Applied Machine Learning

Deep Learning Basic

Computer Science, Fall 2022

Instructor: Xuhong Zhang

Learning Objectives

- Understand the basics of a real neuron
- Understand how real neurons inspired artificial neurons
- Understand the components of a neural network
- Begin to understand the perceptron learning rule

Neural Networks in the Wild

- Applications where Neural Networks have been used

OnMSFT

Microsoft's Custom Neural Voice technology goes out of ...

The technology enables developers to create personalized voices using their audio data and deep neural networks (DNN). "The real ...

4 weeks ago

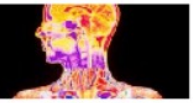
Health Imaging

Deep learning detects common shoulder pain on x-rays—a ...

Subscribe to Health Imaging News. The free newsletter covering the top medical imaging headlines.

1 week ago

IMA INTERNATIONAL JOURNAL OF IMAGING SYSTEMS AND TECHNOLOGY



RESEARCH ARTICLE | Free Access

Convolutional capsule network for COVID-19 detection using radiography images

Shamik Tiwari, Anurag Jain

First published: 02 March 2021 | https://doi.org/10.1002/ima.22566

SECTIONS PDF TOOLS SHARE

Journal of Petroleum Technology

Deep Neural Network Trains To Pinpoint Microseismic Events ...

Deep Neural Network Trains To Pinpoint Microseismic Events in Real Time. Artificial intelligence is opening new ways to analyze data from ...

2 days ago

HPCwire

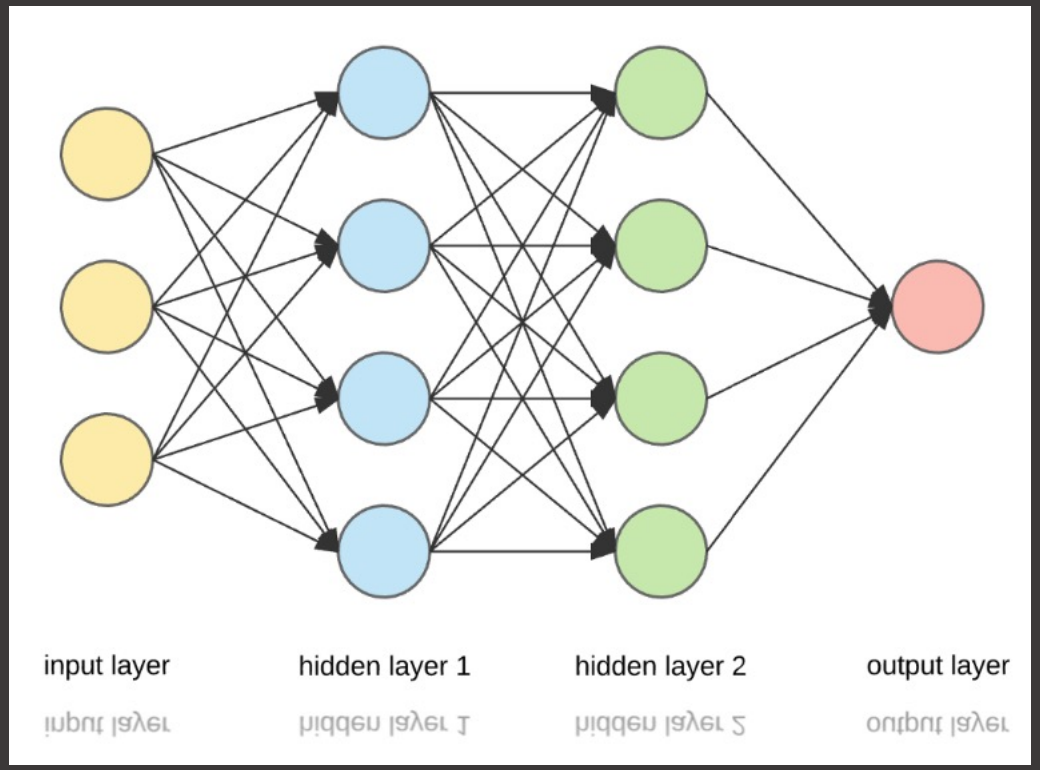
Can Deep Learning Replace Numerical Weather Prediction?

The authors suggest that any eventual deep learning replacement for an NWP would likely consist of several neural networks trained on subsets ...

2 hours ago

Neural Networks

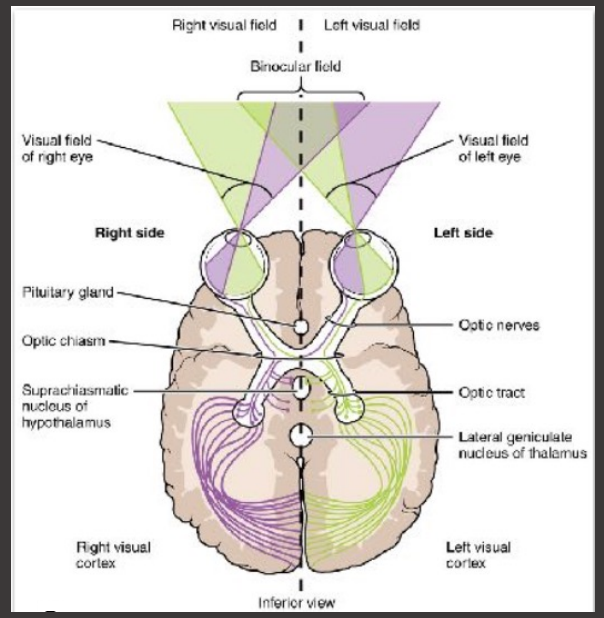
- A network of artificial neurons
- A network that can be used to map input features to an output (regression or classification)
- This artificial neural network (ANN) is used to learn complex non-linear mappings
- Their main goal is to model some of the decision making that we do rather easily

