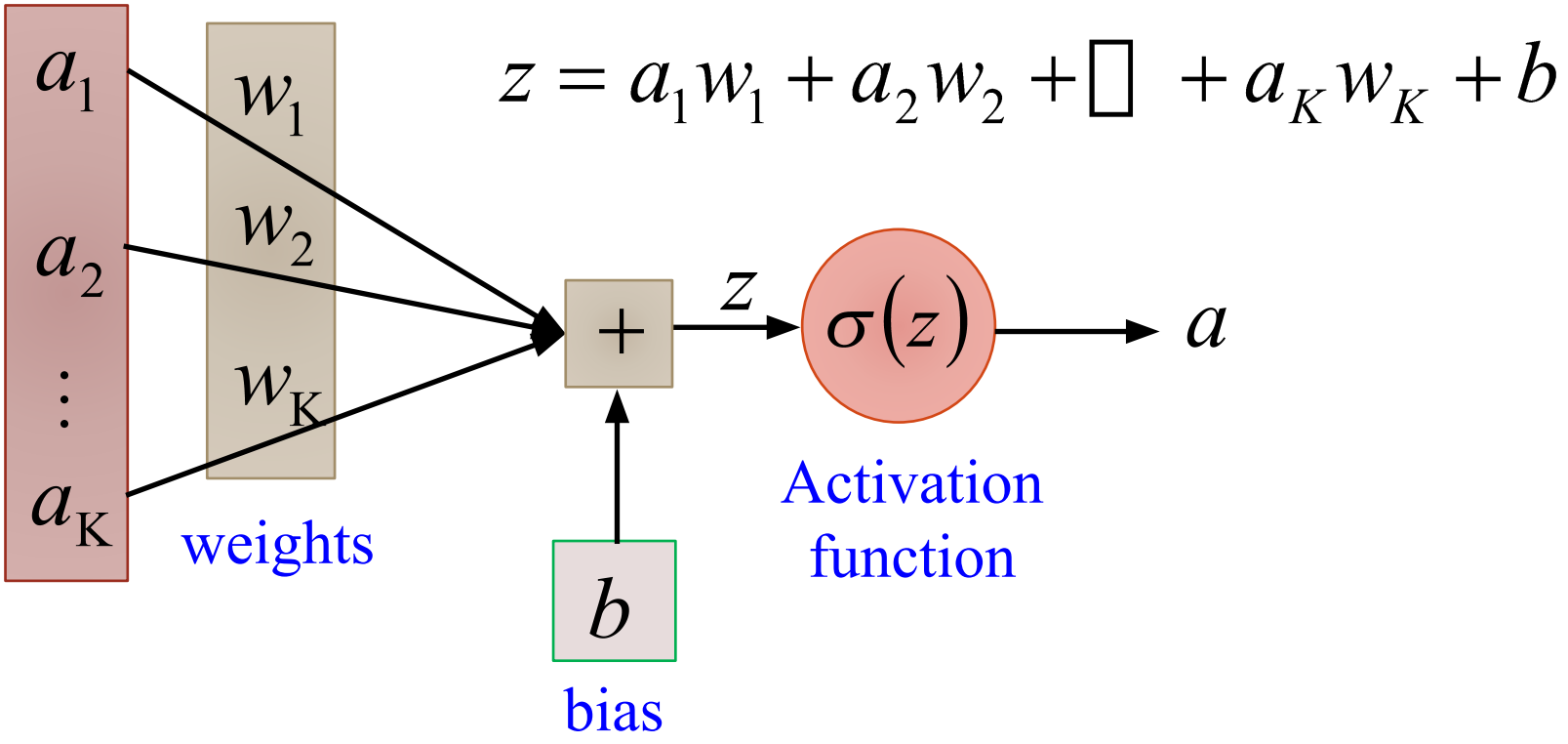
Brain Processing



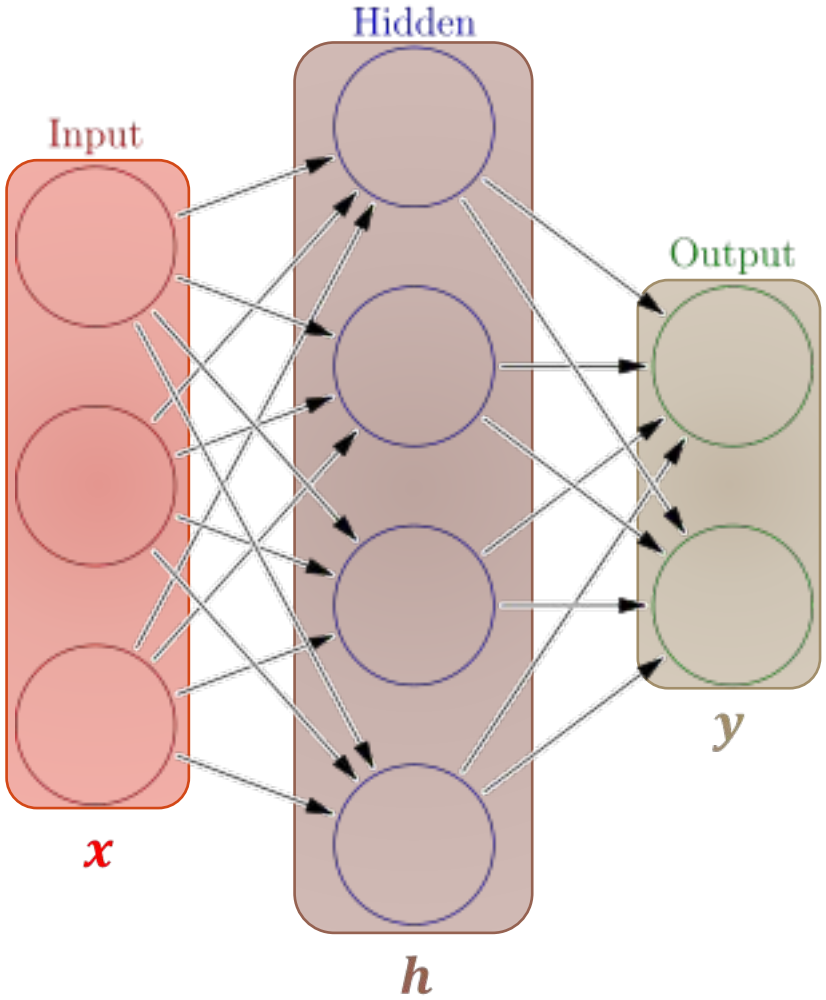
Dendrites: **Input**
Cell body: **Processor**
Synaptic: **Link**
Axon: **Output**

Neuron

$$f: R^K \rightarrow R$$



Neural Network



Weights

$$h = \sigma(W_1 x + b_1)$$

$$y = \sigma(W_2 h + b_2)$$

Activation functions

How do we train?

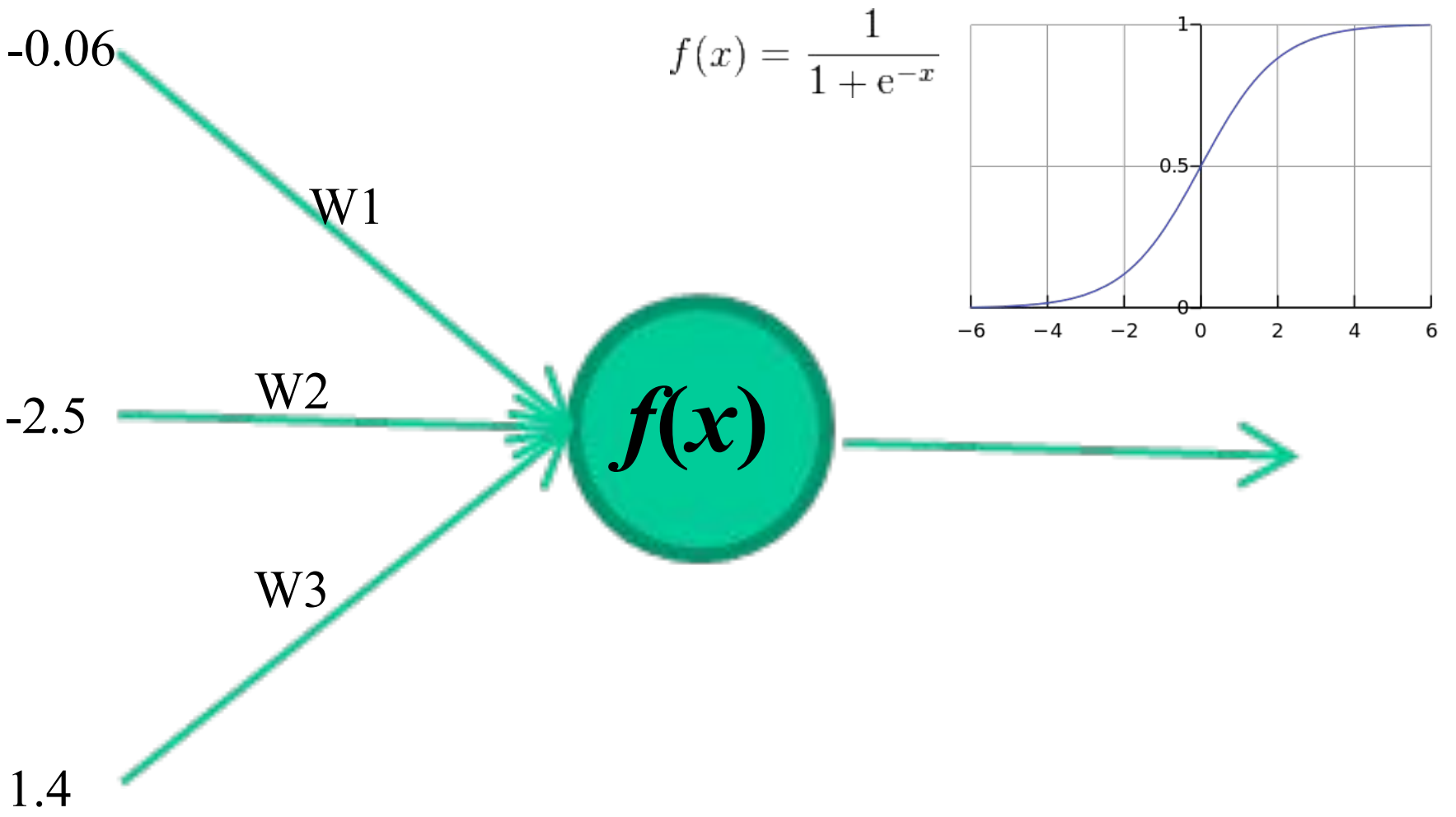
4 + 2 = 6 neurons (not counting inputs)

