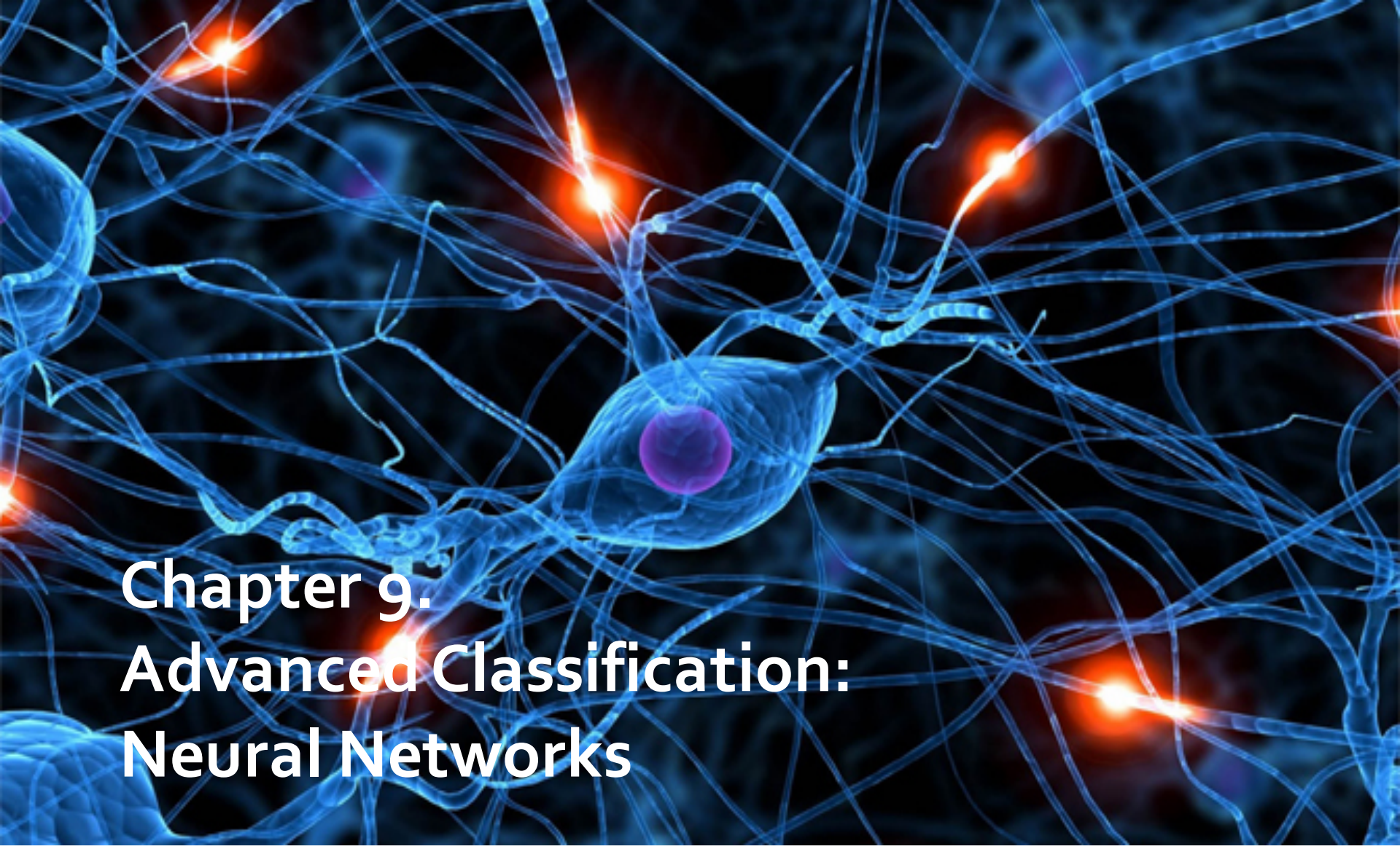


Chapter 9. Advanced Classification: Neural Networks

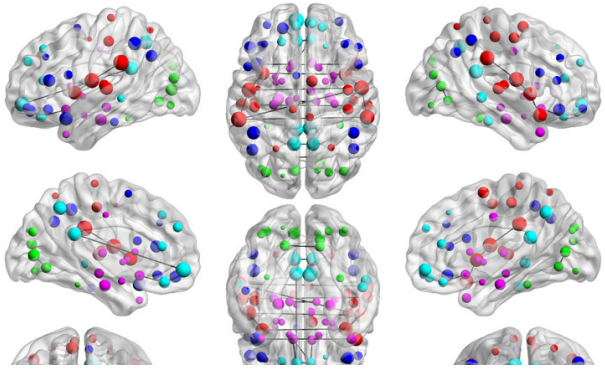
Meng Jiang

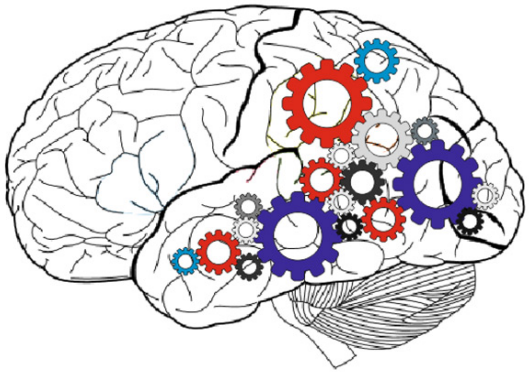
CSE 40647/60647 Data Science Fall 2017

Introduction to Data Mining

Artificial Neural Networks (ANNs)

- **Computing systems** inspired by the *biological neural networks* that constitute *animal brains*



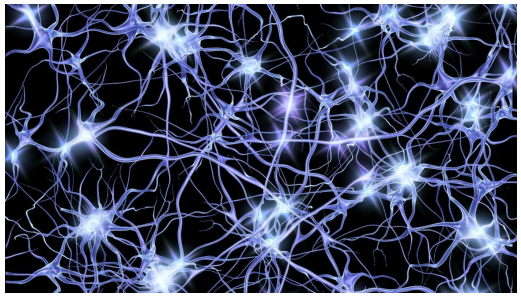
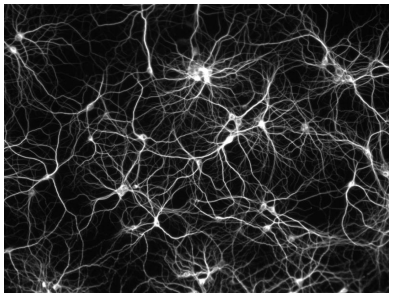
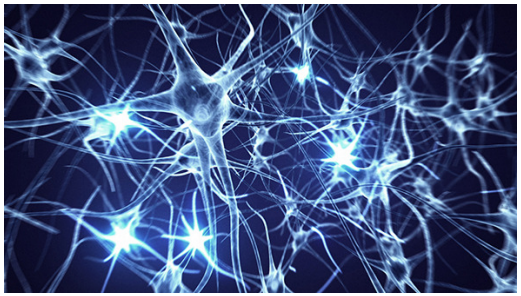


Biological Inspirations

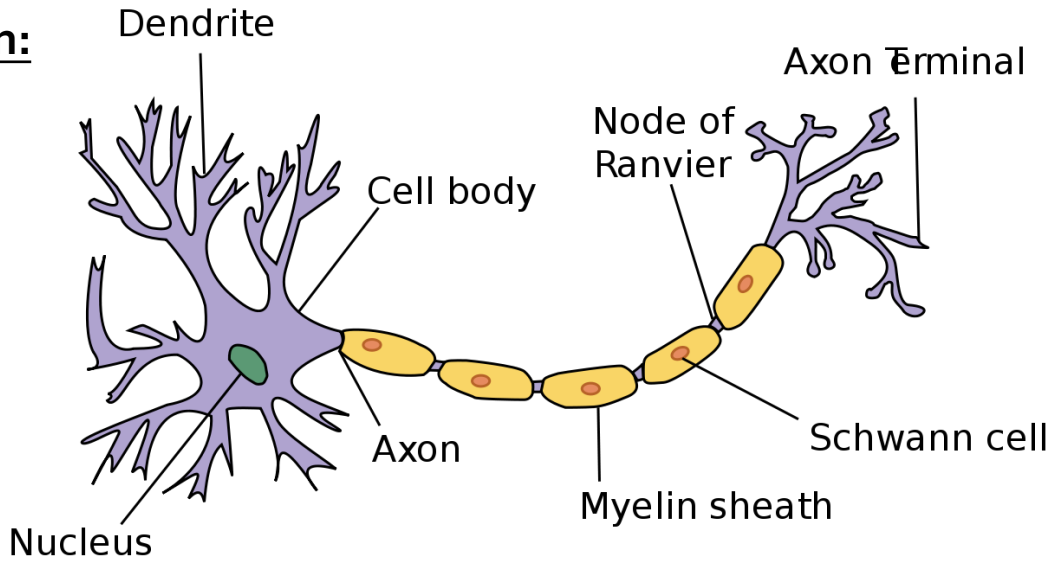
- Some numbers
  - The human brain contains about **10,000,000,000** (10B) nerve cells (neurons)
  - Each neuron is connected to the others through **10,000** synapses
- Properties of the brain
  - It can learn, reorganize itself from experience
  - It adapts to the environment
  - It is robust and fault tolerant