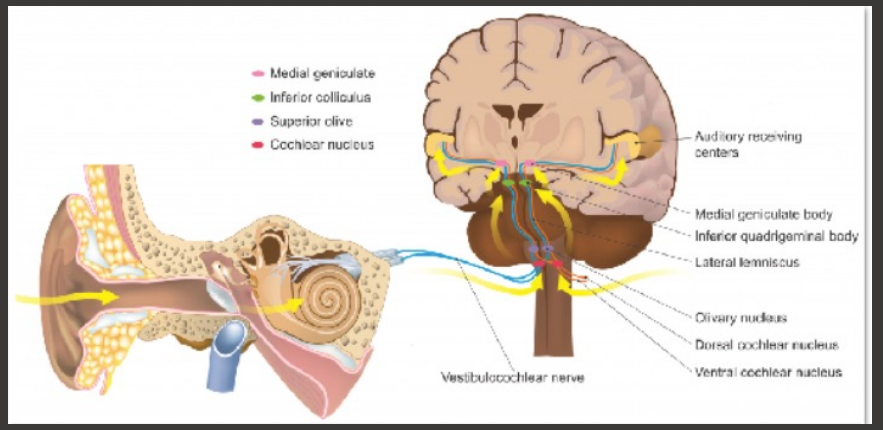


The Human Brain

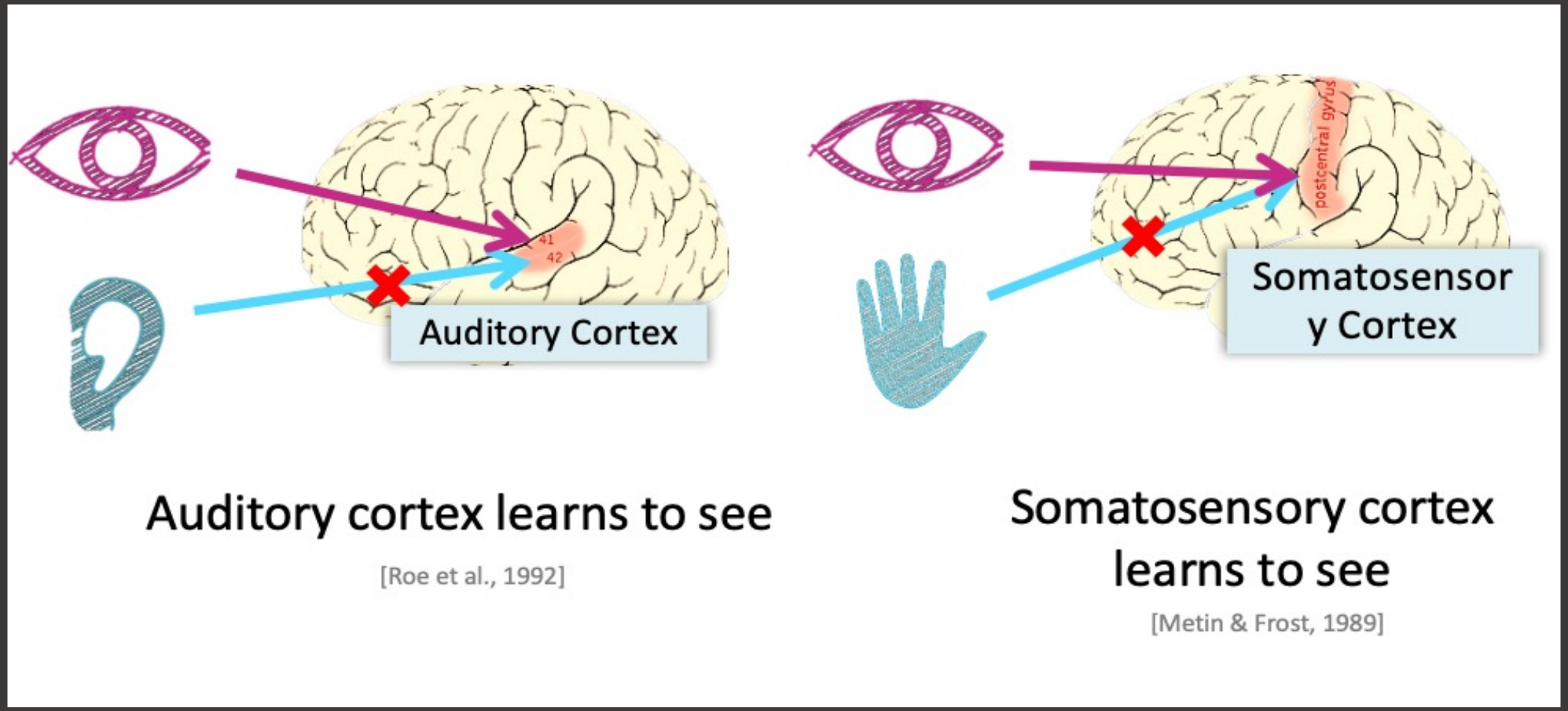
- Neural Networks are (loosely) modeled after our brain
- We make decisions (e.g. classification/regression) from sensory input data (i.e. see, hear, smell, touch, taste)

  - Object detection
  - Voice recognition
  - Distant calculation
  - Etc.

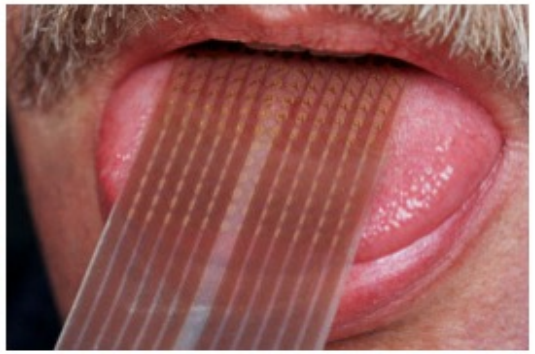




The "One Learning Algorithm"

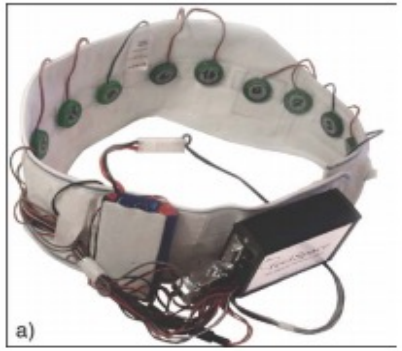


Sensor Representations in the Brain



Seeing with your tongue

Human echolocation (sonar)



Haptic belt: Direction sense

Implanting a 3rd eye

[BrainPort; Welsh & Blasch, 1997; Nagel et al., 2005; Constantine-Paton & Law, 2009]

Slide Credit: Andrew Ng

Comparison of computing power

INFORMATION CIRCA 2012	Computer	Human Brain
Computation Units	10-core Xeon: 10^9 Gates	10^{11} Neurons
Storage Units	10^9 bits RAM, 10^{12} bits disk	10^{11} neurons, 10^{14} synapses
Cycle time	10^{-9} sec	10^{-3} sec
Bandwidth	10^9 bits/sec	10^{14} bits/sec