[3 x 4] + [4 x 2] = 20 weights

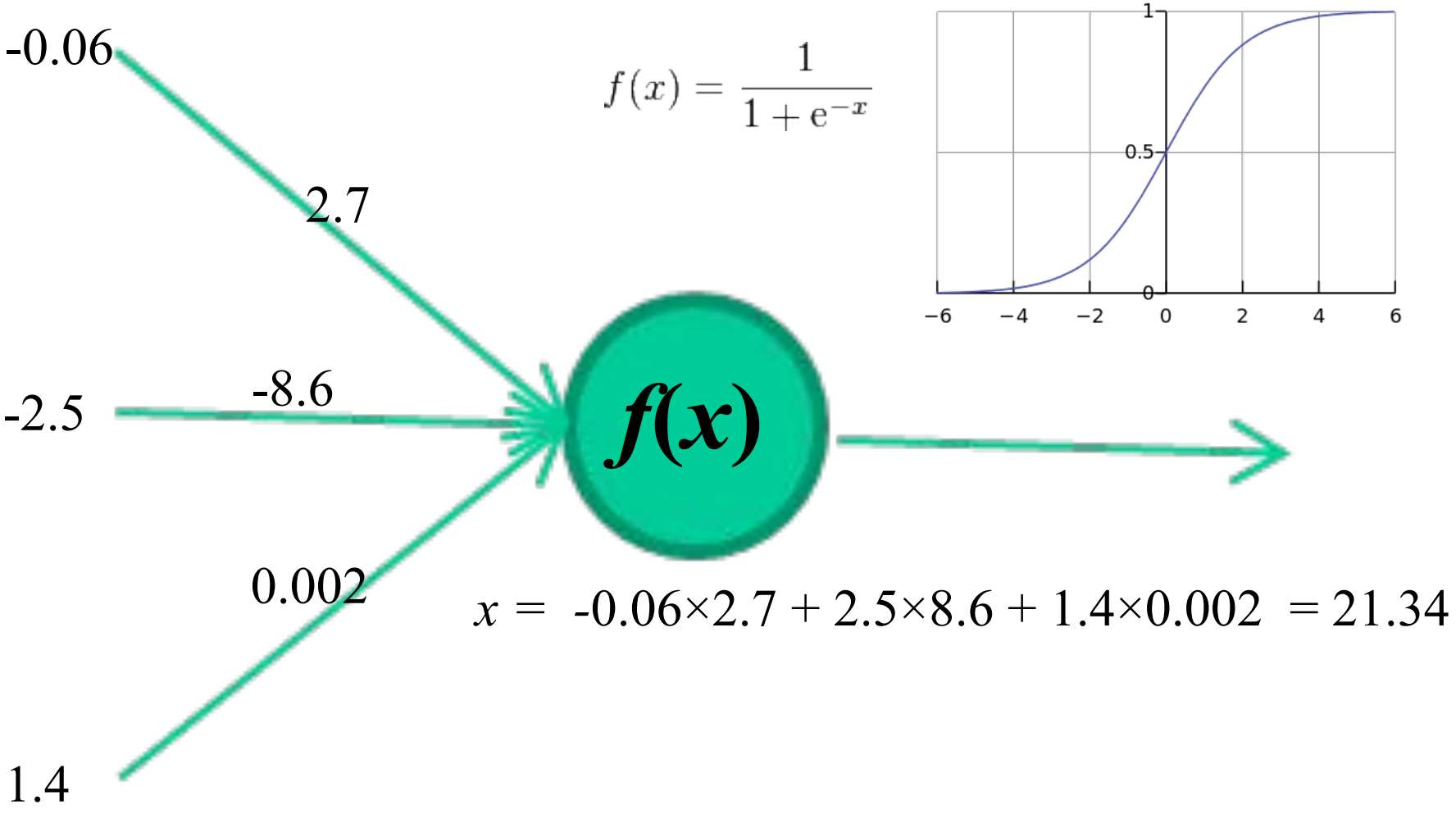
4 + 2 = 6 biases

26 learnable **parameters**

Example 1 of NN



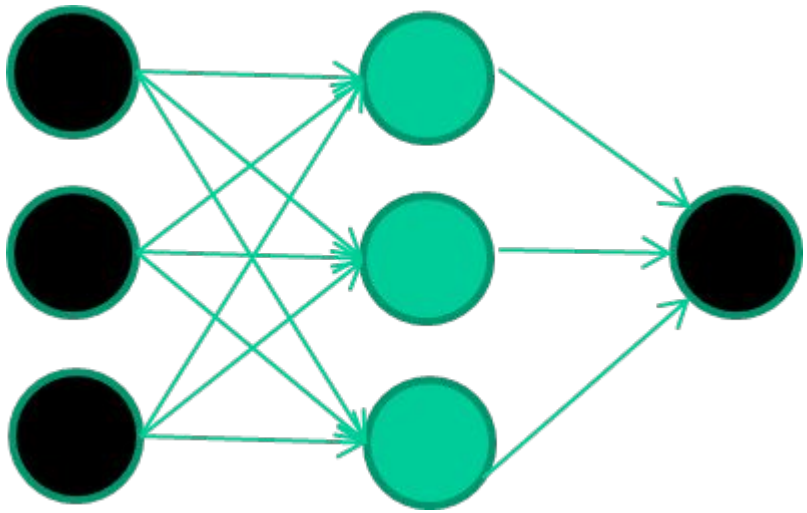
Example 1 of NN…



Example 1 of NN…

A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

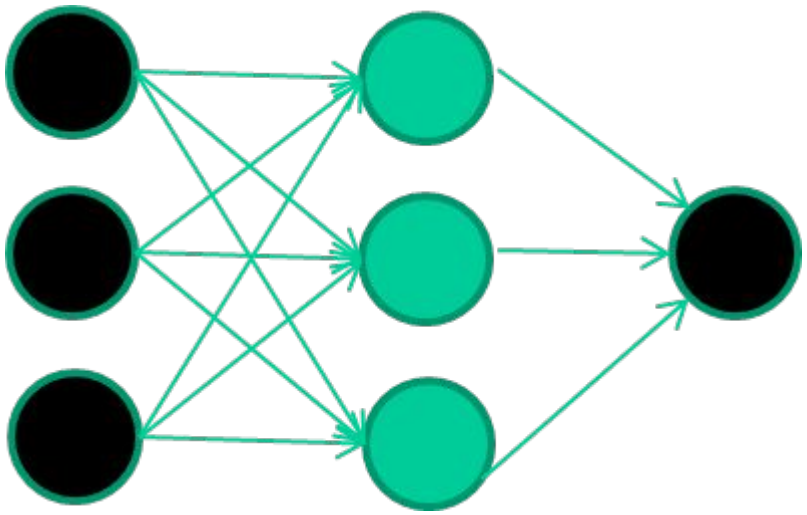


Example 1 of NN…

Training the neural network

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

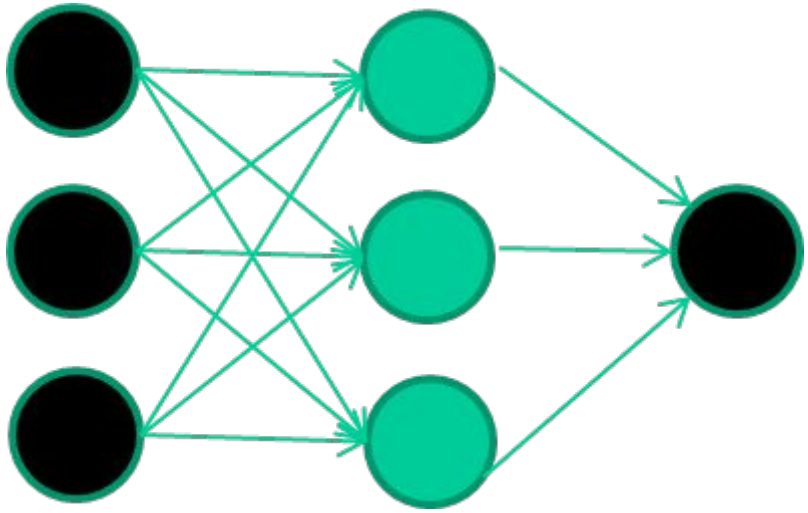


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Initialise with random weights