Neurons

- Started by psychologists and neurobiologists to develop and test computational analogues of ***neurons***

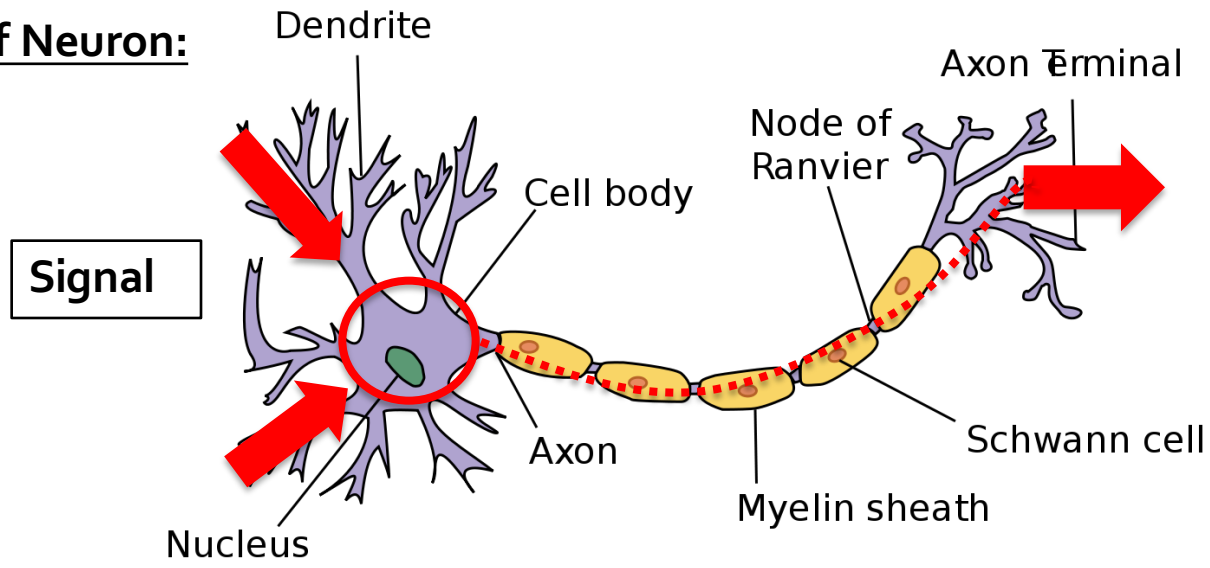
Structure of Neuron:



Neurons (cont.)

- A neuron has
  - A branching input (**dendrites**)
  - A branching output (the **axon**)
- The **information/signal** circulates from the dendrites to the axon via the **cell body**

Structure of Neuron:

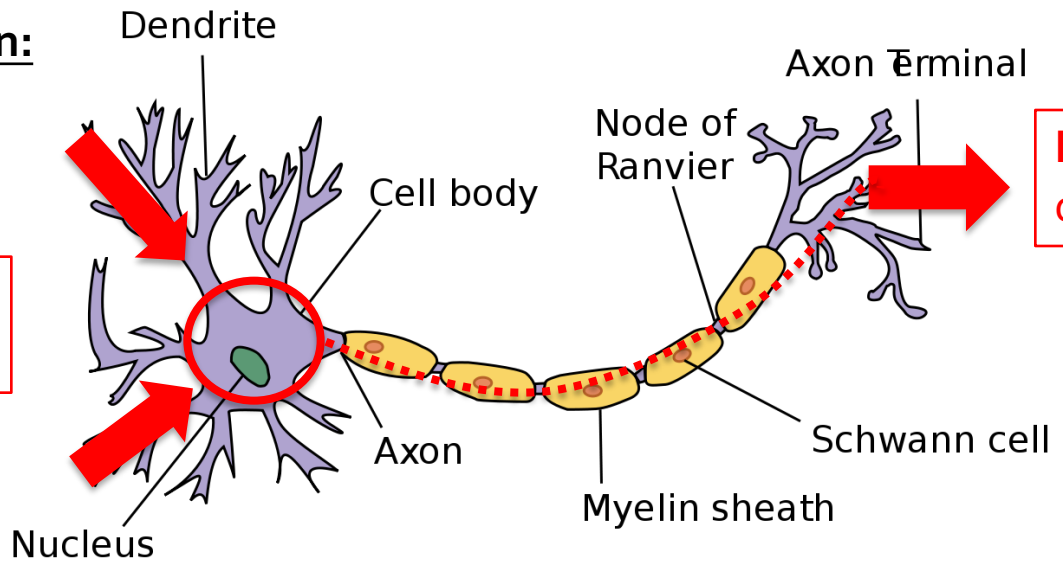


Neurons for Classification

- A neuron has
  - A branching input (**dendrites**)
  - A branching output (the **axon**)
- The ~~**information/signal**~~ circulates from the dendrites to the axon via the **cell body**

feature-label relationship

Structure of Neuron:

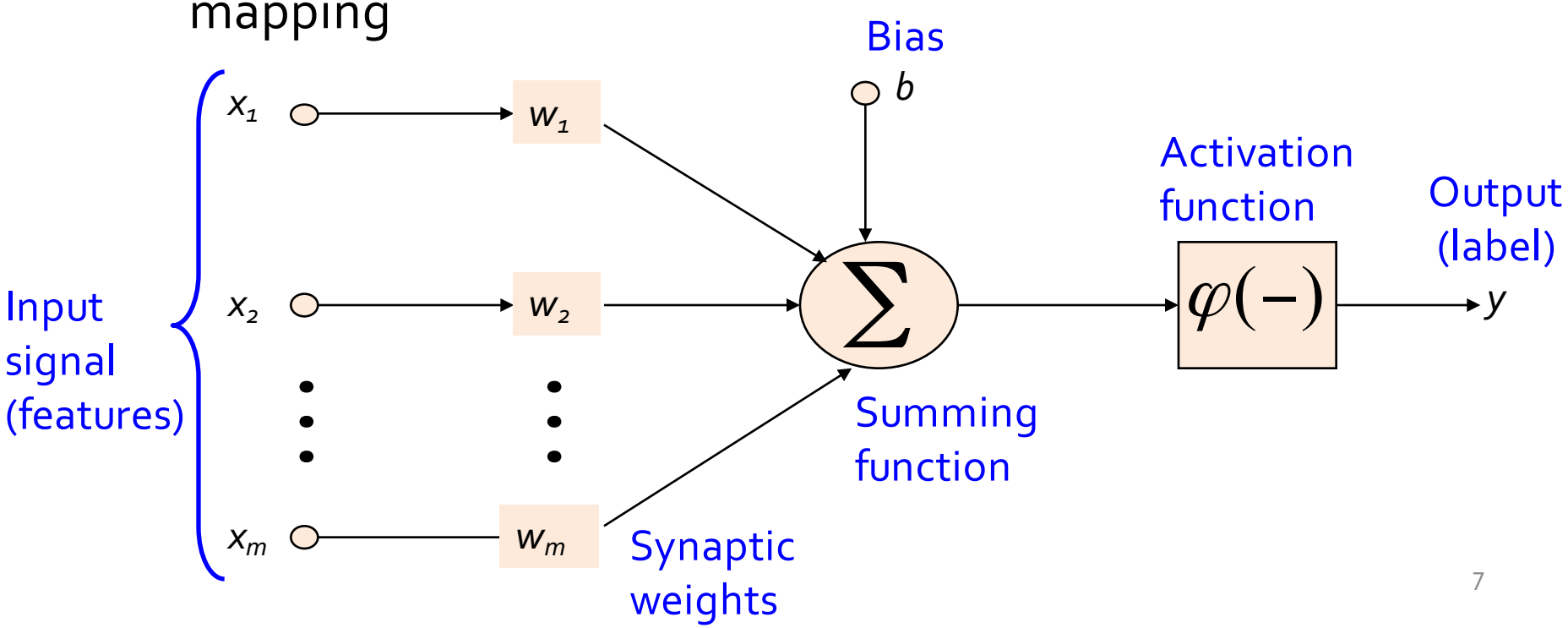


Feature values of a data point

Label of a data point

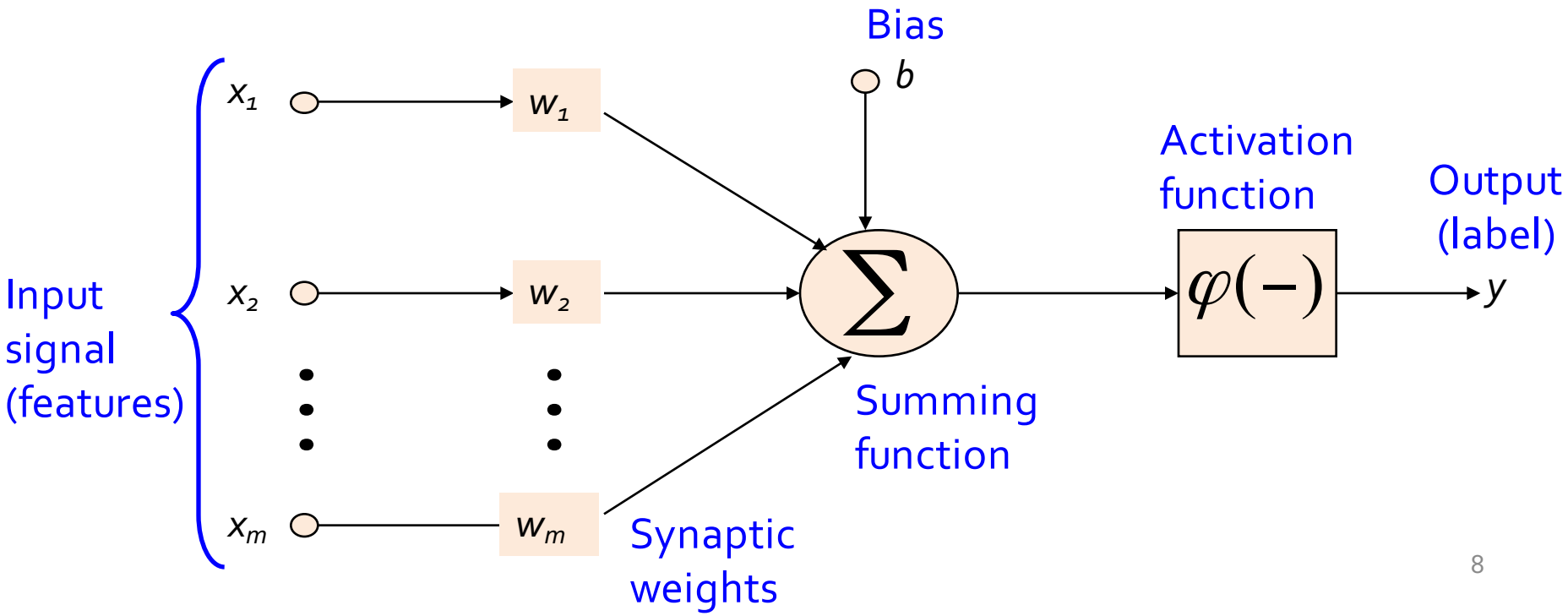
Artificial Neuron

- Definition: **Non-linear, parameterized** function with restricted output range
  - An *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping



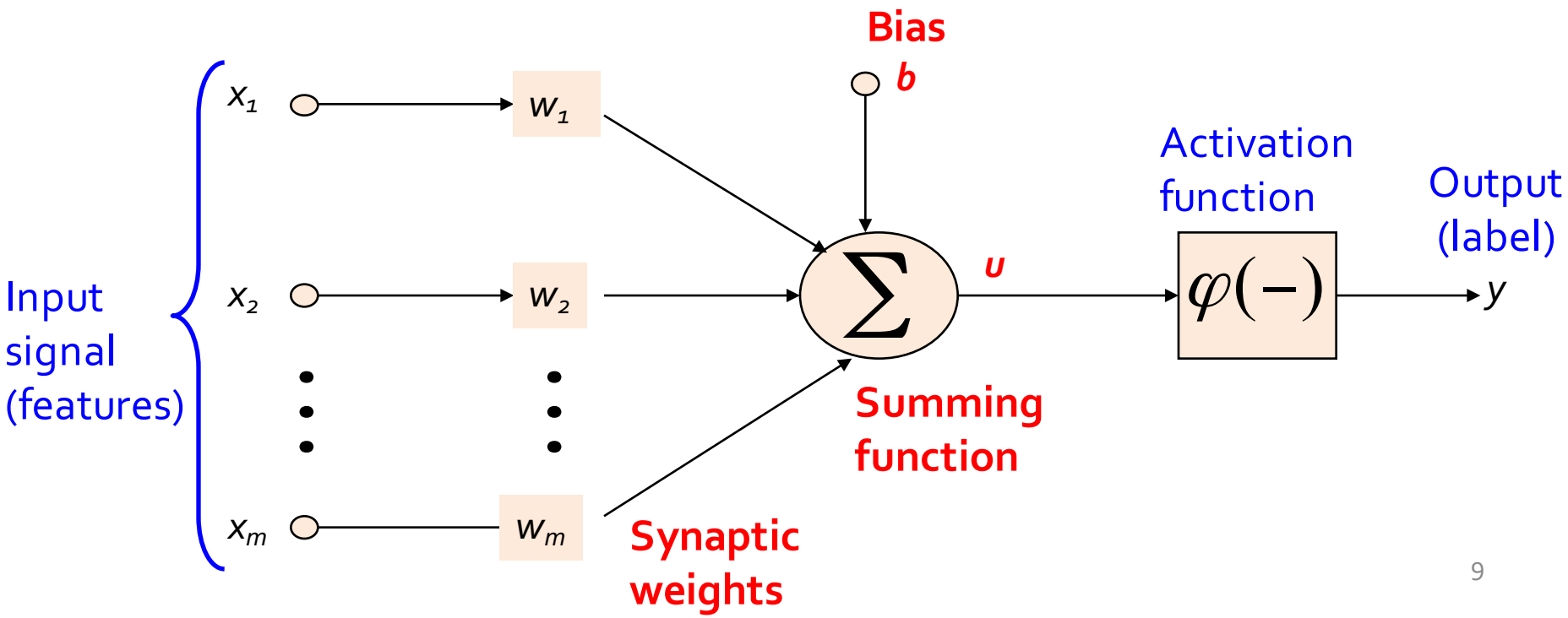
Artificial Neuron (cont.)

- It is the basic information processing **unit** of an ANN.
- The inputs are multiplied by their corresponding weights to form a *weighted sum*, which is added to the *bias* associated with unit. Then a *nonlinear activation function* is applied to it.



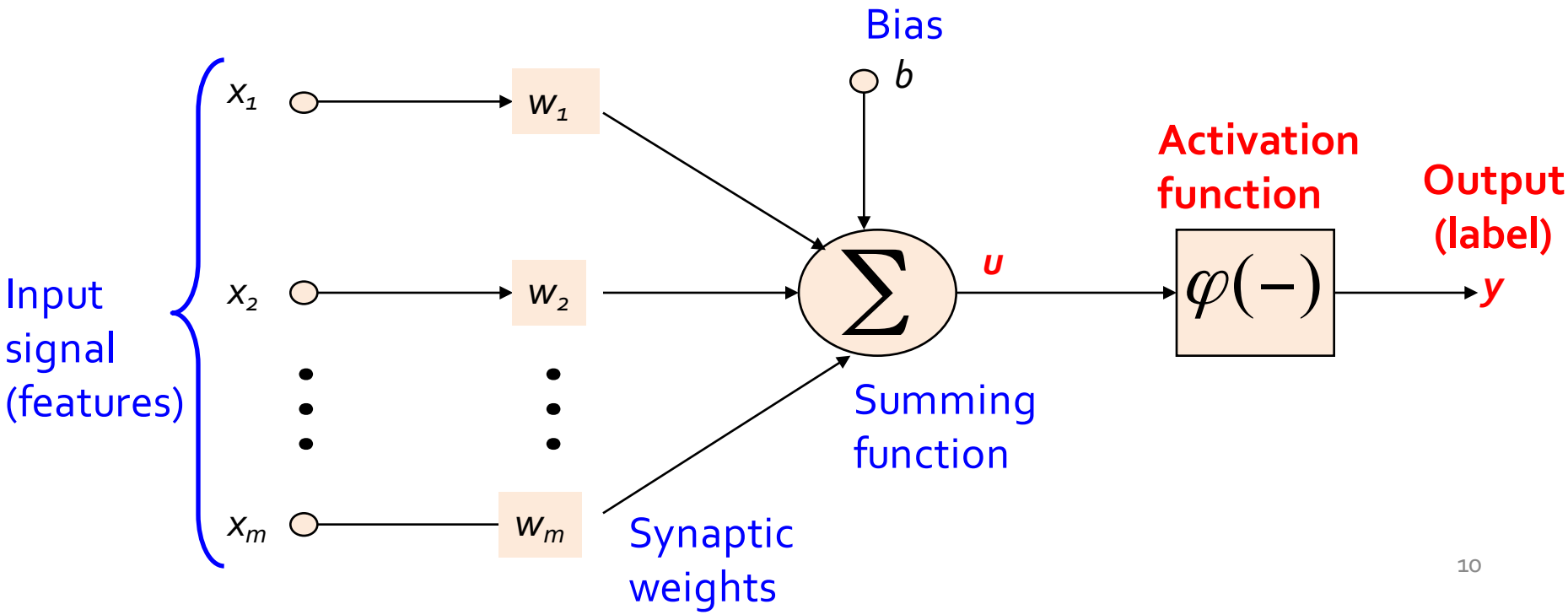
Artificial Neuron (cont.)

- A set of synapses or connecting links, each link characterized by a **weight**: $w_1, w_2, \ldots, w_m$
- A summing function (linear combiner) which computes the **weighted sum of the inputs** plus **bias**: $u = \sum_{i=1}^{m} w_i x_i + b$



Artificial Neuron (cont.)