- Computers are way faster than neurons…
- But there are a lot more neurons than we can reasonably model in modern digital computers, and they all fire in parallel
- Neural networks are designed to be massively parallel
- The brain is effectively a billion times faster

Real Neurons (or Nerve Cell)

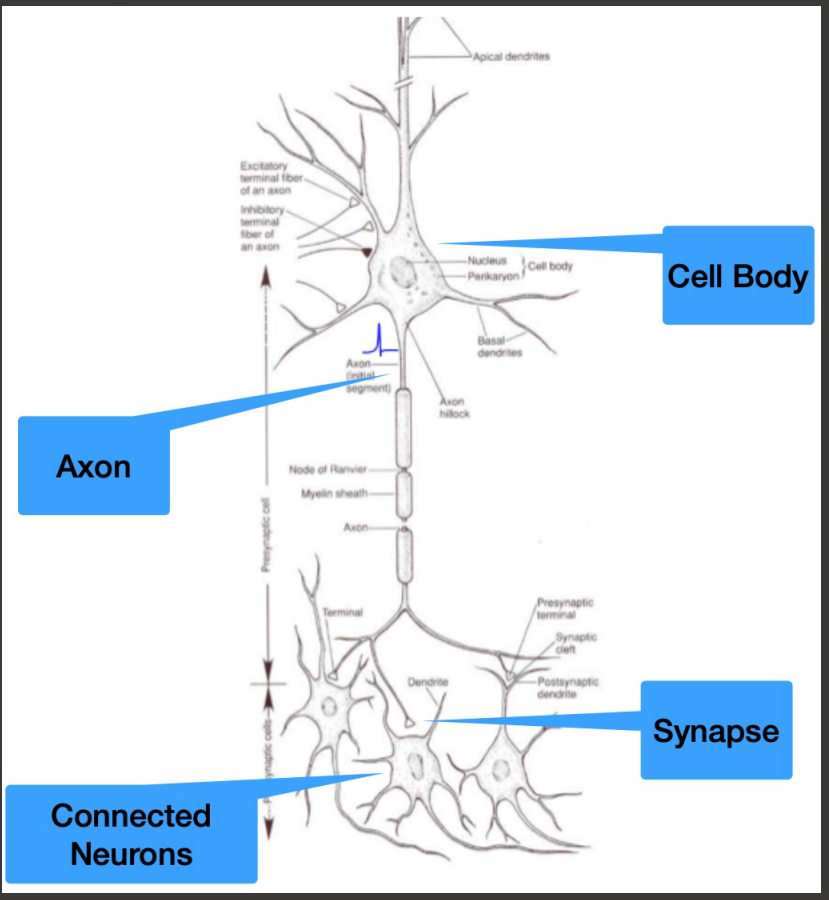
- A neuron has a tree like structure. Main components of a neuron are:
  - *Cell body*: generates a pulse (an impulse)
  - *Axon*: impulse travels through this channel
  - *Synapse (or leaf)*: impulse terminates here. Connects to other neurons.

- A neuron is considered non-existent unless it generates an impulse. Thus it is considered digital (binary), all or none.

Real Neurons (or Nerve Cell)

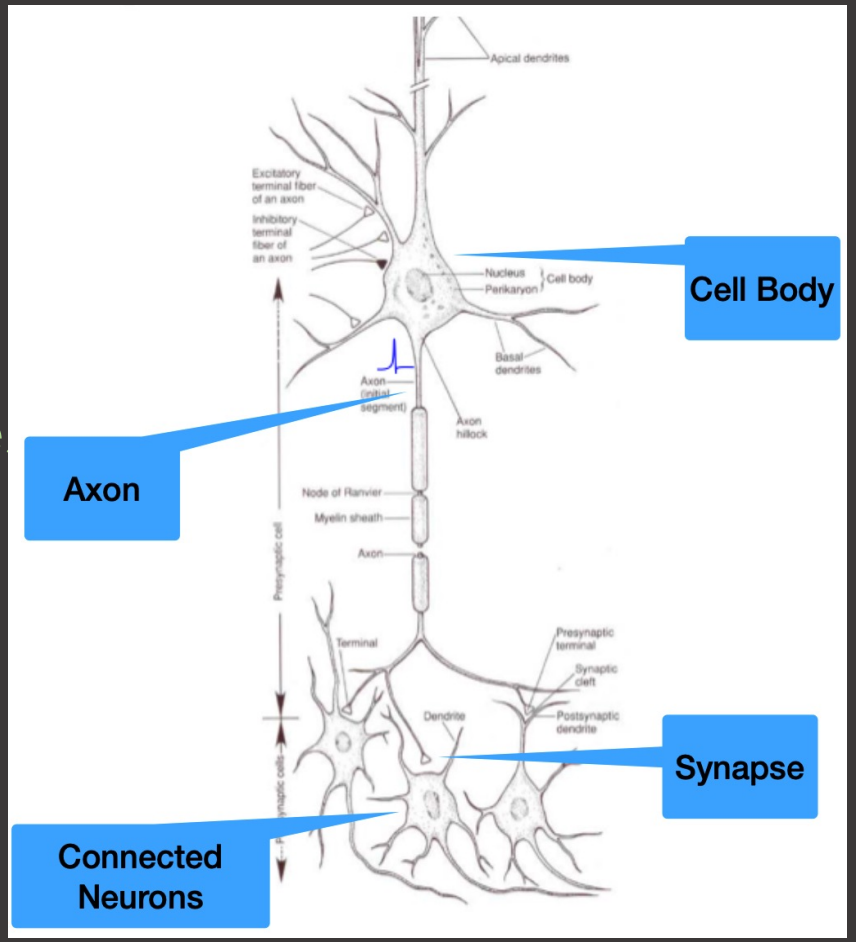
- Our brain consists of roughly 10^{11} neurons, each connected, on average, to 10^4 other neurons. Our brain is an interconnected network.

- It is estimated that there are about 10^{81} electrons in the observable universe.