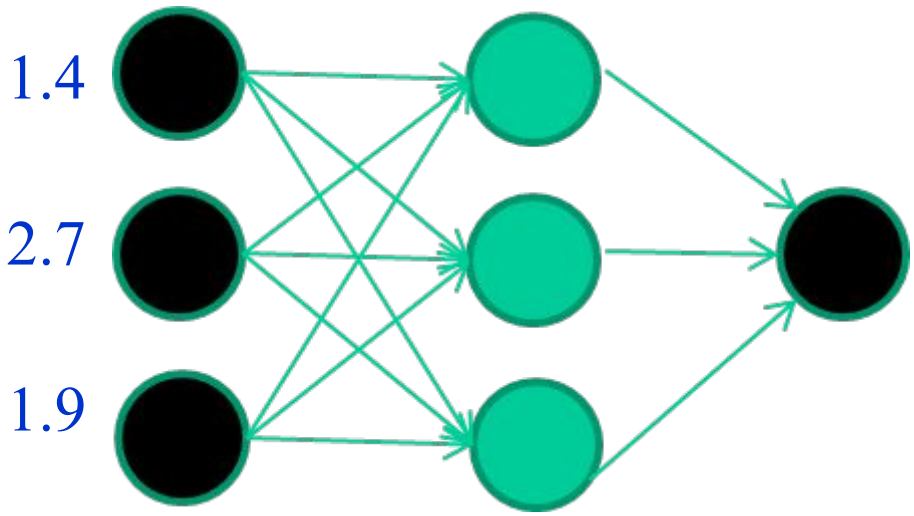


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Present a training pattern

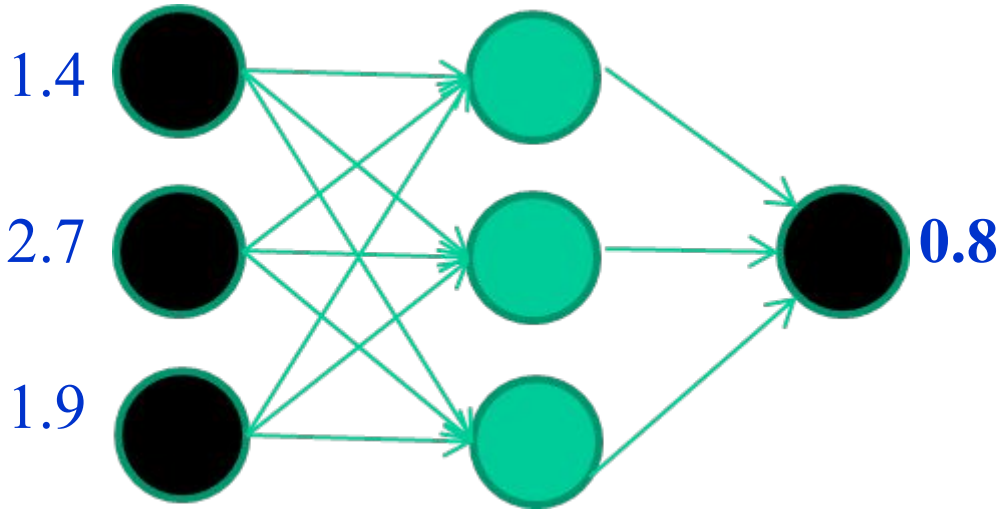


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output

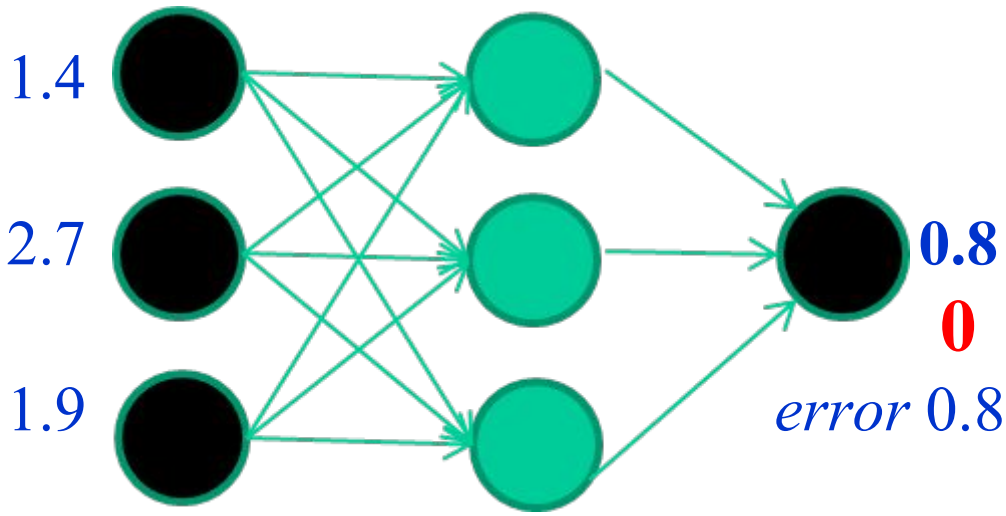


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output

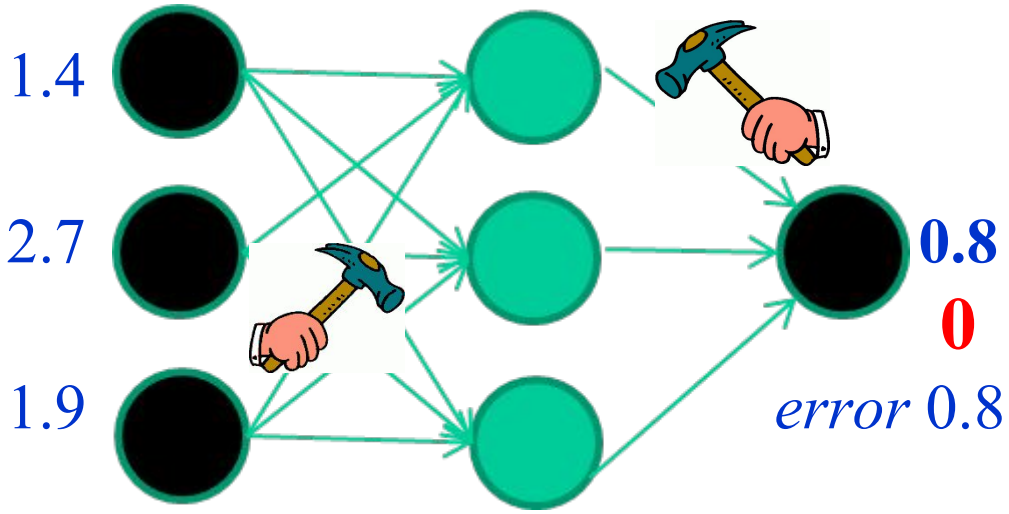


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error

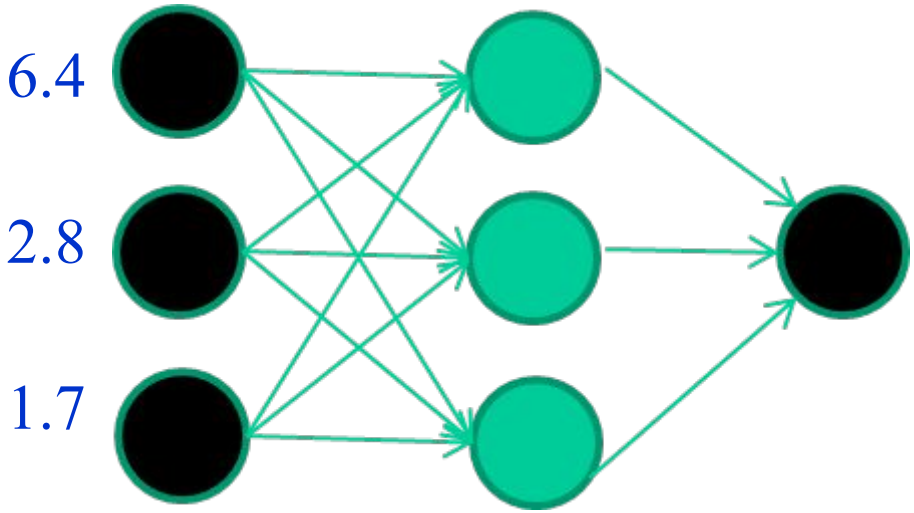


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Present a training pattern

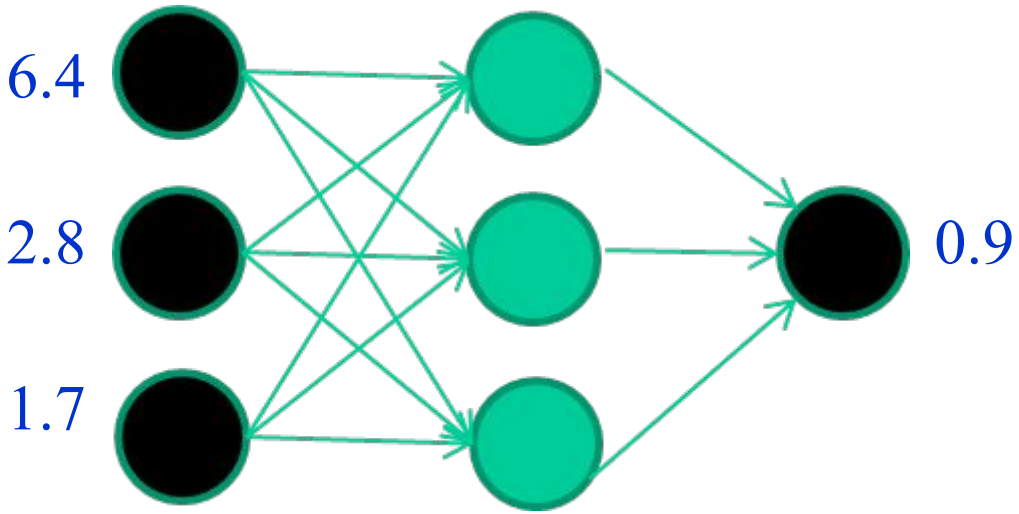


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output

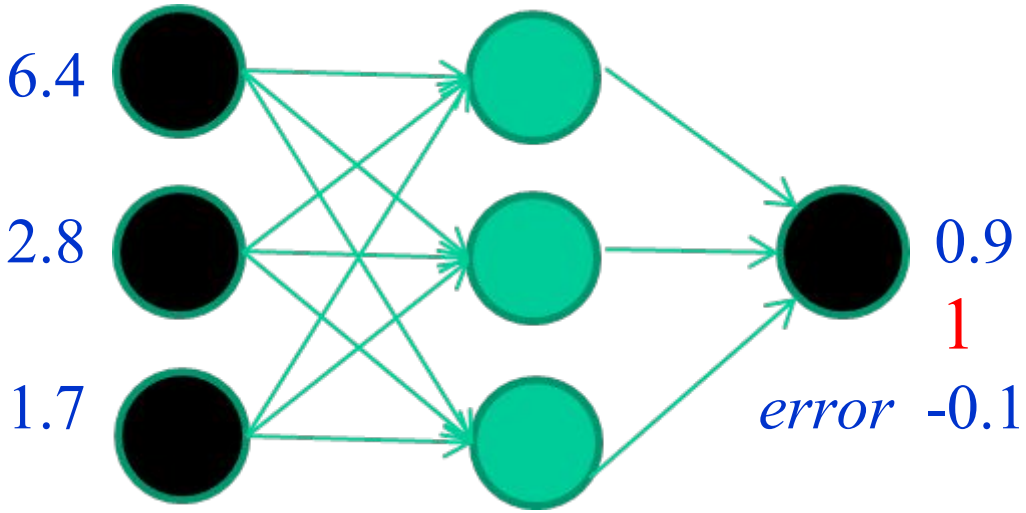


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output

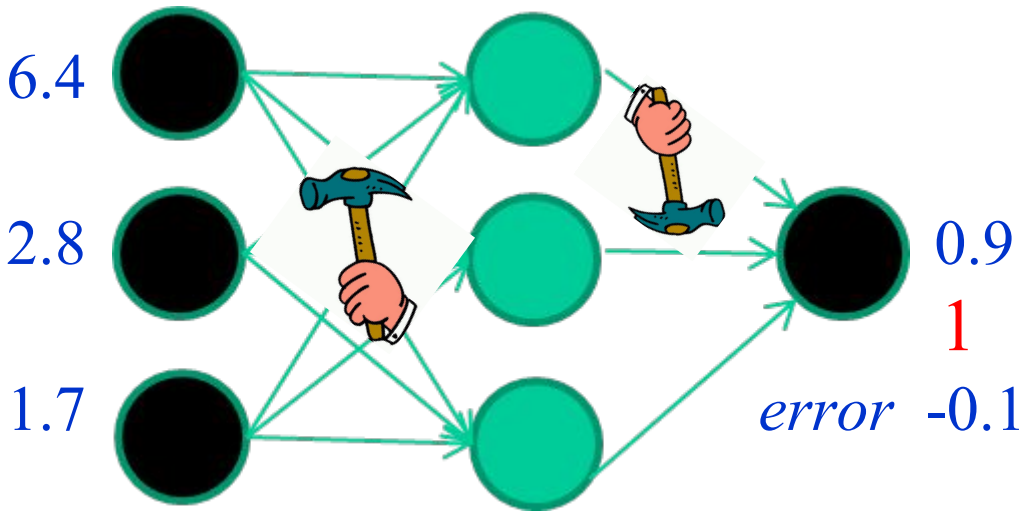


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error

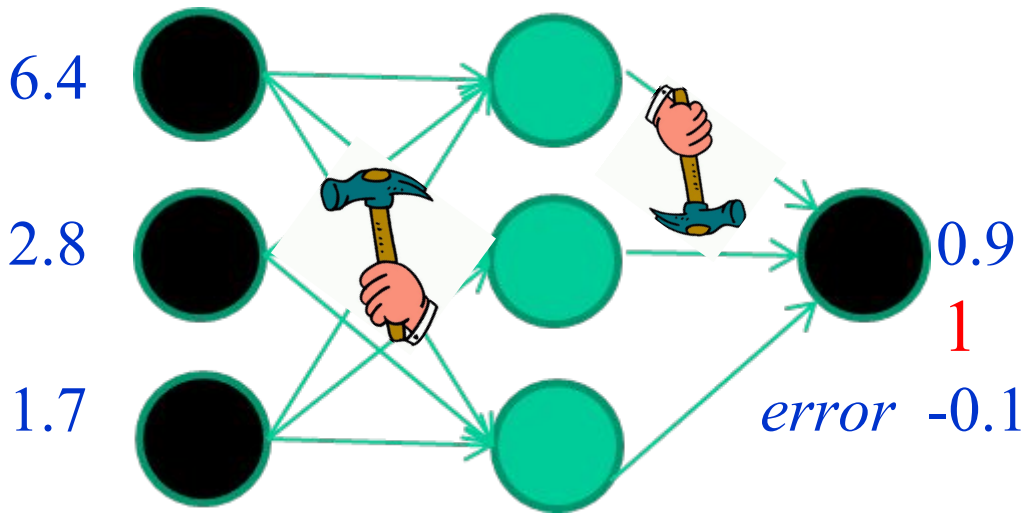


Example 1 of NN…

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

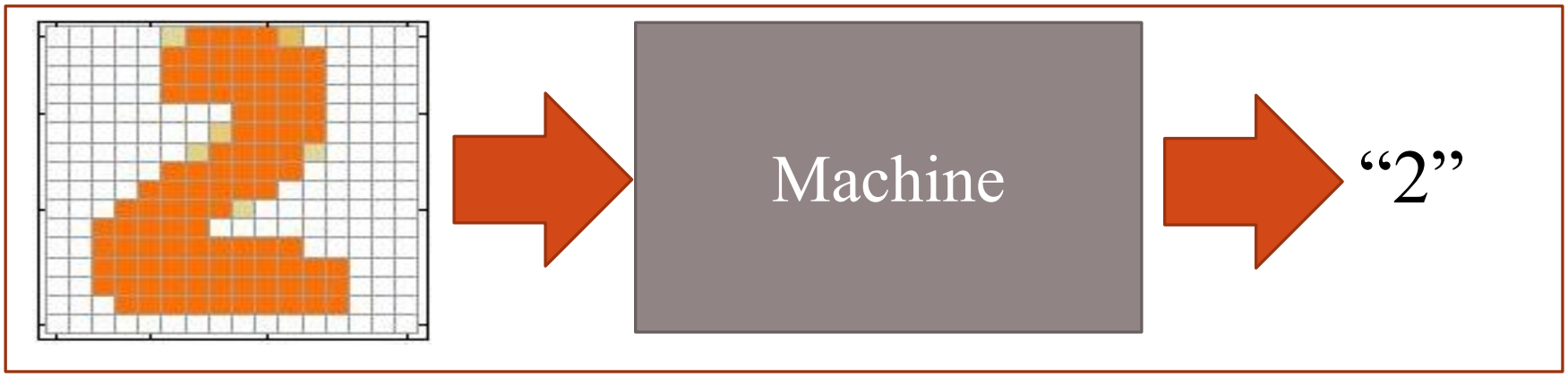
And so on ….