- **Activation function** (squashing function) φ: $u \rightarrow y$ for limiting the amplitude of the output of the neuron
- E.g., **Logistic** functions (including **Sigmoid** function), **Hyperbolic tangent** function

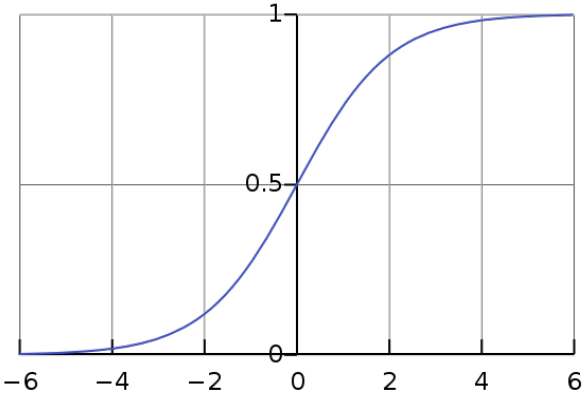


Activation Function: an "S" shape

- **Logistic functions**

$$f(x) = \frac{L}{1+e^{-k(x-x_0)}}$$

  - e = the natural logarithm base (Euler's number)
  - x_0 = the x-value of the sigmoid's midpoint
  - L = the curve's maximum value
  - K = the steepness of the curve



L = 1, k = 1, x_0 = 0

- **Sigmoid functions**

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{x}{2}\right)$$

  if x ∈ (-∞, +∞), then f(x) ∈ (0,1)

- **Hyperbolic tangent function**

$$f(x) = \tanh\left(\frac{x}{2}\right) = \frac{1-e^{-x}}{1+e^{-x}}$$

  if x ∈ (-∞, +∞), then f(x) ∈ (-1,1)

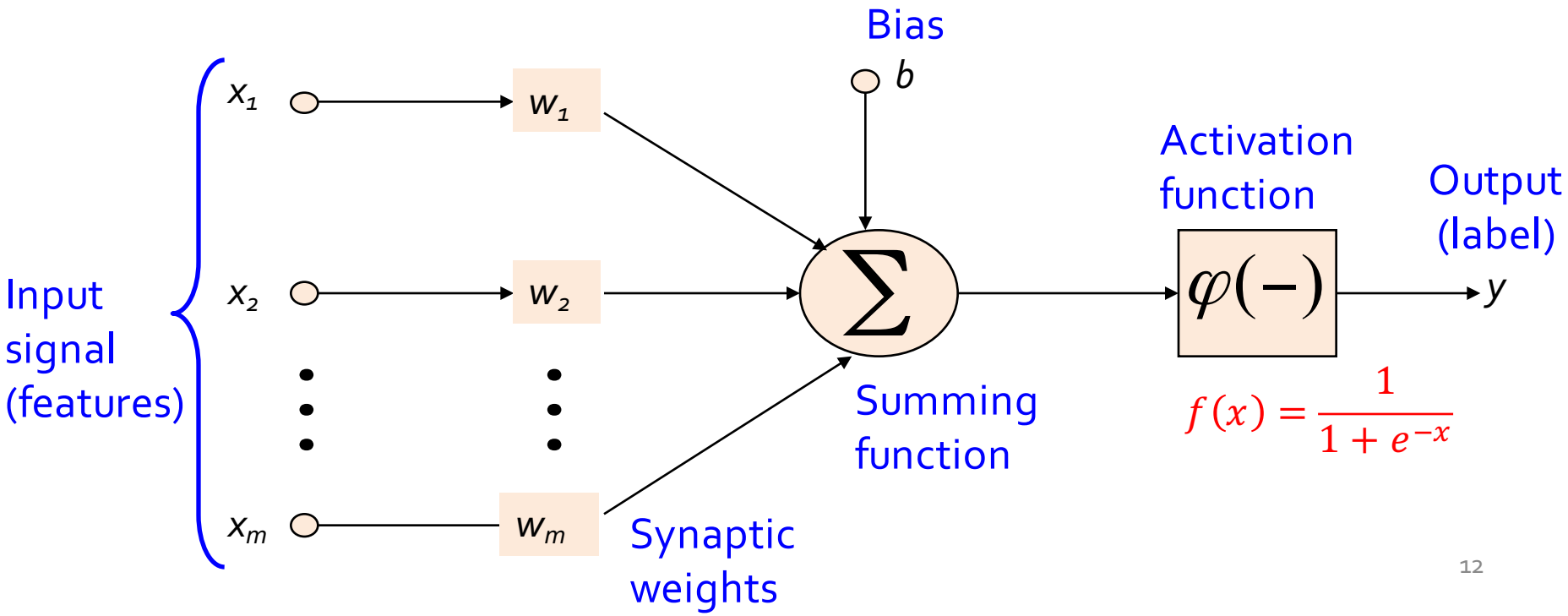
  if x ∈ [0, +∞), then f(x) ∈ [0,1)

Artificial Neuron: An Example

weights = [3.0, -0.5, -2.5, 1.5], b = -0.1

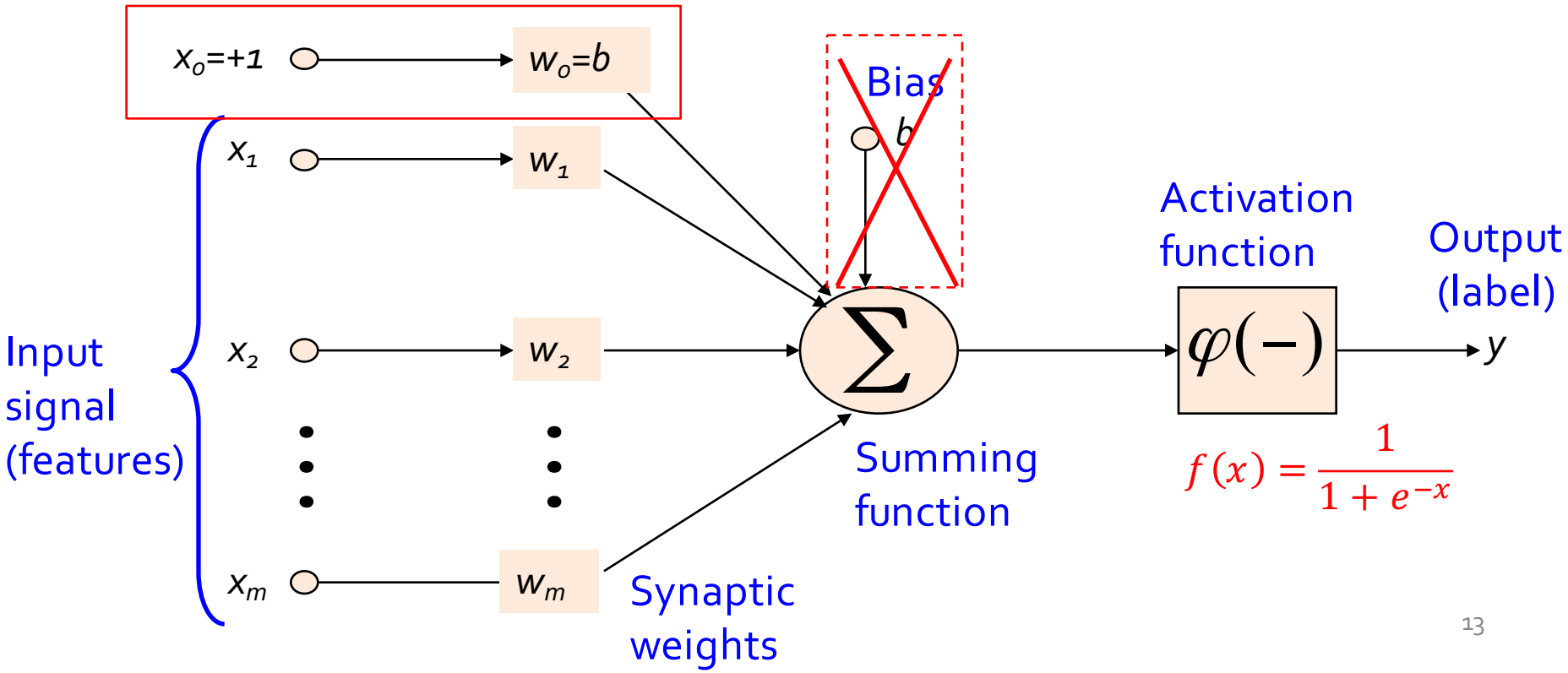
Is KDD?

	"data mining"	"web search"	"click through rate"	"pattern"	Label	**Output**
PID1	1	0	0	1	1	f(4.4)
PID2	0	1	1	0	0	f(-3.1)
PID3	1	1	0	1	1	f(3.9)



Bias

- $u = \sum_{i=1}^{m} w_i x_i + b \rightarrow u = \sum_{i=0}^{m} w_i x_i$
- $y = \varphi(u) = \varphi(\sum_{i=0}^{m} w_i x_i)$



Learning Process

- During the learning process the **weights** are modified in order to model the particular learning task correctly on the **training examples**.
- Given a data point (x_0=+1, x_1, …, x_m; y), use **gradient descent** to learn the weights (w_0, w_1, …, w_m) so that

$$\hat{y} = \varphi(u) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+\cdots+w_mx_m)}} \rightarrow y$$

This is actually Logistic Regression!

- Q: What is gradient descent? What is gradient?