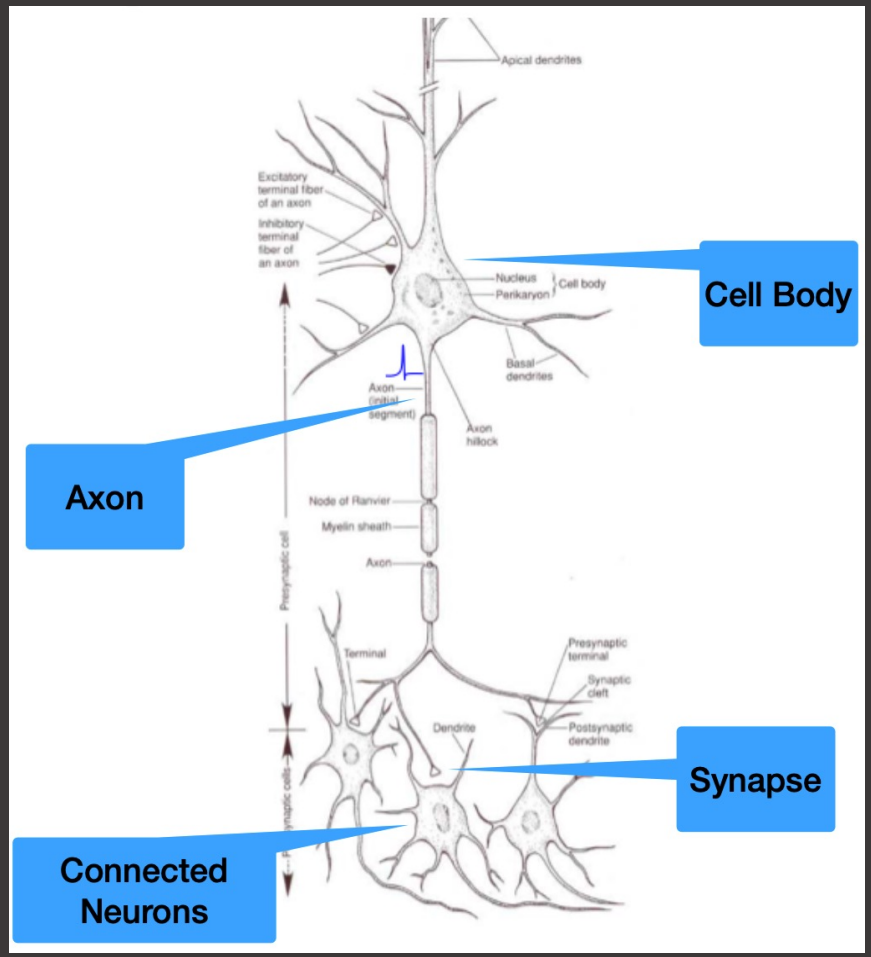
Real Neurons (or Nerve Cell)

- There are two types of neuron contacts (synapses):
  - *Excitatory (positive):* receiving neuron increases activity
  - *Inhibitory (negative)*: receiving neuron decrease activity
  - Neuron can only be excitatory or inhibitory
- The synaptic contact is ***plastic***, meaning it can be modified. This is essential for learning



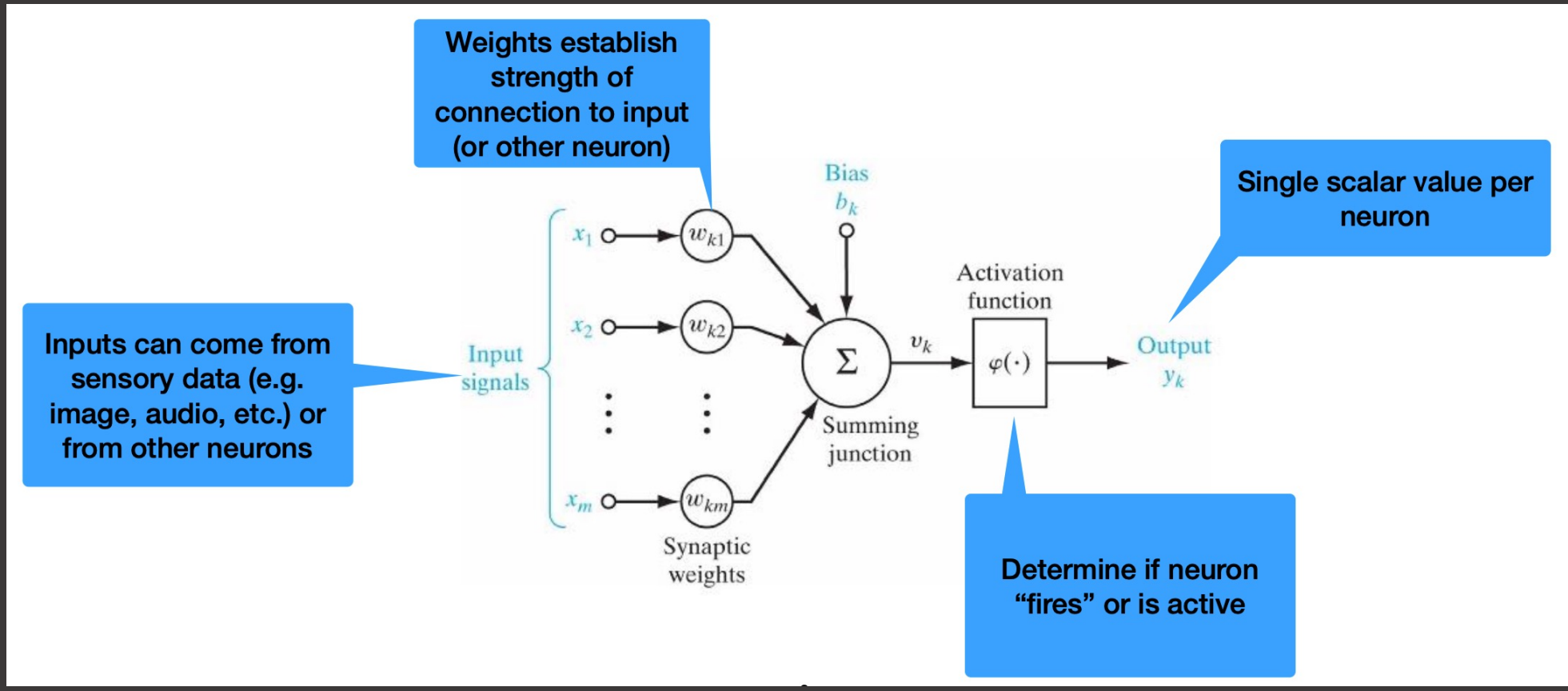
More Terminology

- ***Neuron***: unit or node
- ***Synapse***: connection or architecture
- ***Axon***: impulse travel channel
- ***Synaptic weight***: connection strength (either positive or negative)