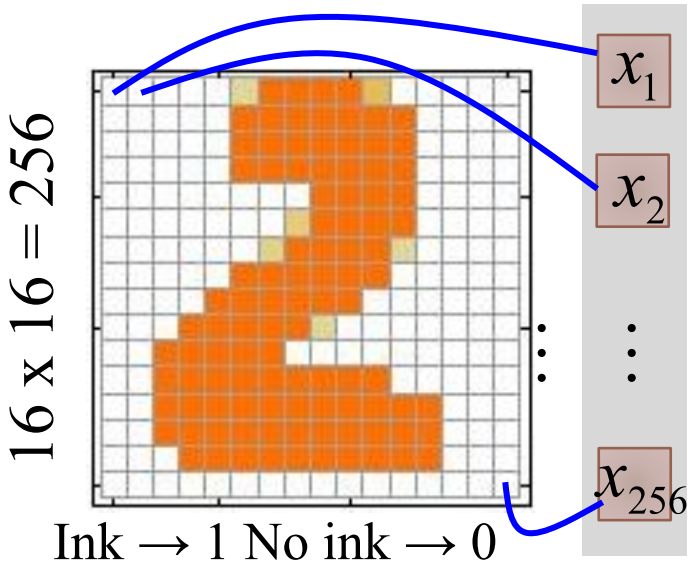


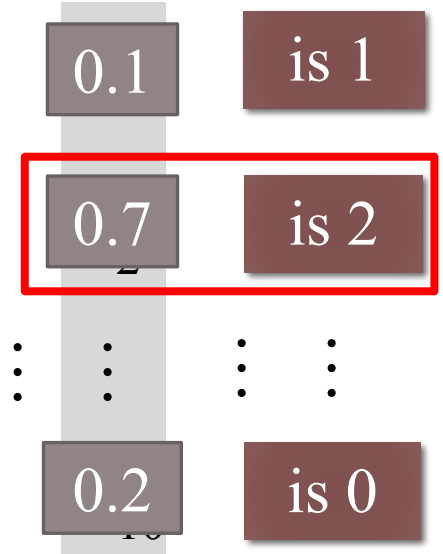
Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments

Algorithms for weight adjustment are designed to make changes that will reduce the error

Example of Digit Recognition



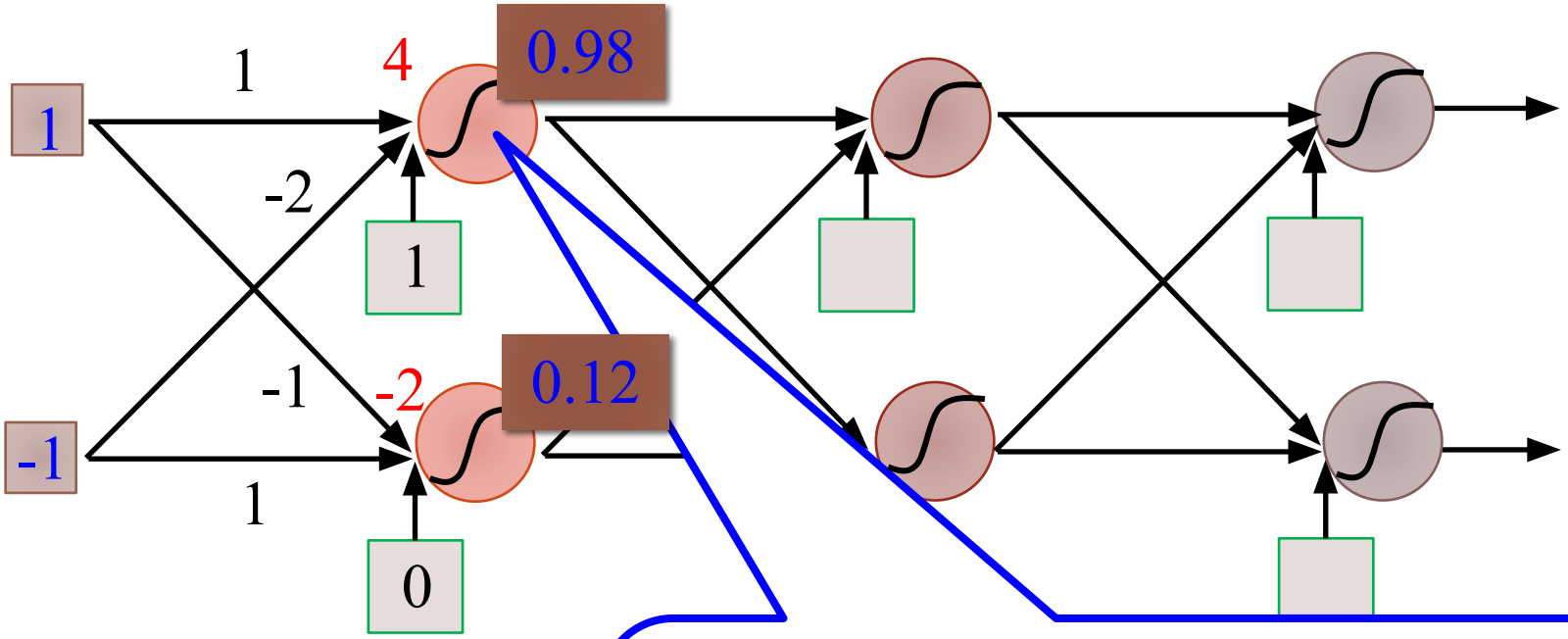




The image is "2"

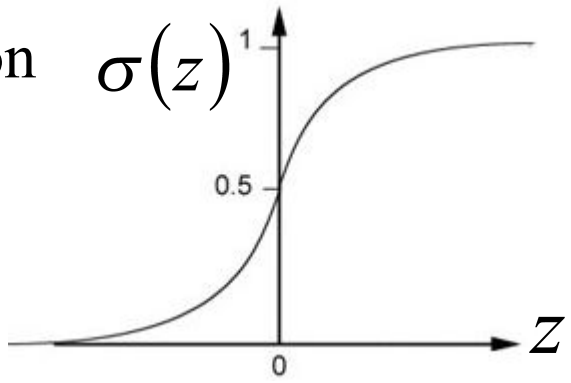
$f: R^{256} \rightarrow R^{10}$

Example of Neural Network

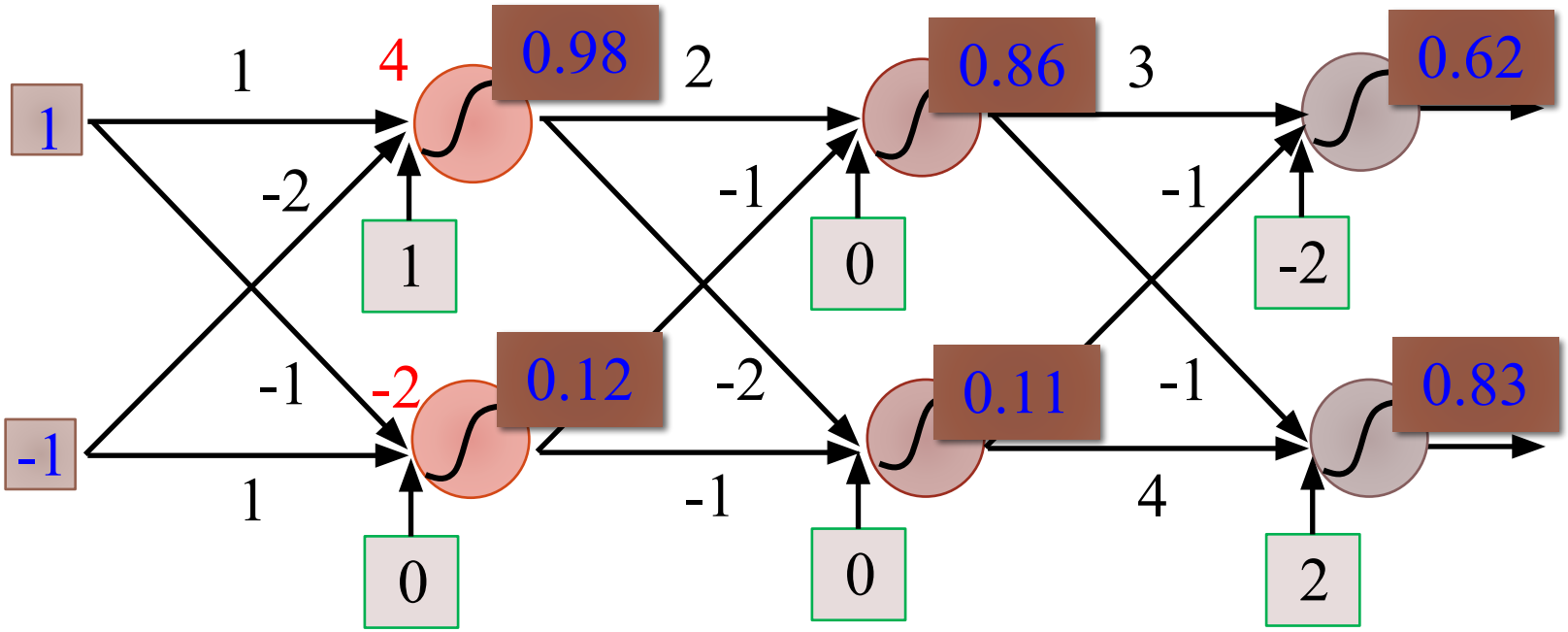


Sigmoid Function

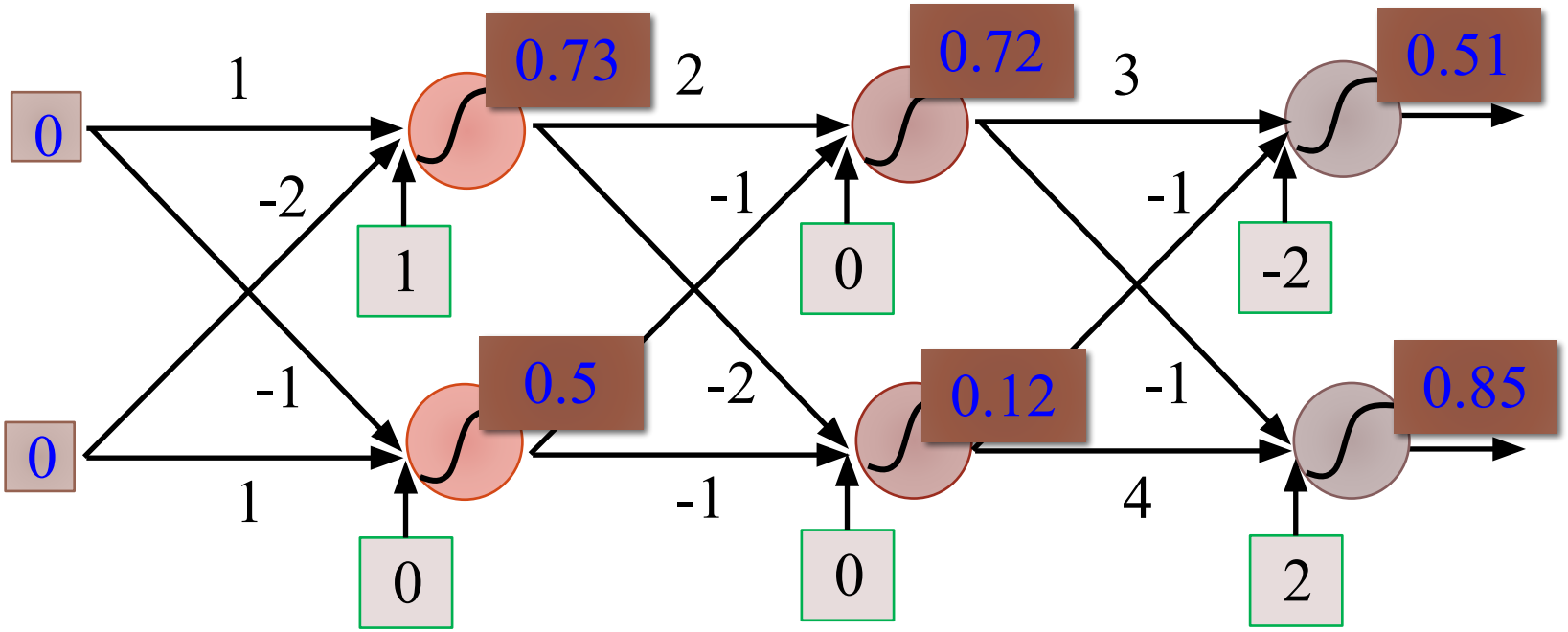
$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Example of Neural Network



Example of Neural Network

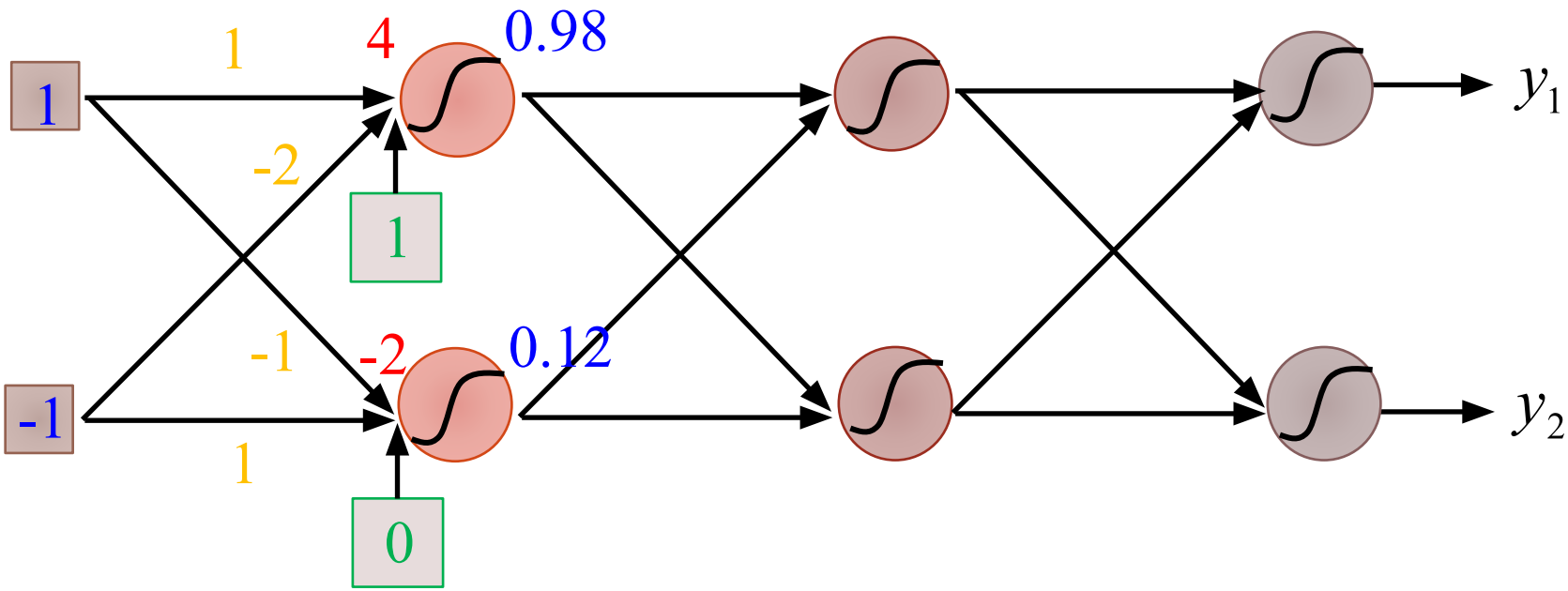


$f: R^2 \rightarrow R^2$

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right) = \begin{bmatrix}0.62\\0.83\end{bmatrix} \quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right) = \begin{bmatrix}0.51\\0.85\end{bmatrix}$$