Gradient

- What is gradient of a function *f(x)*?
  - Rate of change of function *f(x)* w.r.t. input x $\nabla f(x) = \frac{df}{dx}$
- What is gradient of a multiple variable function?
  - $f(x_1, x_2, x_3, \ldots, x_n)$

$$\nabla f(x_1, x_2, \ldots, x_n) = \left[\frac{\delta f}{\delta x_1}, \frac{\delta f}{\delta x_2}, \frac{\delta f}{\delta x_3}, \ldots, \frac{\delta f}{\delta x_n}\right]$$

- Example:

(1) $f(w_0, w_1, w_2, \ldots, w_m) = w_0x_0+w_1x_1+\ldots+w_mx_m$

$\nabla f(w_0, w_1, w_2, \ldots, w_m) = [x_0, x_1, x_2, \ldots, x_m]$

(2) $f(x) = \frac{1}{1+e^{-x}}$

$$\nabla f(x) = \frac{1}{1+e^{-x}} - \left(\frac{1}{1+e^{-x}}\right)^2 = f(x)(1-f(x))$$

Gradient Descent

- Gradient descent is a **first-order iterative optimization** algorithm for finding the minimum of a function.
- To find a **local minimum** of a function using gradient descent, one *takes steps proportional to the negative of the gradient* (or of the approximate gradient) of the function at the current point.
- If instead one *takes steps proportional to the positive of the gradient*, one approaches a **local maximum** of that function; the procedure is then known as gradient ascent.

Gradient Descent (cont.)

$$\nabla f(x_1, x_2, \ldots, x_n) = \left[\frac{\delta f}{\delta x_1}, \frac{\delta f}{\delta x_2}, \frac{\delta f}{\delta x_3}, \ldots, \frac{\delta f}{\delta x_n}\right]$$

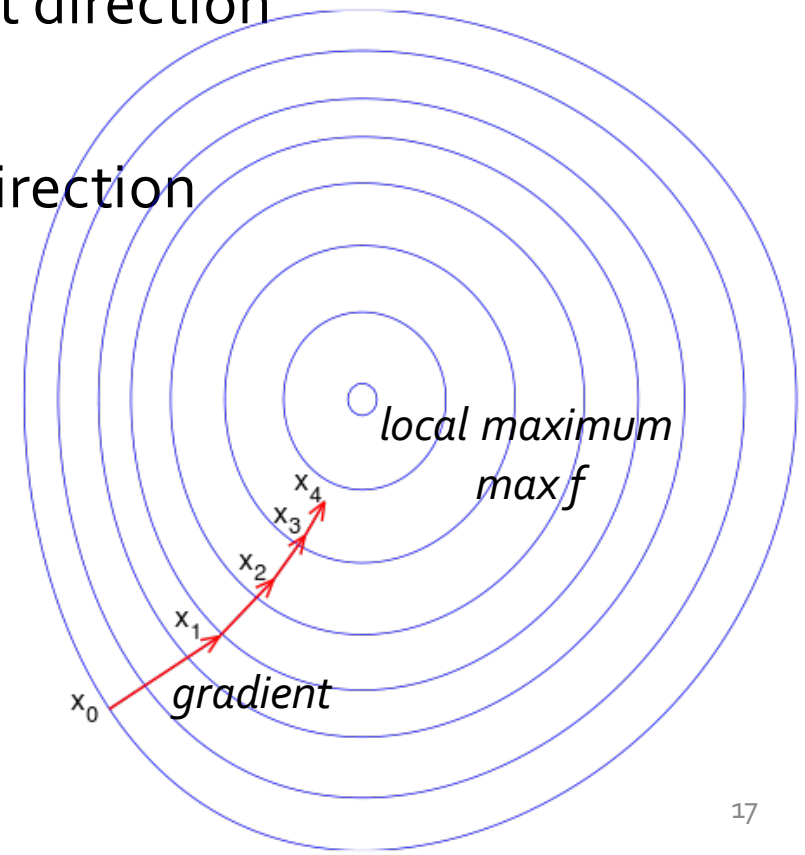
- Gradient: direction of steepest increase
- Magnitude: rate of change in that direction

Local minimum: go in the reverse direction

$$\mathbf{x} \leftarrow \mathbf{x} - R\nabla f(\mathbf{x})$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]$

$$x_i \leftarrow x_i - R\frac{\delta f}{\delta x_i}$$



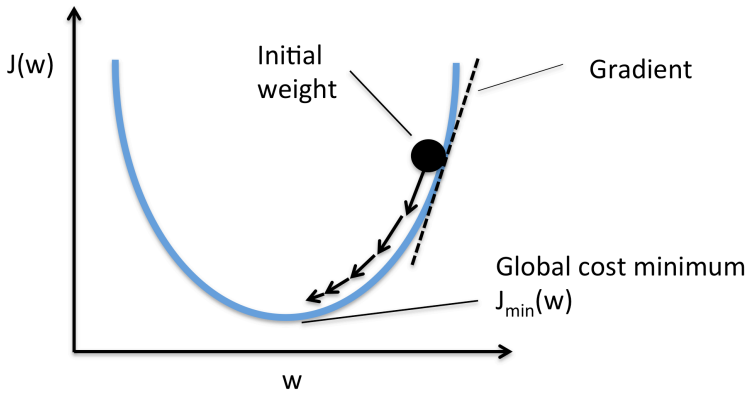
Gradient Descent for a Single Neuron

- Given a data point (x_0=+1, x_1, …, x_m; y), use **gradient descent** to learn the weights (w_0, w_1, …, w_m)

$$\hat{y} = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + \cdots + w_m x_m)}}$$

We want to minimize the loss function:

$$\min J(\boldsymbol{w}) = \frac{1}{D} \sum_{d=1}^{D} (y_d - \hat{y}_d)^2$$



$$\text{Q: } \nabla J(\boldsymbol{w}) = \frac{\delta J(\boldsymbol{w})}{\delta \boldsymbol{w}} = \left[\frac{\delta J}{\delta w_0}, \frac{\delta J}{\delta w_1}, \ldots, \frac{\delta J}{\delta w_m}\right] ?$$


ARE YOU READY
TO COMPUTE SOME PARTIAL DERIVATIVES ?
memegenerator.net

Partial Derivatives

Given $\min J(\boldsymbol{w}) = \frac{1}{D}\sum_{d=1}^{D}(y_d - \hat{y}_d)^2$,

where $\hat{y} = \frac{1}{1+e^{-(w_0x_0+w_1x_1+\cdots+w_mx_m)}}$

$$\frac{\delta J}{\delta w_i} = \frac{2}{D}\sum_{d=1}^{D}(y_d - \hat{y}_d)\cdot(-1)\cdot\hat{y}_d(1-\hat{y}_d)\cdot x_{d,i}$$

for $i = 0, 1, \ldots, m$

Example

Initialized weights = [0.1, 0.1, 0.1, 0.1, 0.1]

	x_0	"data mining"	"web search"	"click through rate"	"pattern"	Label y_d
PID1	+1	1	0	0	1	1
PID2	+1	0	1	1	0	0
PID3	+1	1	1	0	1	1

$$\frac{\delta J}{\delta w_i} = \frac{2}{D}\sum_{d=1}^{D}(y_d - \hat{y}_d)\cdot(-1)\cdot\hat{y}_d(1-\hat{y}_d)\cdot x_{d,i}$$

$$\hat{y}_1 = \varphi(0.1 + 0.1 + 0 + 0 + 0.1) = \varphi(0.3) = 0.574$$

$$\hat{y}_2 = \varphi(0.3) = 0.574$$

$$\hat{y}_3 = \varphi(0.4) = 0.599$$

$$(y_1 - \hat{y}_1)\cdot(-1)\cdot\hat{y}_1(1-\hat{y}_1) = (1-0.574)\cdot(-1)\cdot 0.574\cdot(1-0.574) = -0.104$$

$$(y_2 - \hat{y}_2)\cdot(-1)\cdot\hat{y}_2(1-\hat{y}_2) = (0-0.574)\cdot(-1)\cdot 0.574\cdot(1-0.574) = 0.140$$

$$(y_3 - \hat{y}_3)\cdot(-1)\cdot\hat{y}_3(1-\hat{y}_3) = (1-0.599)\cdot(-1)\cdot 0.599\cdot(1-0.599) = -0.096$$

Example (cont.)