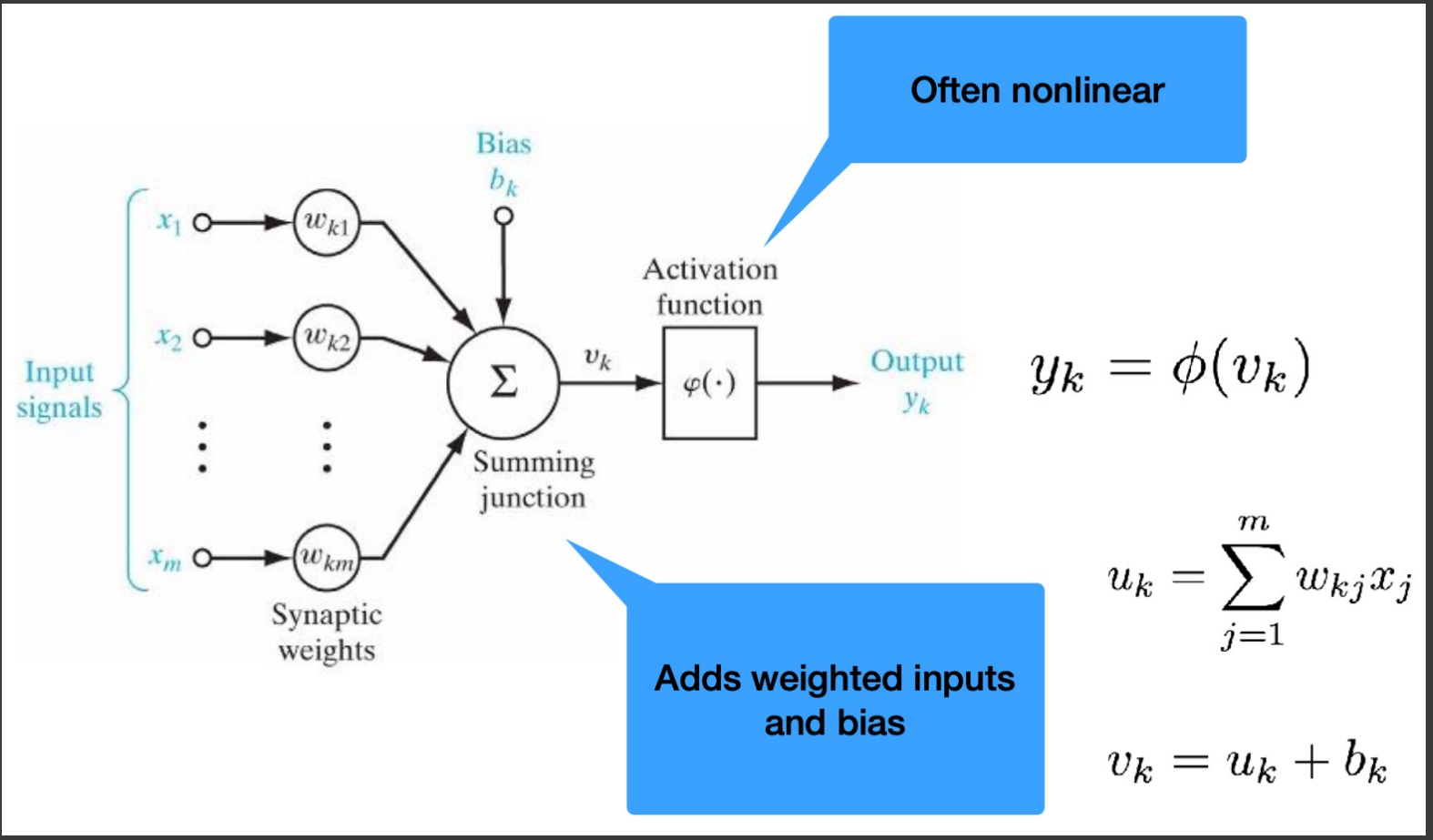


Artificial Neuron

- Components of an Artificial Neuron



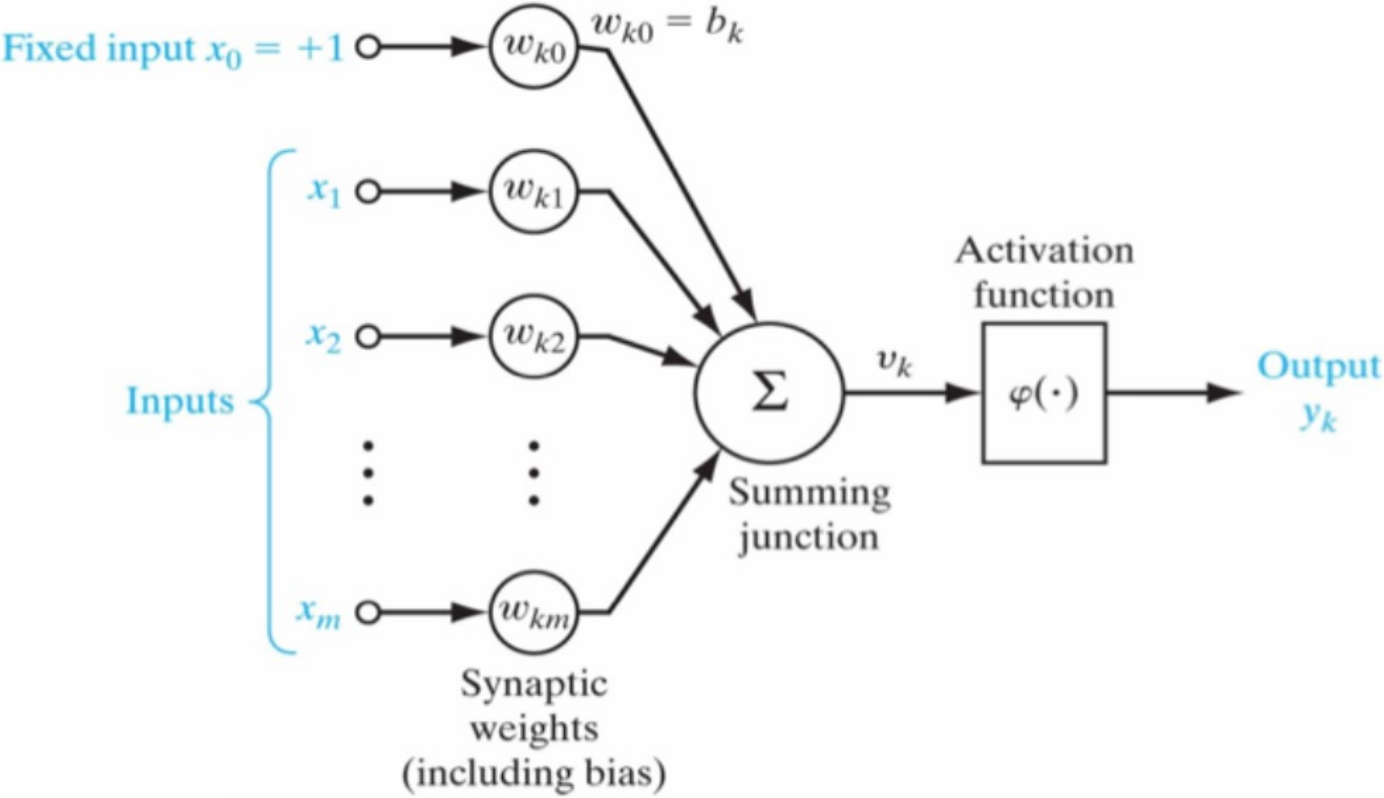
Artificial Neuron



Output $y_k = \phi(v_k)$:

- ϕ: activation function
- $v_k = u_k + b_k$: Activation potential, b_k is bias
- $u_k = \sum_{j=1}^{m} w_{kj} x_j$: Adder, weighted sum, linear combiner

Another way of including bias



- As with linear regression, the bias can be seen as another input
  - Set $x_0 = 1$ and $w_{k0} = b_k$
- So we have
  - $v_k = \sum_{j=0}^{m} w_{kj} x_j = w_k^T x$
  - $x = [x_0, x_1, \dots, x_m]^T$
  - $w_k = [w_{k0}, w_{k1}, \dots, w_{km}]^T$

McCulloch-Pitts Neuron Model

- McCulloch-Pitts Networks (1943) are the first class of abstract computing machines: finite-state automata. Finite-state automata can compute any logic (Boolean) function
- Uses bipolar inputs and generates bipolar outputs
- Uses specific activation function

Bipolar input : $x_i \in \{-1, +1\}$

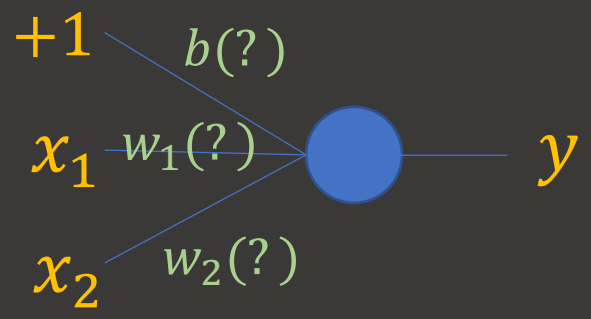
Output : $y = \phi(\sum_{i=1}^{m} w_i x_i + b)$

Activation function : $\phi(v) = \begin{cases} 1, & if\ v \geq 0 \\ -1, & if\ v < 0 \end{cases}$

McCulloch-Pitts Neuron Model

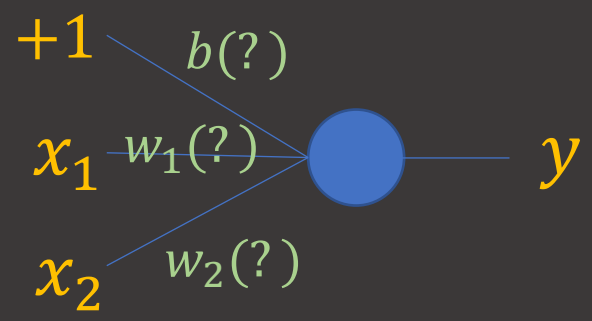
- Given the McCulloch-Pitts Model, define the weights that produce the correct output (y) for each input pair (x_1, x_2)