Different parameters define different function

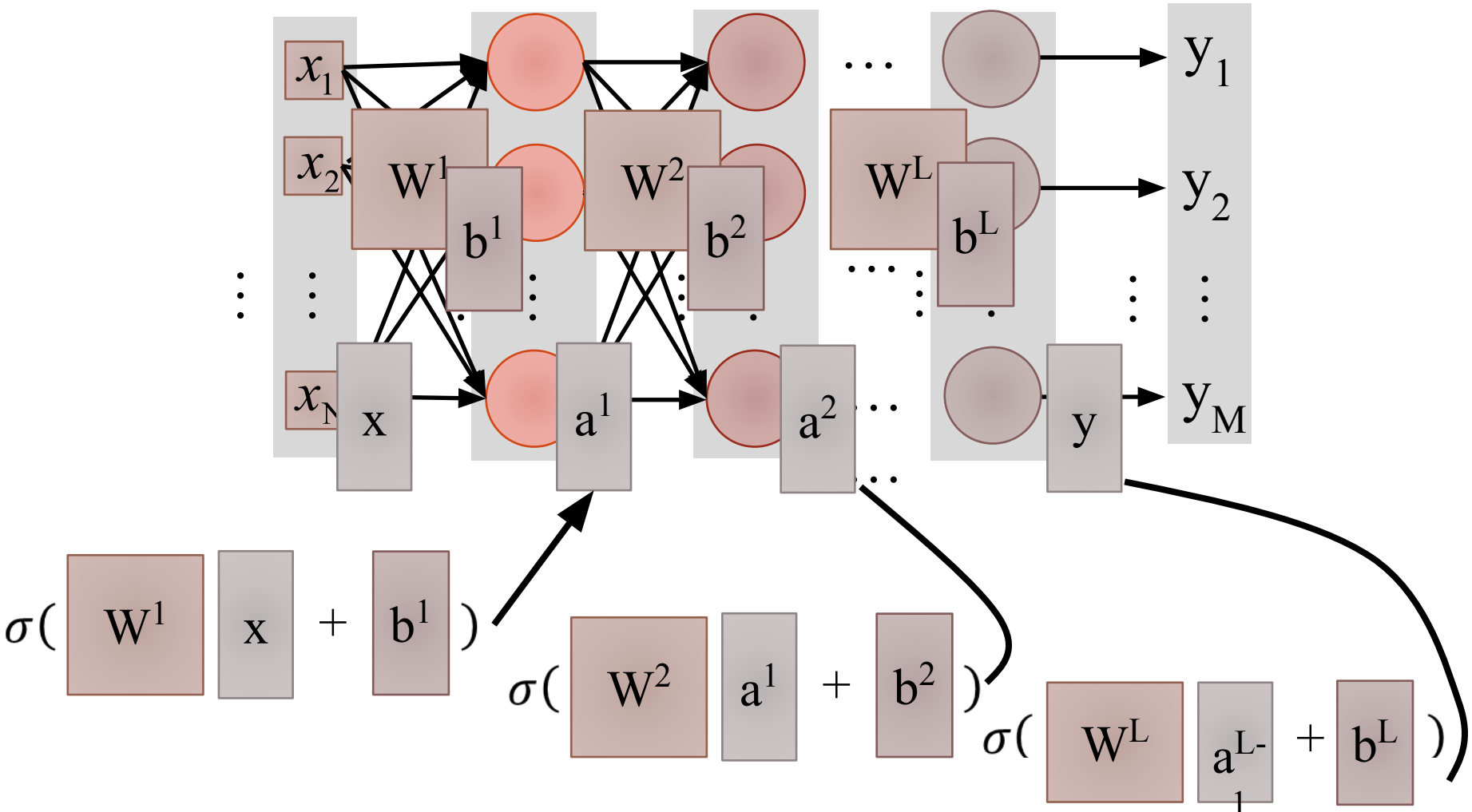
Example of Neural Network



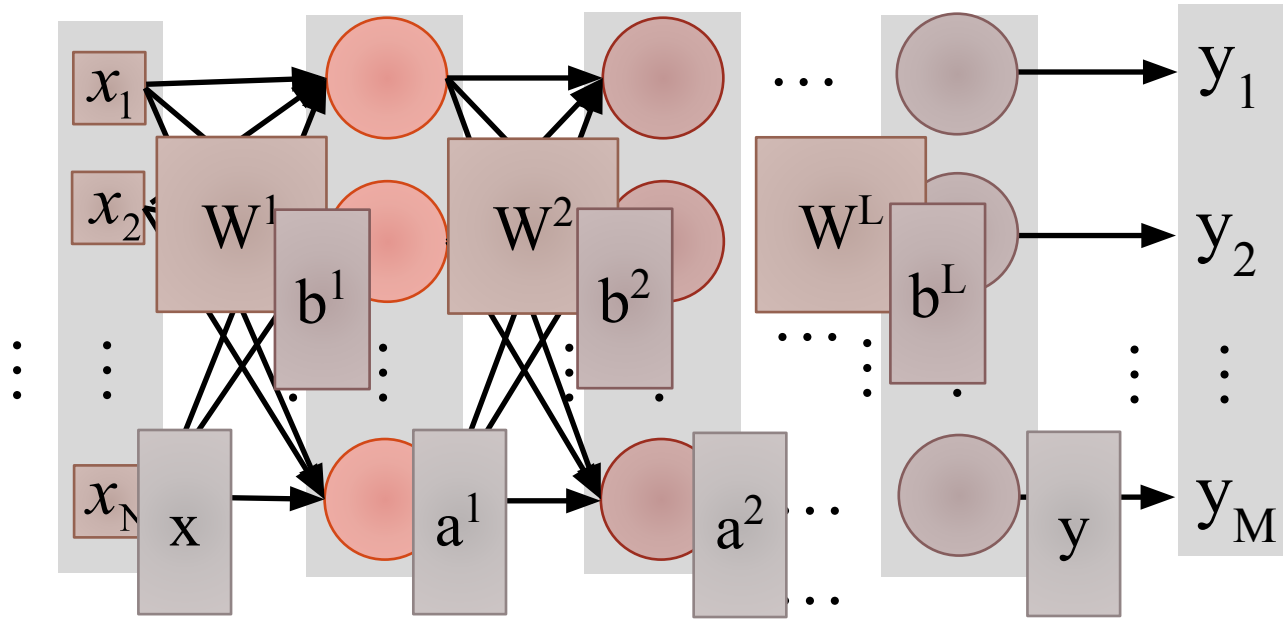
$$\sigma\left(\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

$$\begin{bmatrix} 4 \\ -2 \end{bmatrix}$$

Example of Neural Network



Neural Network



$$\mathbf{y} = f(\mathbf{x})$$

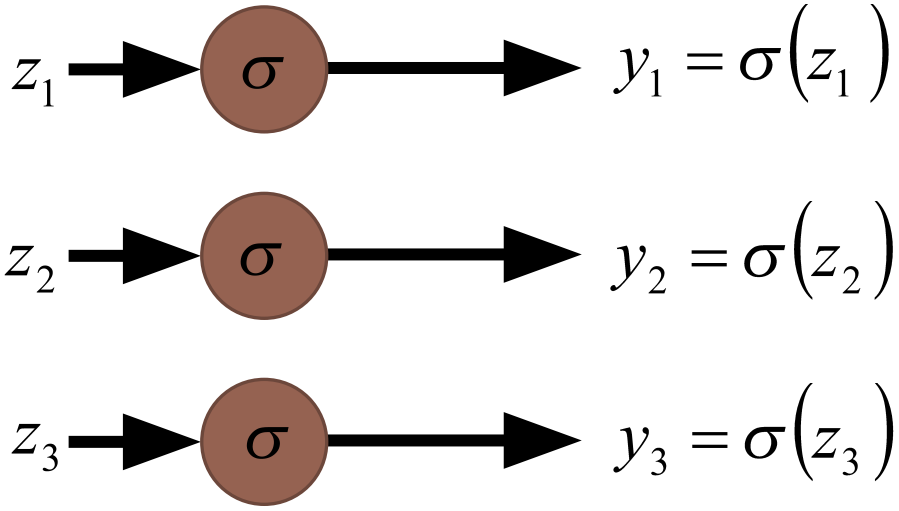
Using parallel computing techniques to speed up matrix operation

$$= \sigma(\mathbf{W}^L \cdots \sigma(\mathbf{W}^2 \sigma(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) \cdots + \mathbf{b}^L)$$

Softmax

- Softmax layer as the output layer

Ordinary Layer



In general, the output of network can be any value.