Initialized weights = [0.1, 0.1, 0.1, 0.1, 0.1]

	x_0	"data mining"	"web search"	"click through rate"	"pattern"	Label y_d
PID1	+1	1	0	0	1	1
PID2	+1	0	1	1	0	0
PID3	+1	1	1	0	1	1

$$\frac{\delta J}{\delta w_i} = \frac{2}{D}\sum_{d=1}^{D}(y_d - \hat{y}_d)\cdot(-1)\cdot\hat{y}_d(1-\hat{y}_d)\cdot x_{d,i}$$

$\frac{\delta J}{\delta w_0}$ = 2/3 * (-0.104*1+0.140*1-0.096*1) = -0.04 < 0

$\frac{\delta J}{\delta w_1}$ = 2/3 * (-0.104*1+0.140*0-0.096*1) = -0.133 < 0

$\frac{\delta J}{\delta w_2}$ = 2/3 * (-0.104*0+0.140*1-0.096*1) = 0.029 > 0

$\frac{\delta J}{\delta w_3}$ = 2/3 * (-0.104*0+0.140*1-0.096*0) = 0.093 > 0

$\frac{\delta J}{\delta w_4}$ = 2/3 * (-0.104*1+0.140*0-0.096*1) = -0.133 < 0

If the learning rate (magnitude) is 1.0, new weights = [0.14, 0.233, 0.071, 0.007, 0.233]

Hidden Layer of Neurons

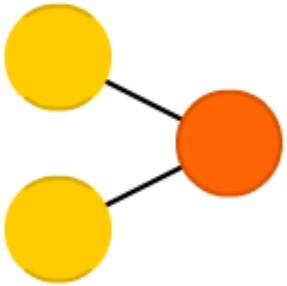
Input: data points of n=2 features
Output: one label

Input Cell

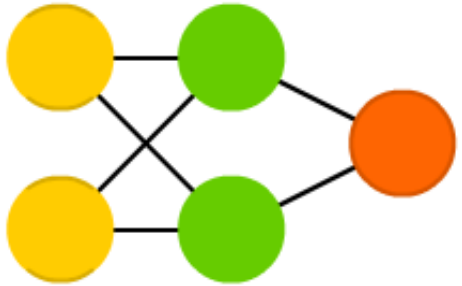
Output Cell

Hidden Cell

Perceptron (P)



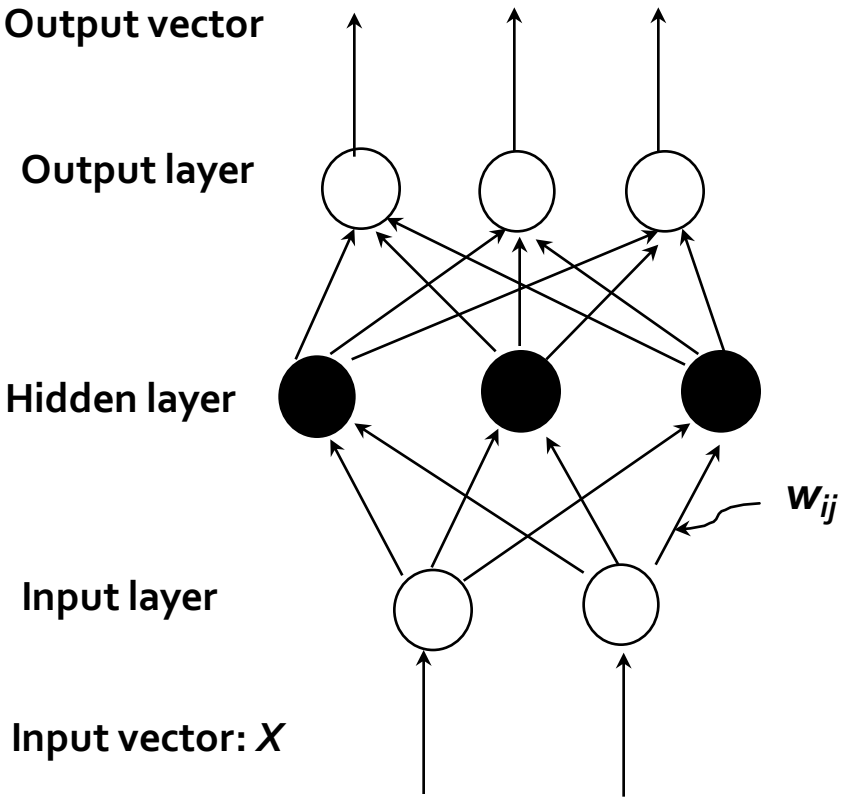
Feed Forward (FF)

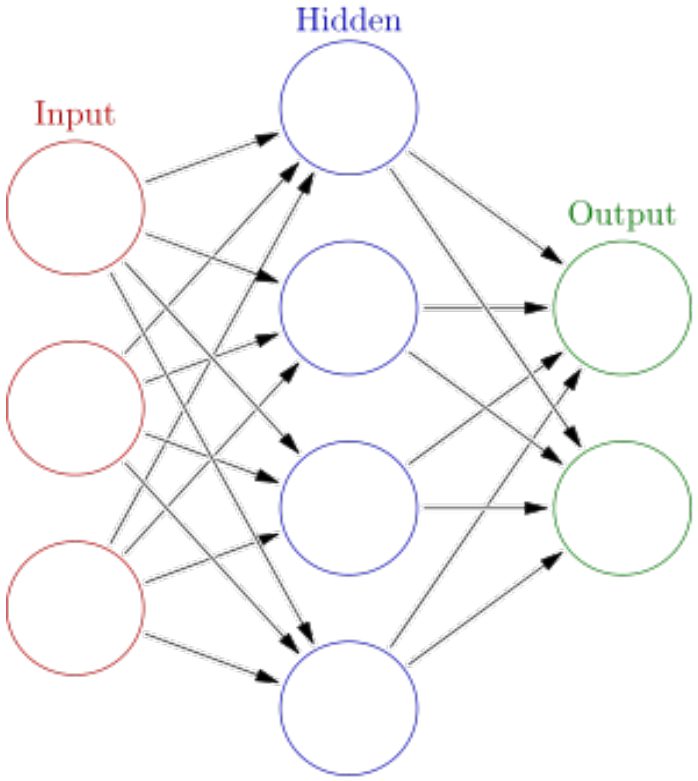


Each hidden cell (neuron)'s output is a new feature (that is a non-linear function result of old features) as input for the output cell.

Can we have multiple output cells? Yes!

A Three-Layer Feed-Forward Neural Network





Q: How many weight parameters?

This is a 2-3-3 ANN.

2*3+3*3 = 15

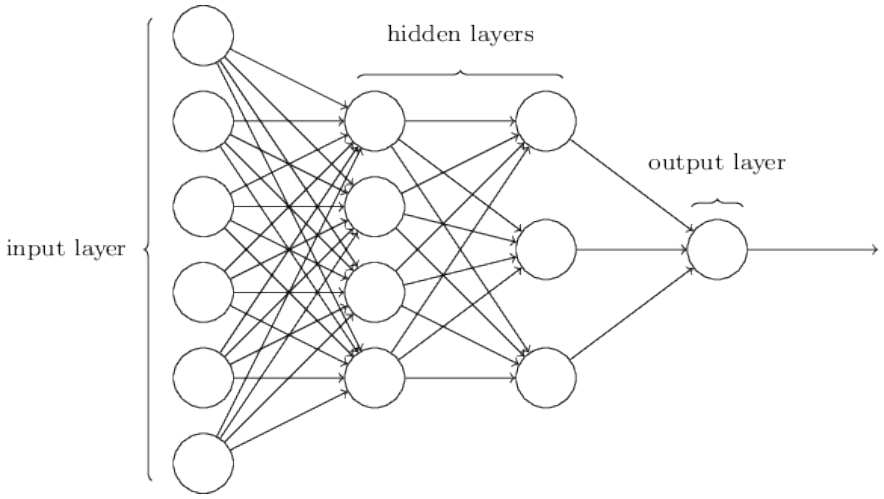
This is a 3-4-2 ANN.

3*4+4*2 = 20

Q': How many weight parameters did we have in an SVM?

Can we have multiple hidden layers? Yes!

A Two-Hidden-Layer Feed-Forward Neural Network



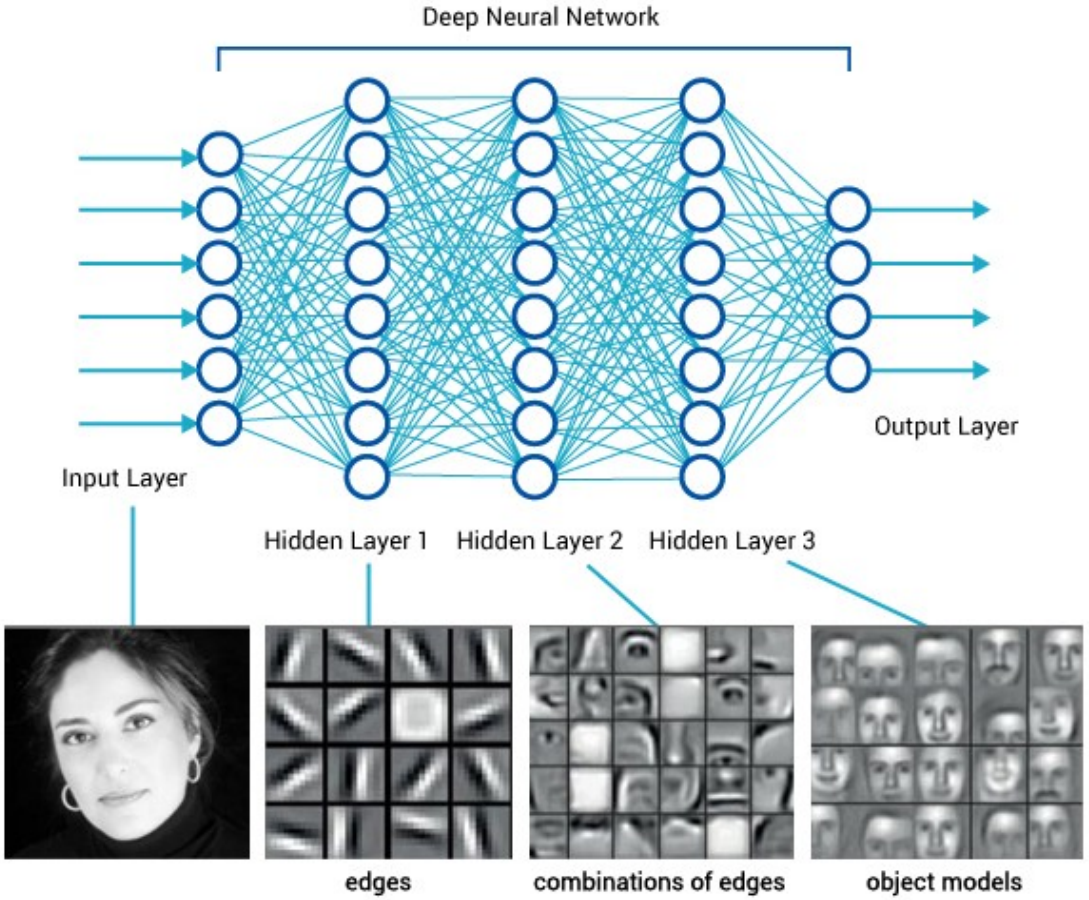
Suppose we have one hidden layer of 7 neurons. It is a 6-7-1 network.
We have 6*7+7*1 = 49 weight parameters to learn.