OR Function



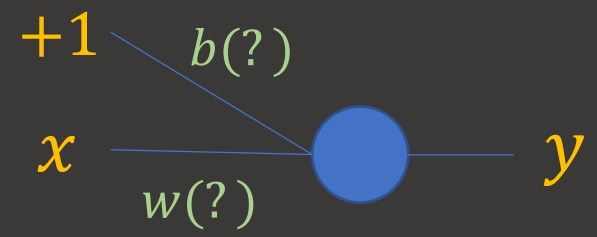
x1	x2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

AND Function



x1	x2	y
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

NOT Function



x	y
-1	1
1	-1

Modified Version of McCulloch-Pitts Neuron: Perceptron Neuron

- McCulloch-Pitts neuron are limited by their inputs: Perceptrons address this by using ***real-valued inputs***
- Perceptrons are also used for *learning*, whereas McCullonch-Pitts neurons are not
- Invented by Rosenblatt in 1957
- Useful for *binary classification*

Perceptron Neuron Model

Real-valued input : $x_i \in \mathbb{R}$

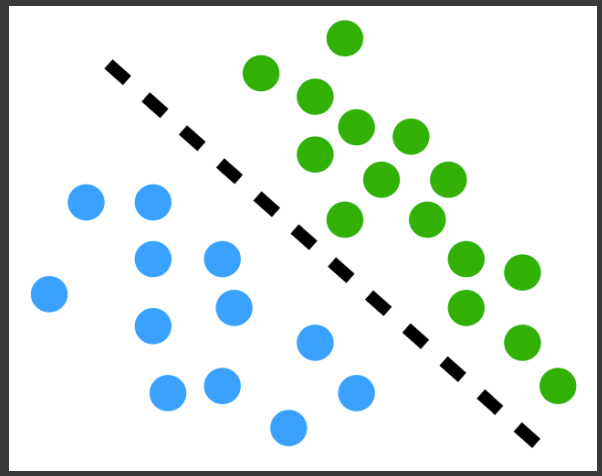
Output : $y = \phi(v)$

Activation potential : $v = \sum_{i=1}^{m} w_i x_i + b$

Activation function : $\phi(v) = \begin{cases} 1, & if\ v \geq 0 \\ -1, & if\ v < 0 \end{cases}$

Perceptrons for Classification

- Goal: Find values for weight vector (and bias), so that output of the perceptron matches the desired output, for each input
  - For bipolar outputs, decide between -1 or 1
  - The weights (and bias) result in a decision boundary (e.g. discriminative approach), where features above the boundary are classified as one class, while features below the boundary are classified as the other