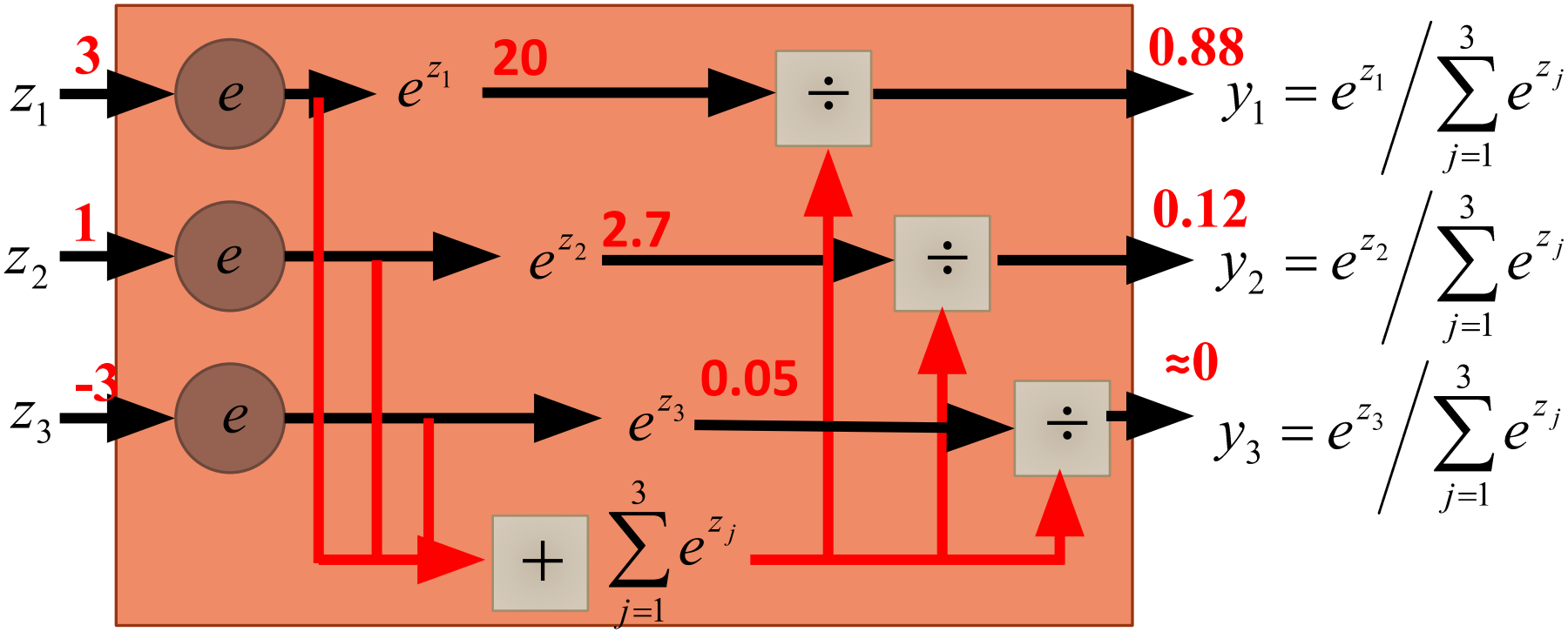
May not be easy to interpret

Softmax

- Softmax layer as the output layer

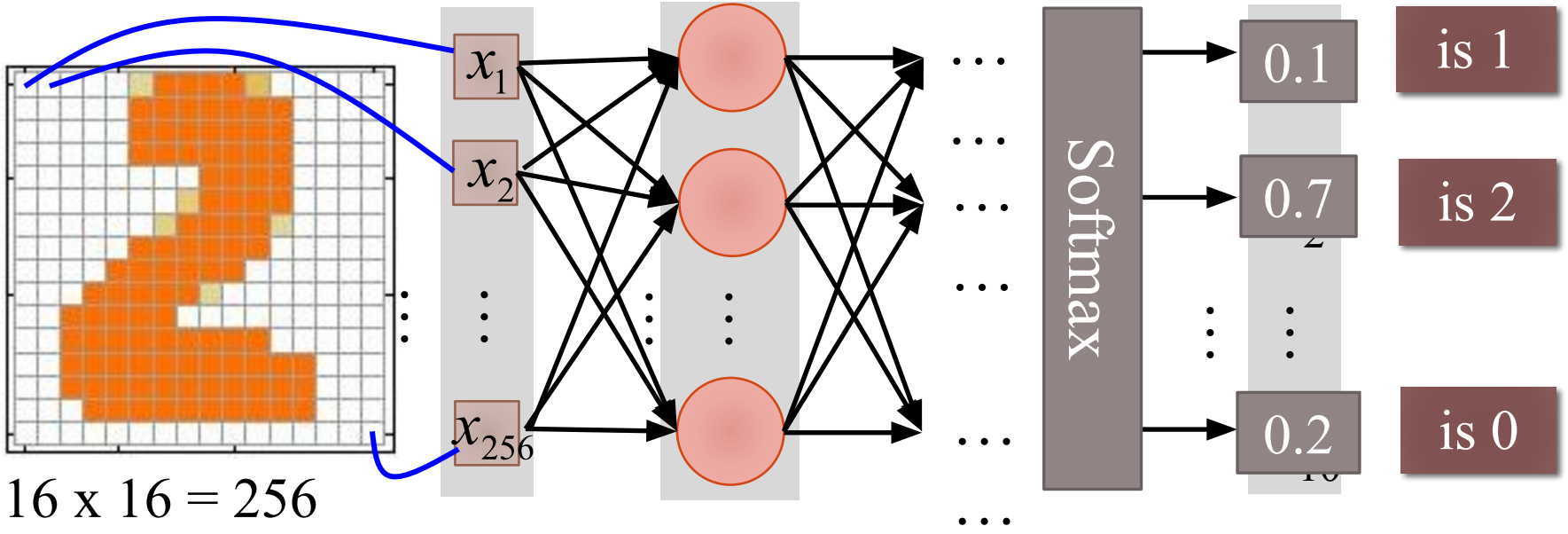
Probability:

- $1 > y_i > 0$
- $\sum_i y_i = 1$



Network Parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



16 x 16 = 256

Ink → 1

No ink → 0

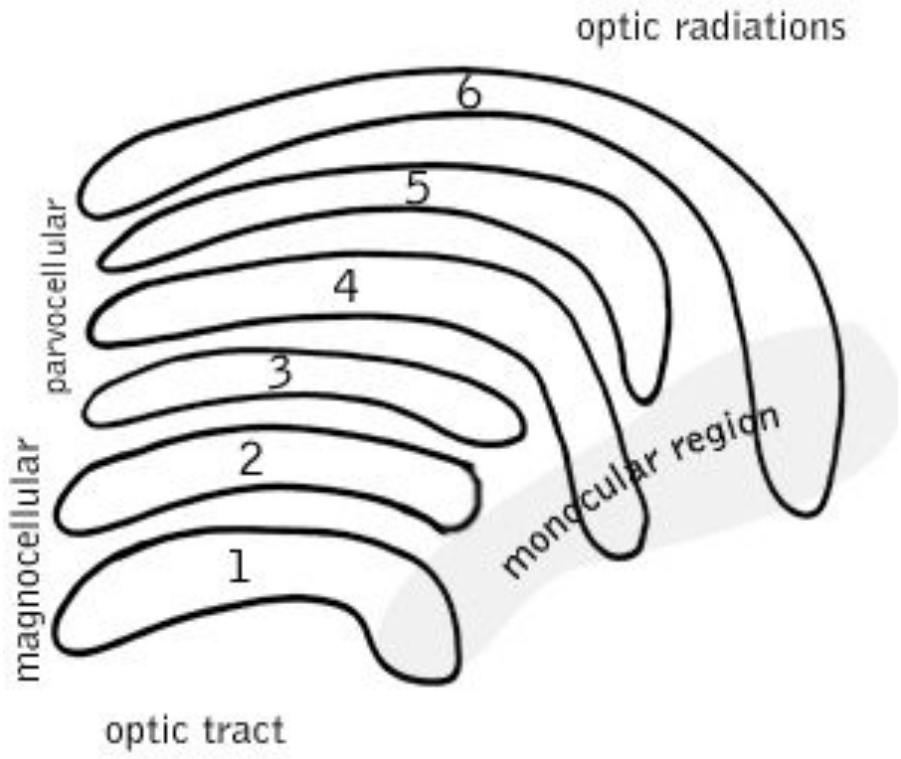
Set the network parameters such that ……

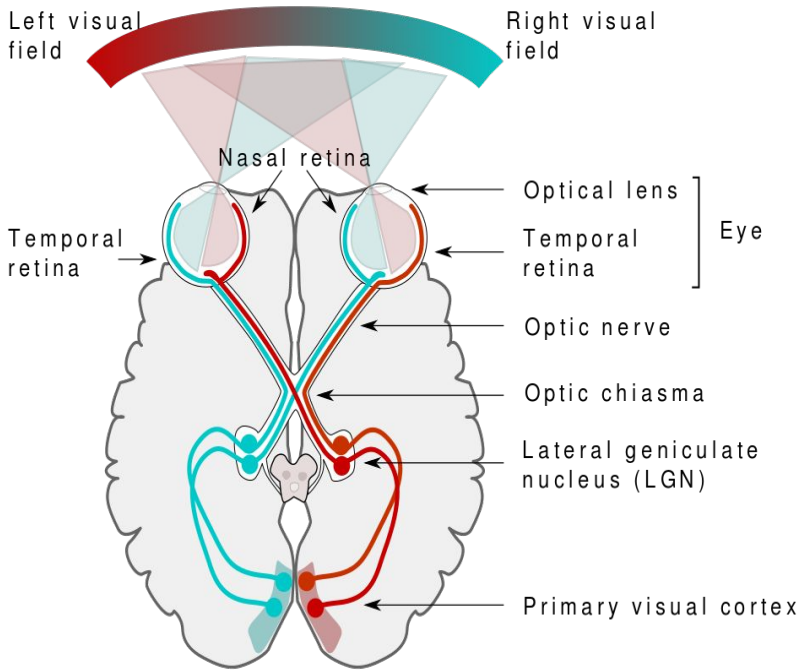
Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

Visual Information Processing

- Visual information processed by our brain is multi-layered.



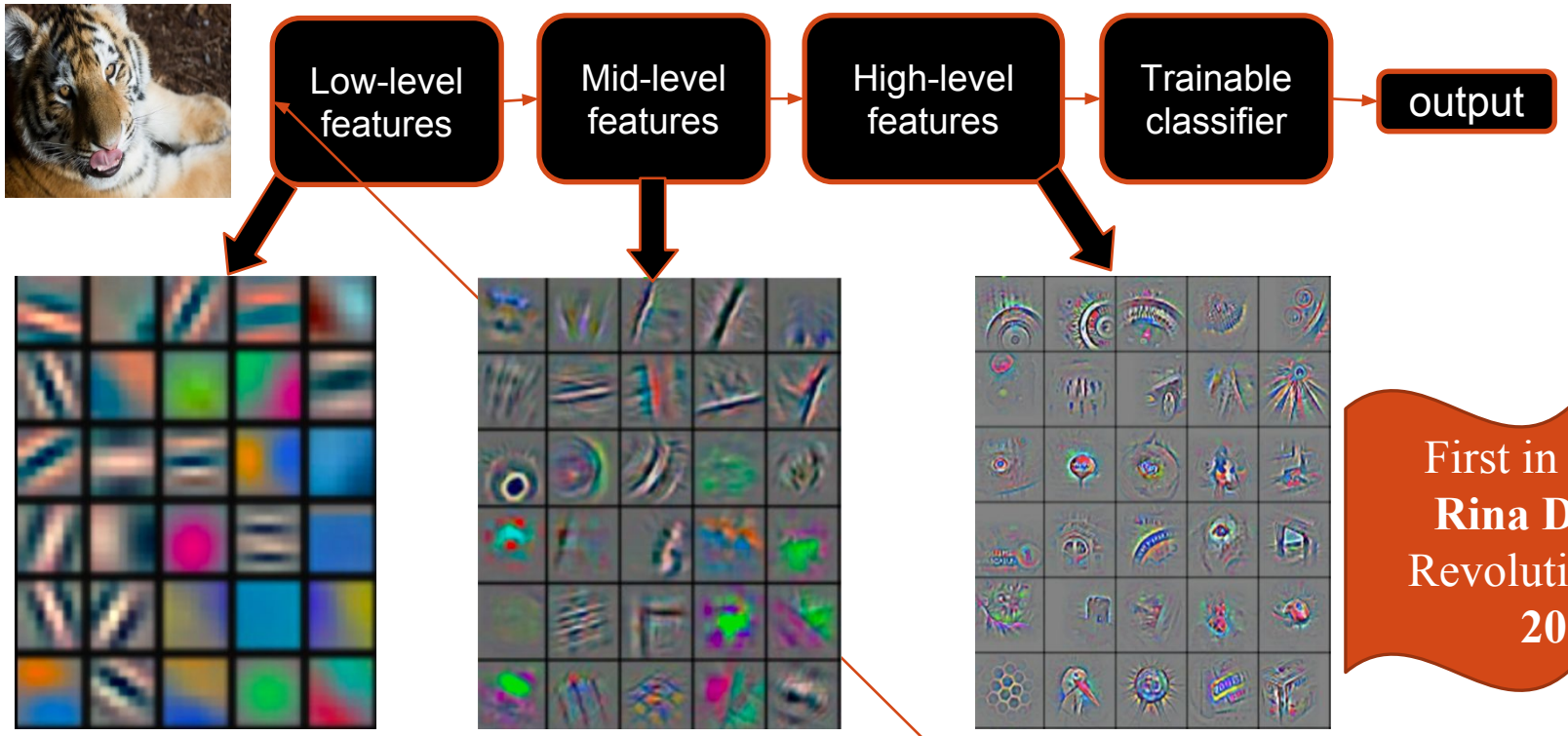


Enabling Factor of DL

- Training of deep networks was made computationally feasible by:
  - Faster CPU's
  - The move to parallel CPU architectures
  - **Advent of GPU computing**
- Neural networks are often represented as a matrix of weight vectors.
- GPU's are optimized for very fast matrix multiplication
- 2008 - Nvidia's CUDA library for GPU computing is released.