Here we have two hidden layers (4+3). It is a 6-4-3-1 network.
We have 6*4+4*3+3*1 = 39 weight parameters to learn.

When the network's architecture becomes more complicated, the number of parameters may **NOT** become more, but the performance may become better:
It can model more ***complicated non-linear distributions***.

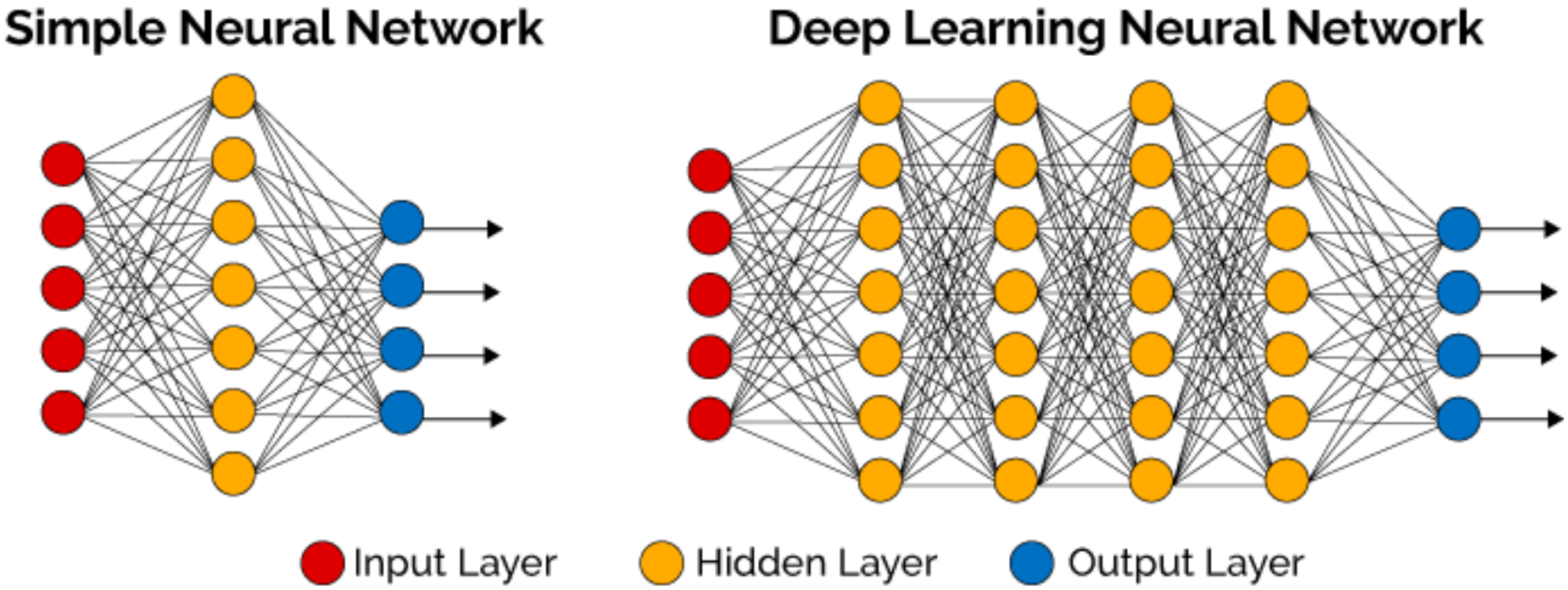
Deep Neural Networks

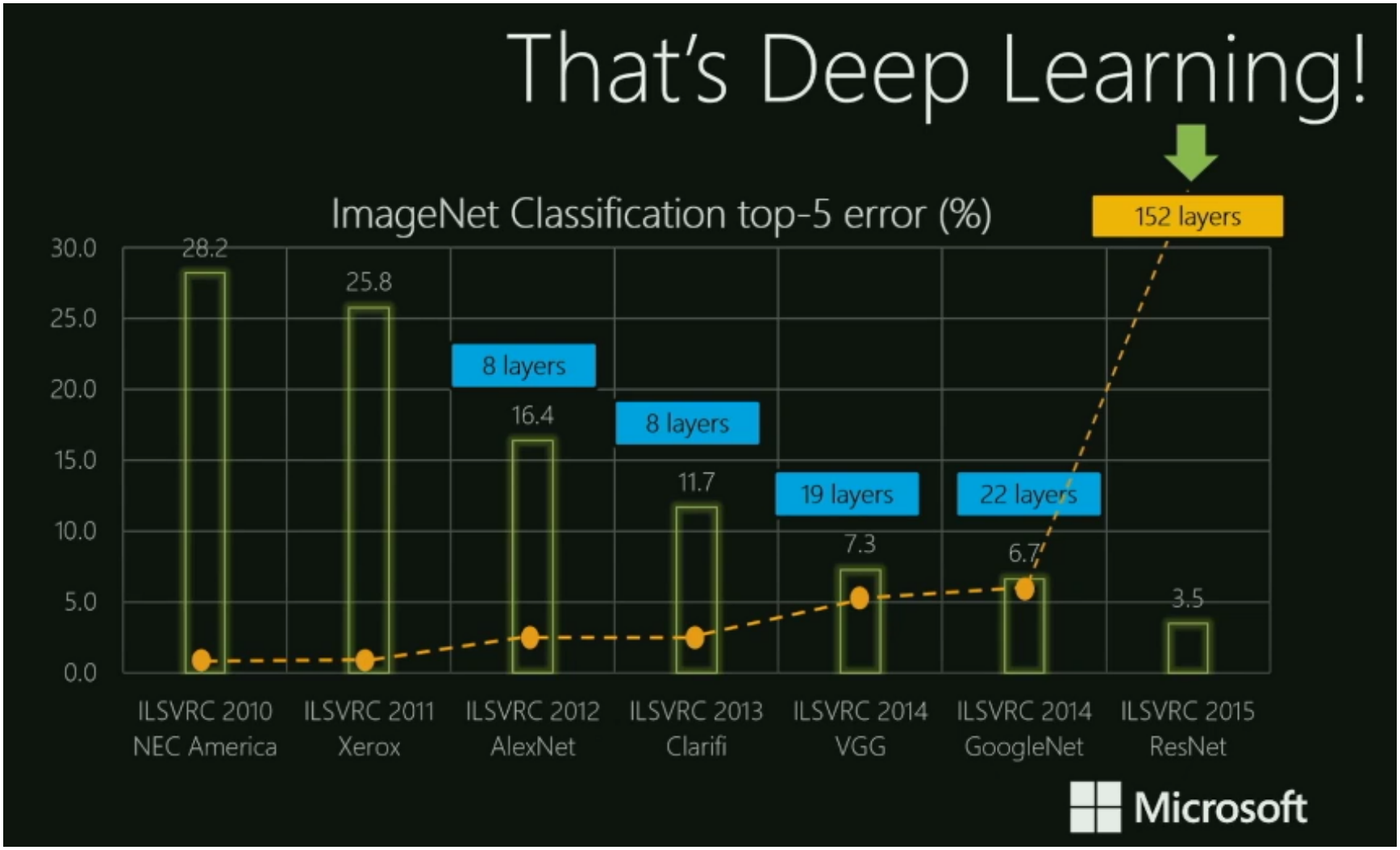
- One of the main deep learning architectures.



Deep Neural Networks (cont.)