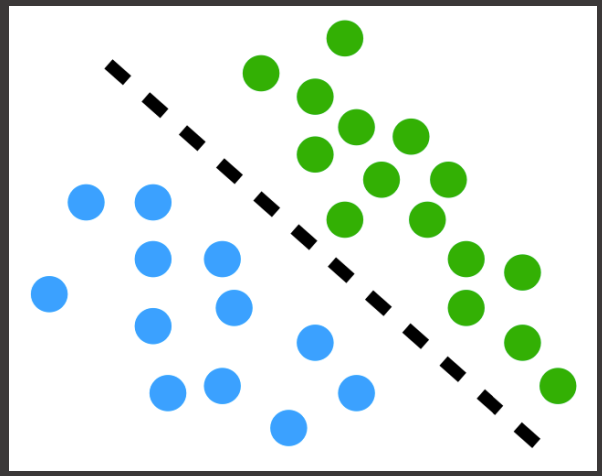


Perceptrons for Classification

- The ***decision boundary*** of a perceptron for a given w:

$$g(x) = \sum_{i=1}^{m} w_i x_i + b$$

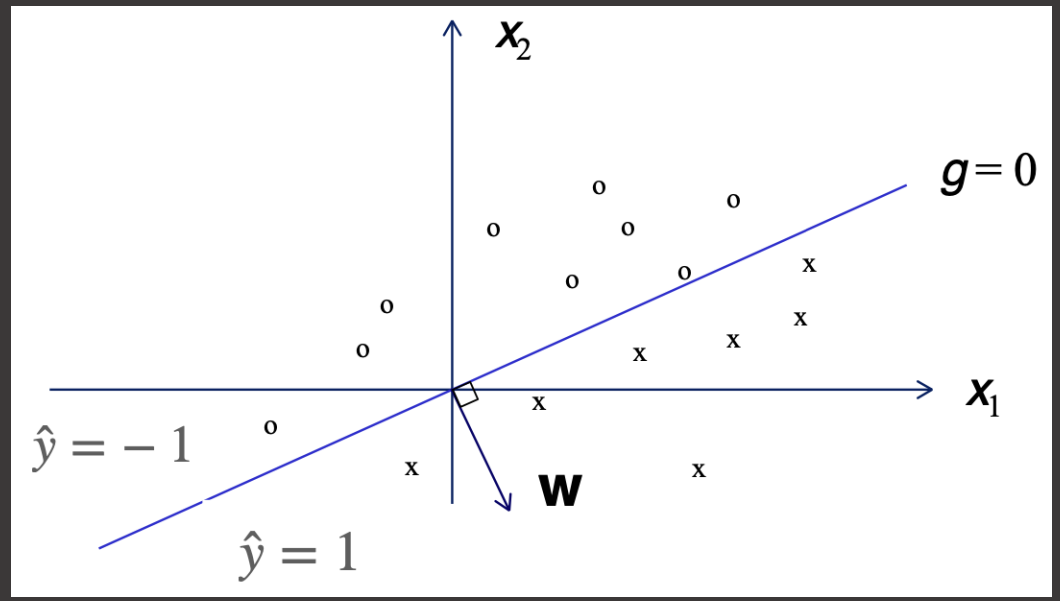
- $g(x)$ is also called the discriminant function for the perceptron, and it is linear function of input x. Hence, it is a linear discriminant function



Decision Boundary

- For an m-dimensional input space, the decision boundary is an $(m-1)$---dimensional hyperplane perpendicular to w.
  - By definition, the weight vector is orthogonal to the decision boundary.

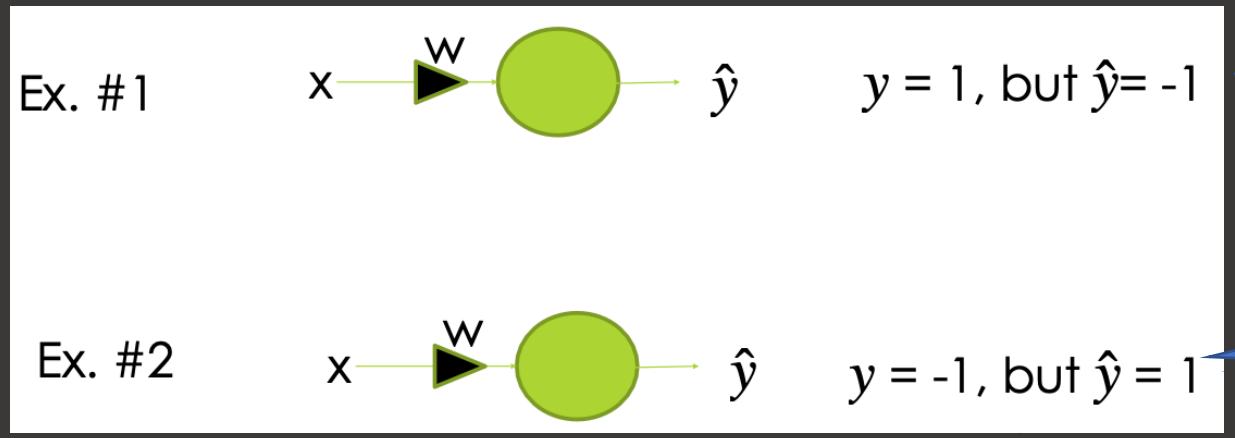
- The hyperplane separates the input space into two halves
  - One half having $\hat{y} = 1$
  - The other half having $\hat{y} = -1$
  - When $b = 0$, the hyperplane goes through the origin



Perceptron Learning Rule

- Perceptron Learning Rule:
  - Strengthen the weight if the neuron fails to fire when it should have fired
  - Weaken the weight if the neuron fires when it should not have fired

- Example: Single perceptron with single weight



If want $\hat{y}$ to be 1, but the output is -1, then this neuron should be strengthened.
In other words, the weight is too small.

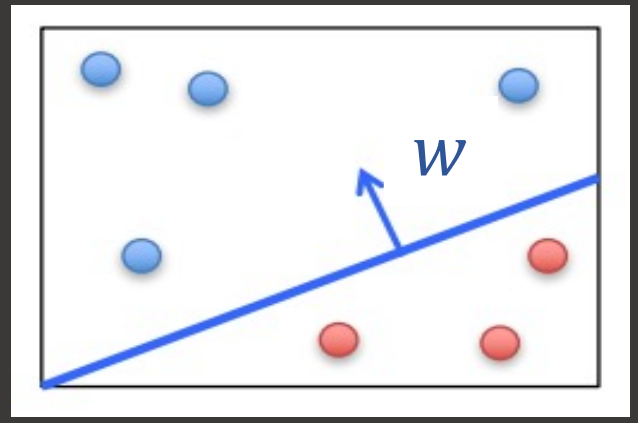
This neuron should be weakened. The weight should decrease.