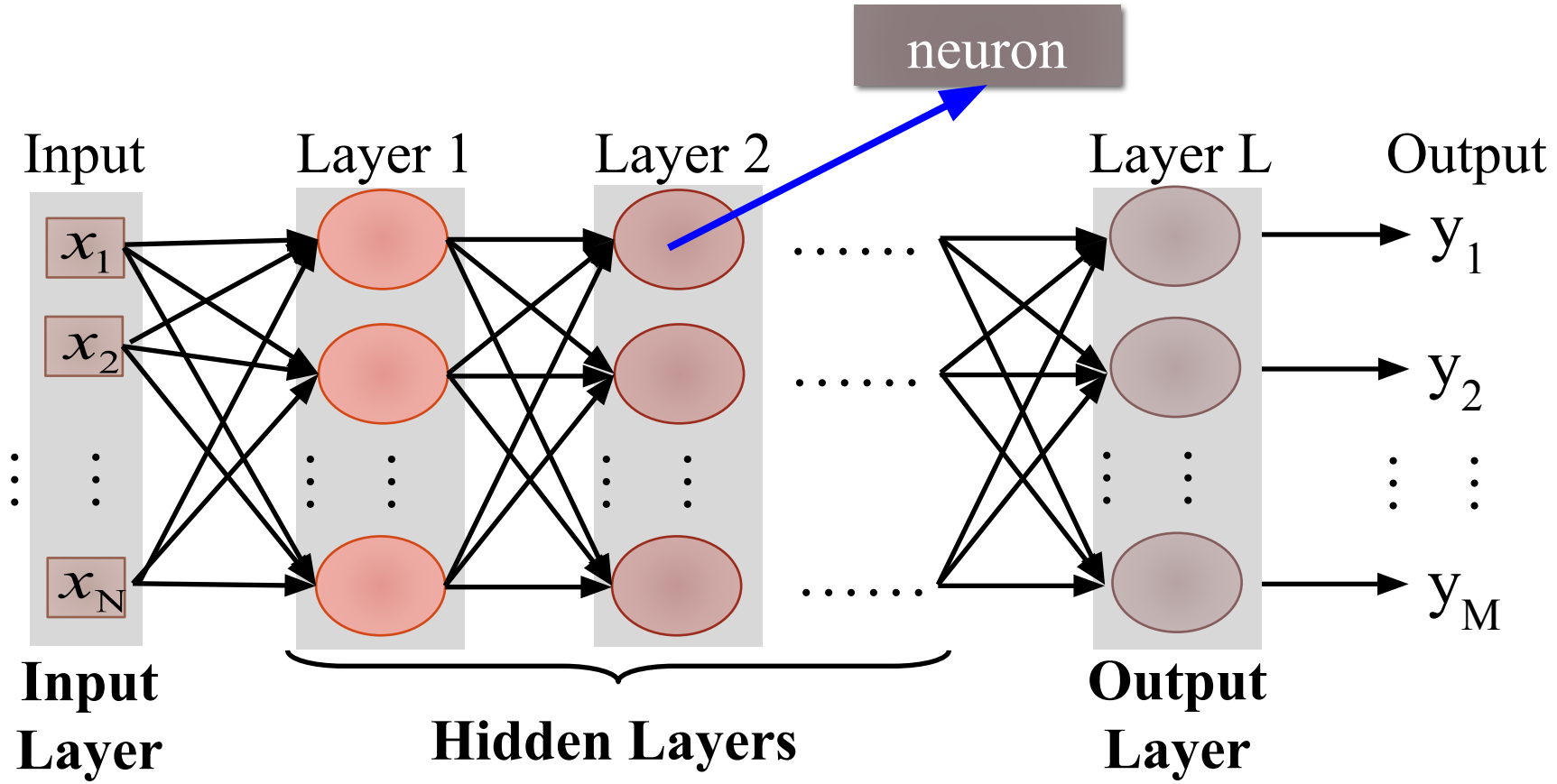
Hierarchical Learning

Inspired from visual information processing, a representation of Hierarchical Learning is developed, also know as **"Deep Learning"**



First in 1986 by **Rina Dechter**
Revolution **since 2012**

Deep Neural Network



Deep means many hidden layers

Why Deep Network?

Layer X Size	Word Error Rate (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

Not surprised, more parameters, better performance

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Why Deep Network?

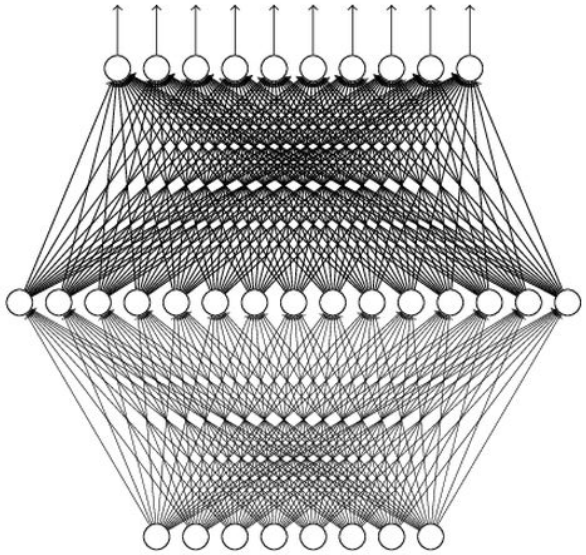
- Universal Theorem

  Any continuous function f

$$f : R^N \rightarrow R^M$$

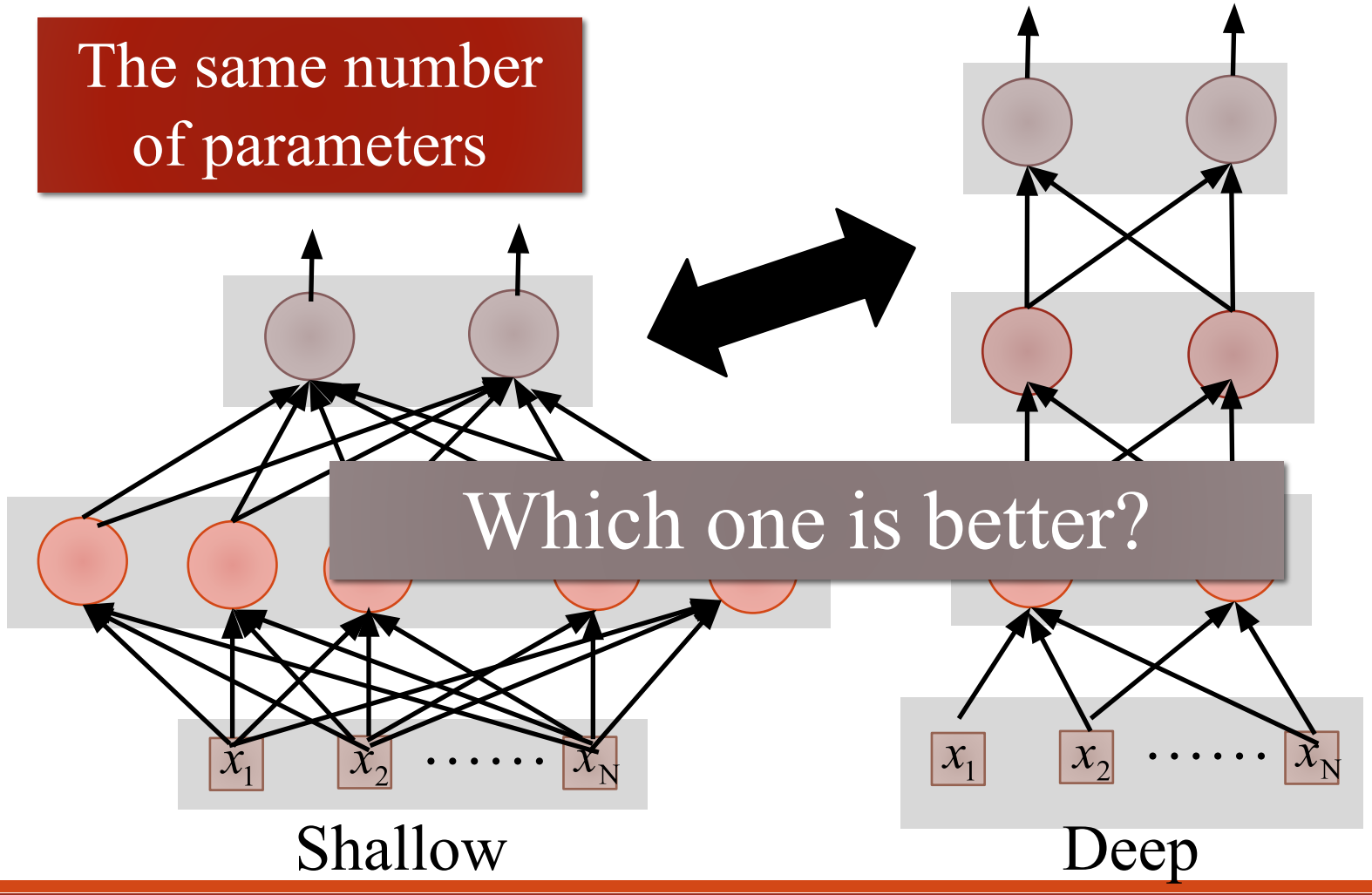
  Can be realized by a network with one hidden layer

  (given **enough** hidden neurons)



Why “Deep” neural network not “Fat” neural network?

Fat + Short v.s. Thin + Tall

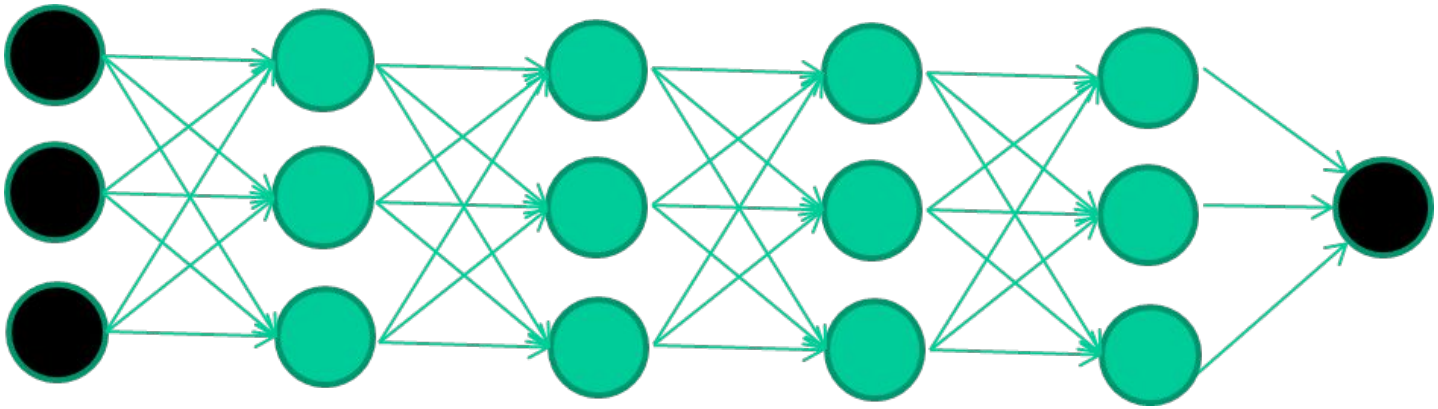


Fat + Short v.s. Thin + Tall

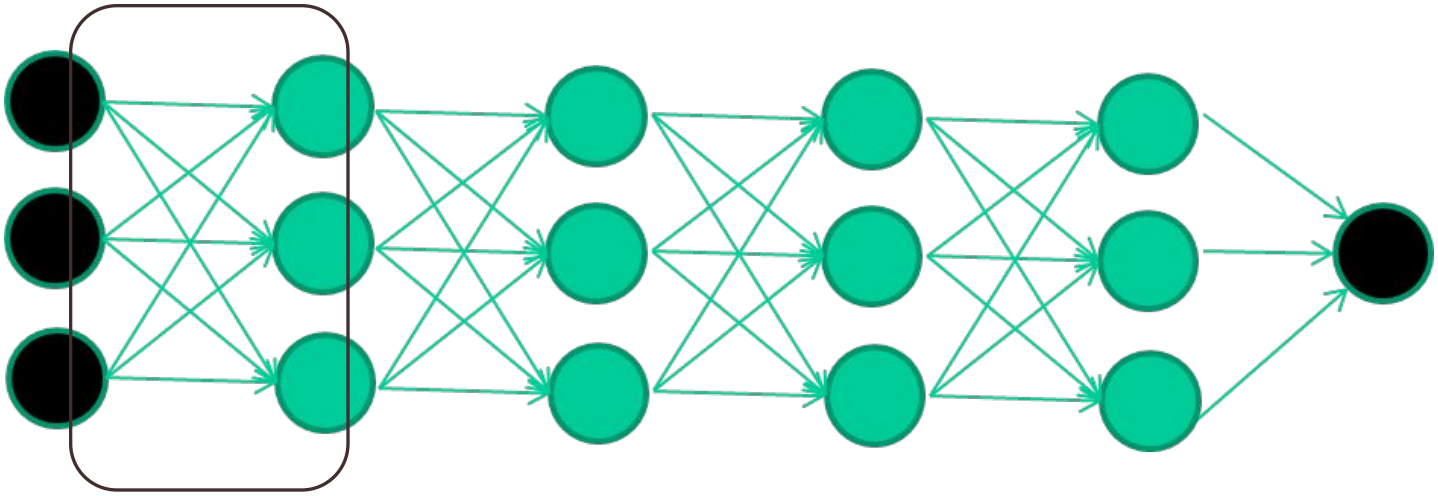
Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Training multi-layer NNs (DNN)