- *How many layers is a DEEP network architecture?*




That's Deep Learning!
ImageNet Classification top-5 error (%)
152 layers
30.0
25.0
20.0
15.0
10.0
5.0
0.0
28.2
25.8
8 layers
16.4
8 layers
11.7
19 layers
7.3
22 layers
6.7
3.5
ILSVRC 2010
NEC America
ILSVRC 2011
Xerox
ILSVRC 2012
AlexNet
ILSVRC 2013
Clarifi
ILSVRC 2014
VGG
ILSVRC 2014
GoogleNet
ILSVRC 2015
ResNet
Microsoft

Summary: Components of a Feed-Forward Neural Network

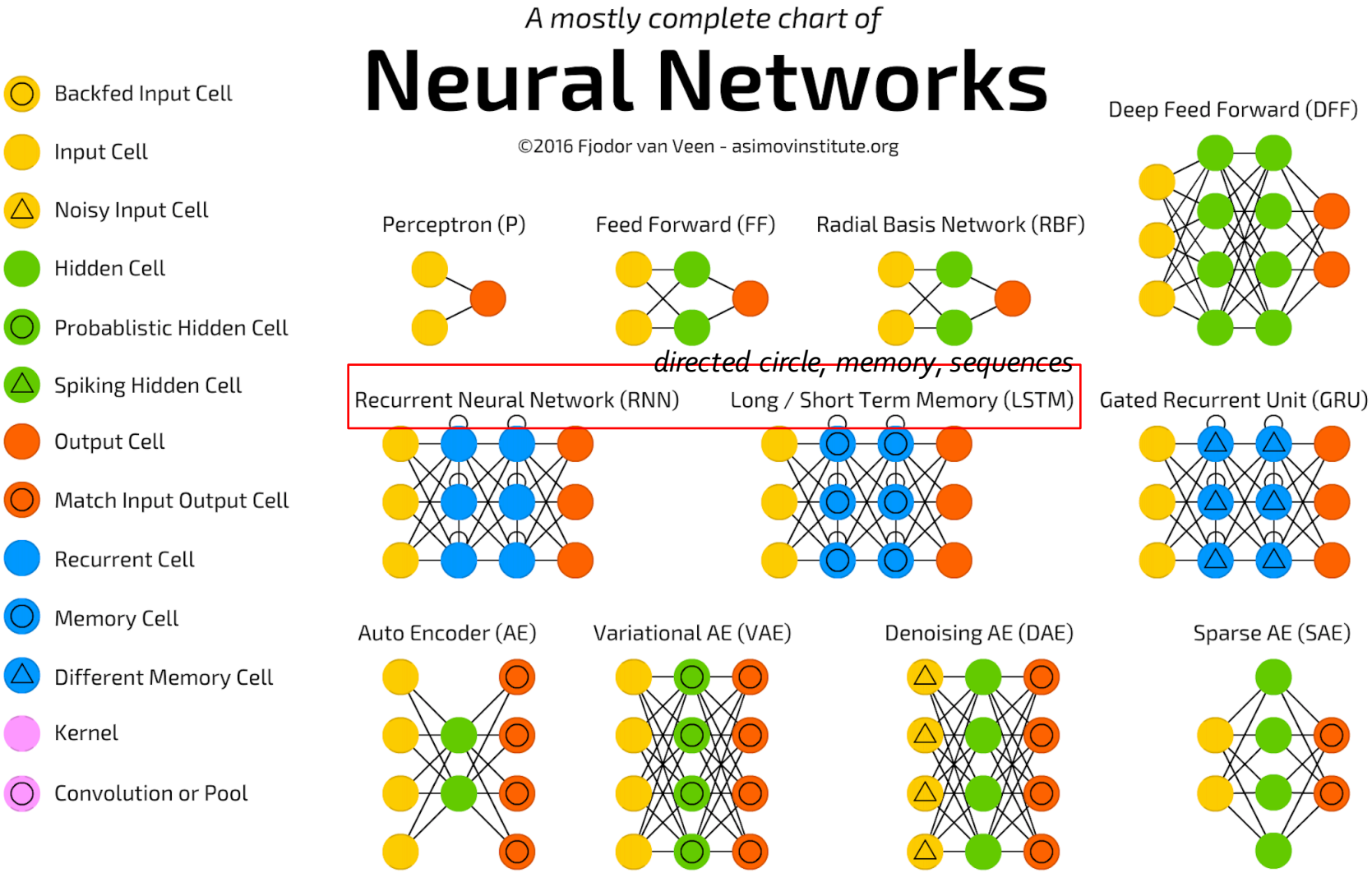
- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of **hidden layers** is arbitrary
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward**: None of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**
  - Given **enough** hidden units and **enough** training samples (**and enough what?**), they can closely approximate ***any*** function

Limitations of Neural Networks

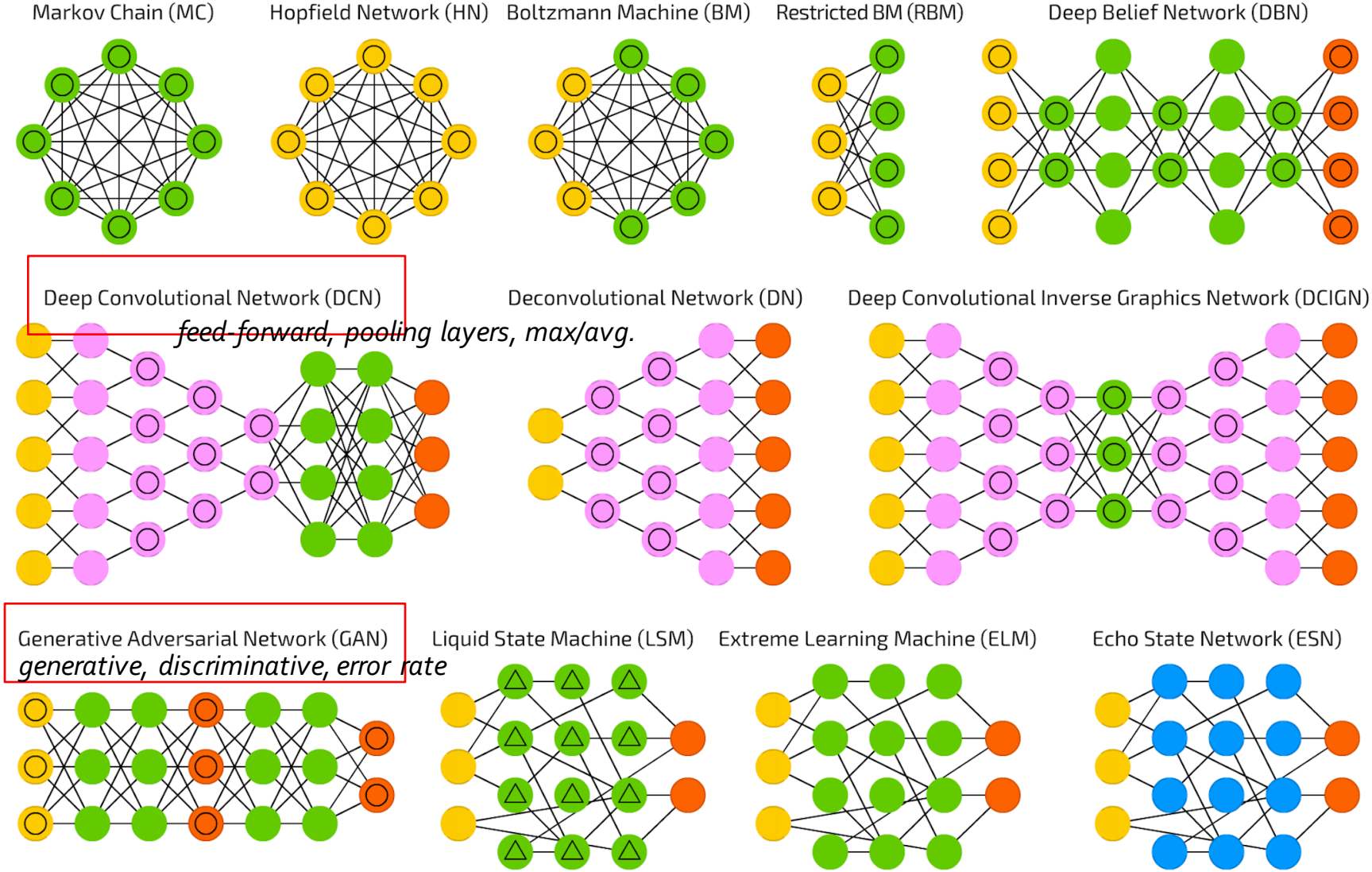
Random initialization + **densely connected networks** lead to:

- High cost
  - Training the entire neural network is to train **all the interconnected logistic regressions**.
- Difficult to train as the number of hidden layers increases
  - Recall that logistic regression is trained by **gradient descent** that is looking for **local optimum**.
- Stuck in local optima
  - The objective function of the neural network is **usually not convex**.
  - The random initialization does not guarantee starting from the proximity of global optima.
- Difficult to determine the network topology
  - Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a **different network topology** or **a different set of initial weights.**

Beyond Input/Hidden/Output Cells and Beyond Feed-Forward Network Architectures



Continue...


Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
feed-forward, pooling layers, max/avg.
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
generative, discriminative, error rate
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)

More Details on RNNs, CNNs, GANs?

- Please register Machine Learning course by Prof. Nitesh Chawla in Spring'18.
- Our Data Science is a prerequisite for his class.

Deep Learning and Alchemy?