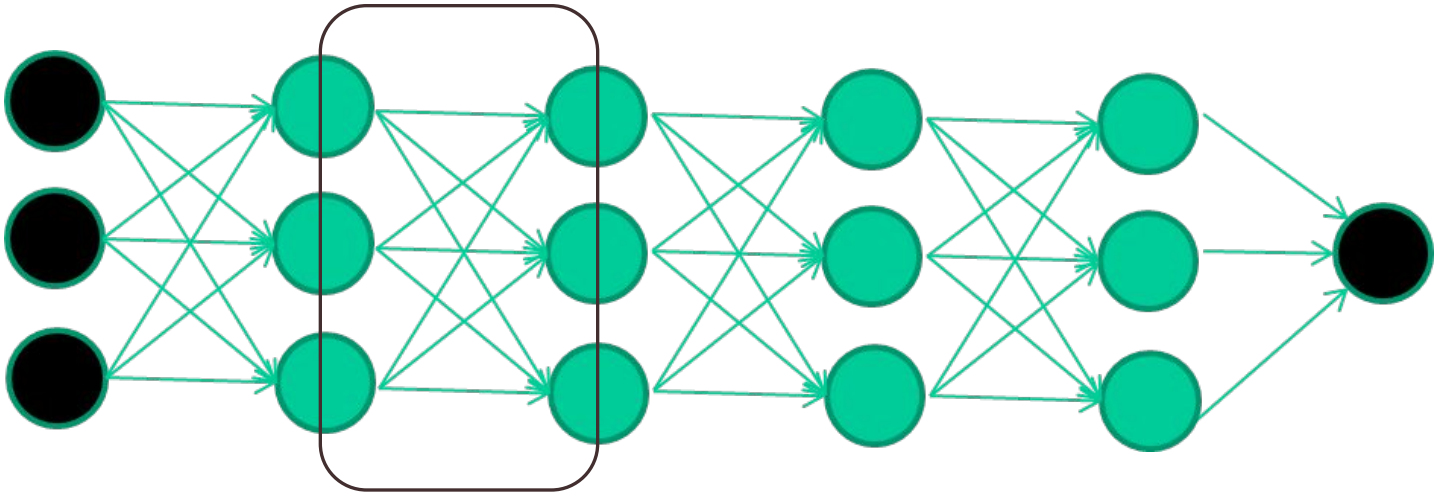


Training multi-layer NNs



Train **this** layer first

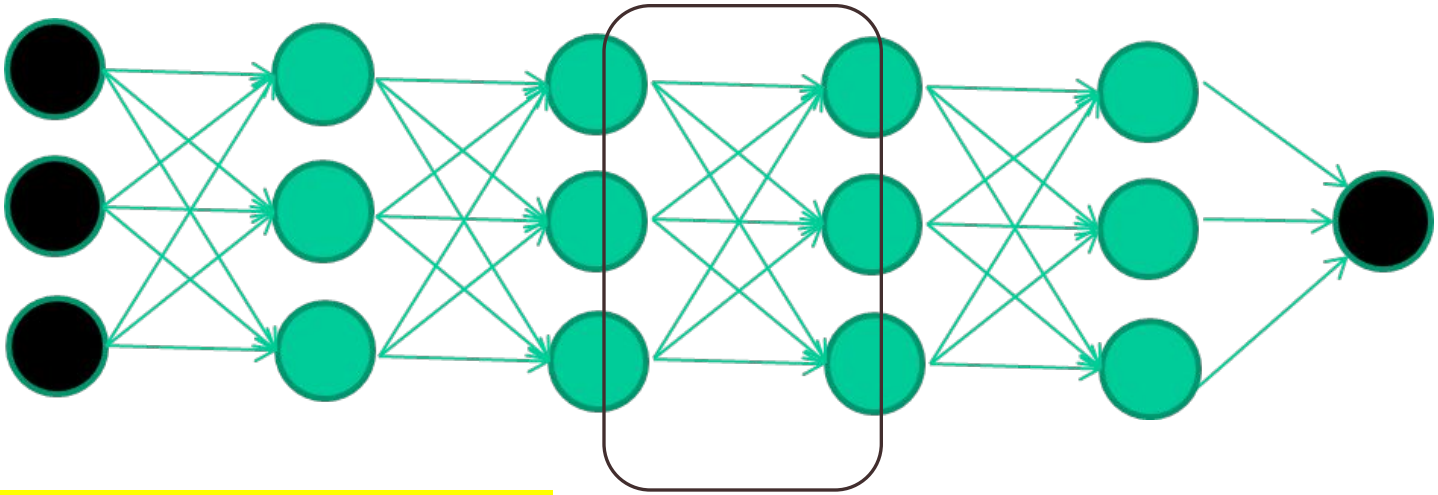
Training multi-layer NNs



Train **this** layer first

then **this** layer

Training multi-layer NNs

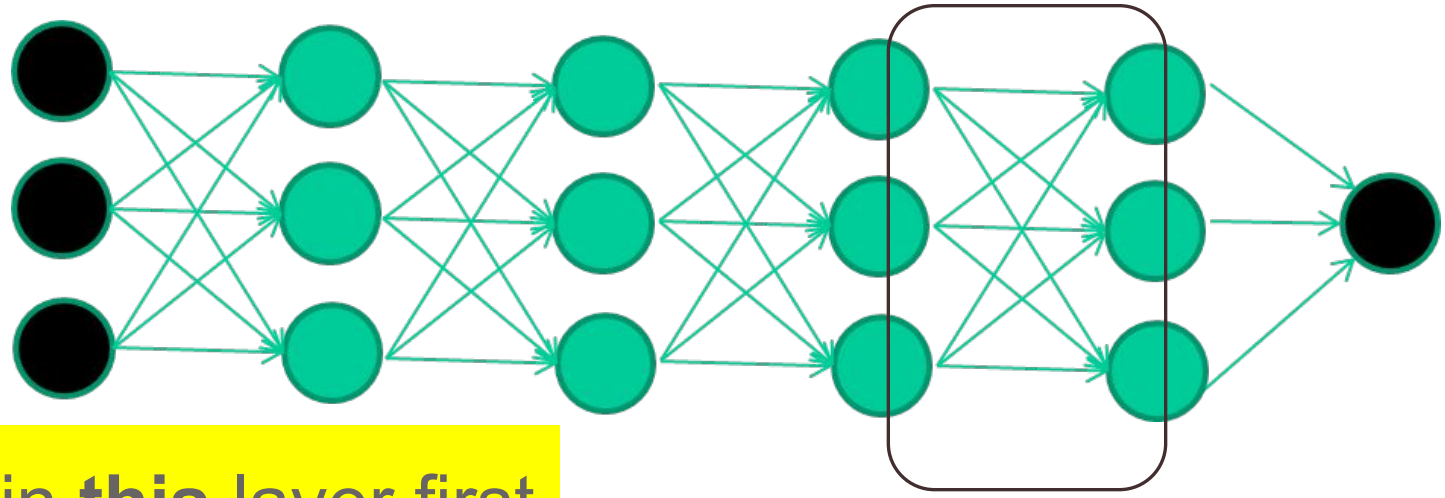


Train **this** layer first

then **this** layer

then **this** layer

Training multi-layer NNs



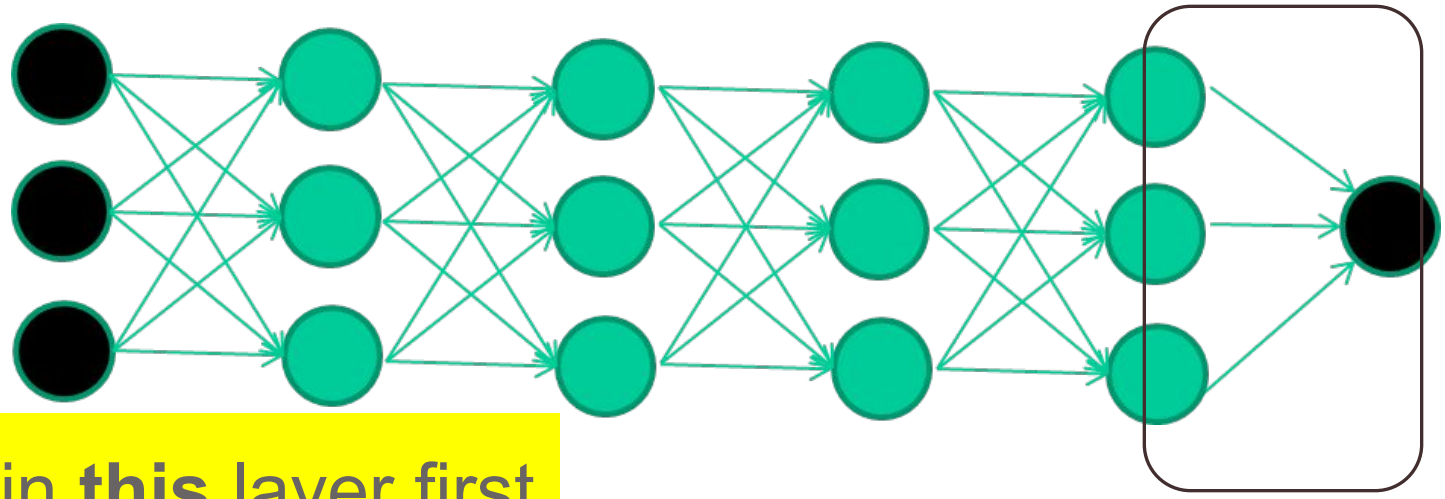
Train **this** layer first

then **this** layer

then **this** layer

then **this** layer

Training multi-layer NNs



Train **this** layer first

then **this** layer

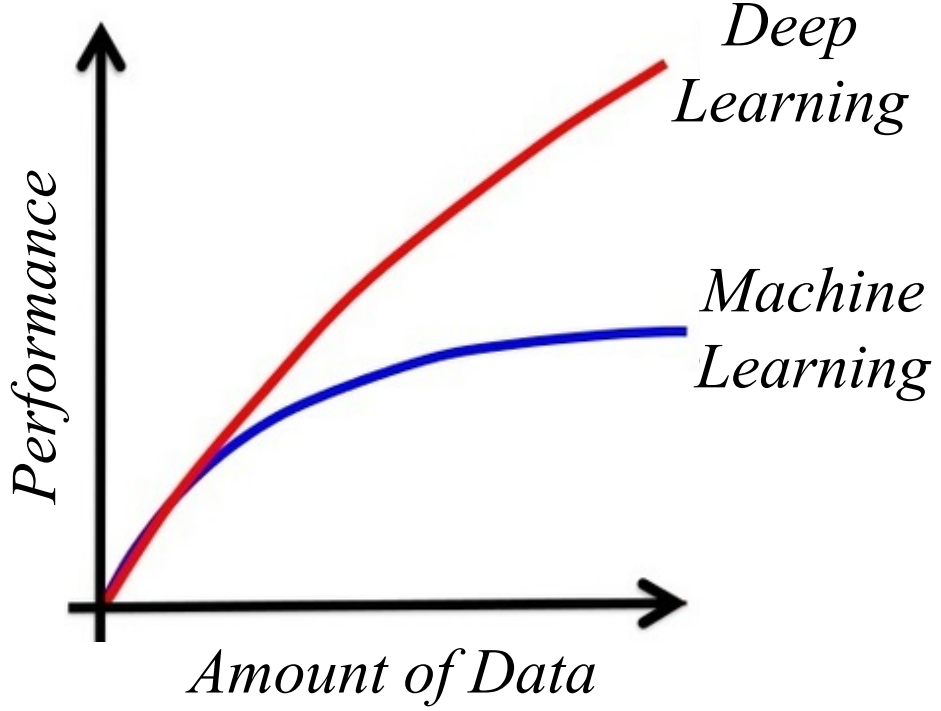
then **this** layer

then **this** layer

finally **this** layer

When to use Deep Learning?

- Data size is large
- High end infrastructure
- Lack of domain understanding
- Complex problem such as image classification, speech recognition etc.



Fuel of deep learning is the big data by Andrew Ng

Limitations of Deep Learning

- Very slow to train
- Models are very complex, with lot of parameters to optimize:

✔ Initialization of weights

✔ Layer-wise training algorithm

✔ Neural architecture

- Number of layers
- Size of layers
- Type – regular, pooling, max pooling, soft max

✔ Fine-tuning of weights using back propagation

Thank you!

dinesh@dtu.ac.in