Linear Classification: The Perceptron

- **Linear Classifiers:** represent decision boundary by hyperplane

- Binary Classification: $\{+1, -1\}$
- Given data point $x^T = [1, x_1, \ldots, x_d]$ and $w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$

- We have $sign(z) = \begin{cases} 1 & if\ z \geq 0 \\ -1 & if\ z < 0 \end{cases}$
  - Note that $h(x) = \begin{cases} \text{sign}(w^T x) > 0 \Rightarrow y = +1 \\ \text{sign}(w^T x) < 0 \Rightarrow y = -1 \end{cases}$



Perceptron

- Motivation:
  - Recall what we talk about the high dimensional space.
- Assumption:
  - The data is linearly separatable.
  - So we can always find some hyperplane to separate the data.
- Hyperplane:
  - $\mathcal{H} = \{x\colon\ w^T \cdot x + w_0 \cdot 1 = 0\}$ or $\mathcal{H} = \{x\colon\ w^T \cdot x = 0\}$

Perceptron

- $h(x) = \begin{cases} \text{sign}(w^T x) > 0 \Rightarrow y = +1 \\ \text{sign}(w^T x) < 0 \Rightarrow y = -1 \end{cases}$
- Steps:
  - Initialize the weights w with 0 or small random values.
  - For training data point $x^{(i)}$, calculate $h_w(x^{(i)})$.
  - Update the weights w:

$$w_j \leftarrow w_j - \frac{\alpha}{2}\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

    - If the prediction matches the label, make no change
    - Otherwise, adjust w

Perceptron

- Re-write as $w_j \leftarrow w_j + \alpha y^{(i)} x_j^{(i)}$, only upon misclassification
  - Can eliminate α, since its only effect is to scale w by a constant, which doesn't effect performance

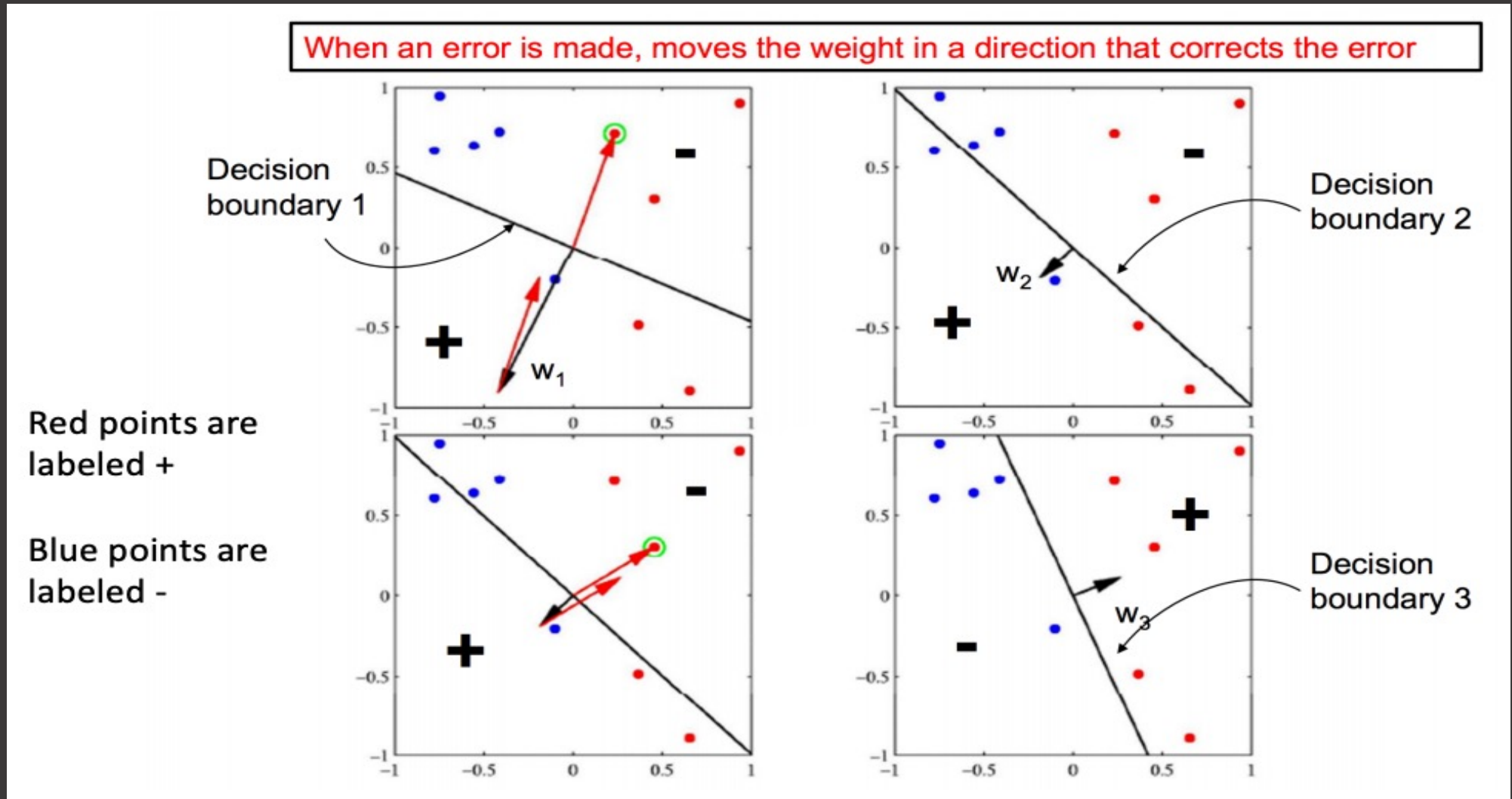
Perceptron Update Rule: If $\mathbf{x^{(i)}}$ is misclassified, do $w_j \leftarrow w_j + \alpha y^{(i)} x_j^{(i)}$

Perceptron Coverage Theorem

- Theorem: If a classification problem is linearly separable, a perceptron will reach a solution in a finite number of iterations
  - In other words, a perceptron can be trained to solve all linearly separable problems
  - A detailed proof is shown in any ML book, but we will not go over this in class

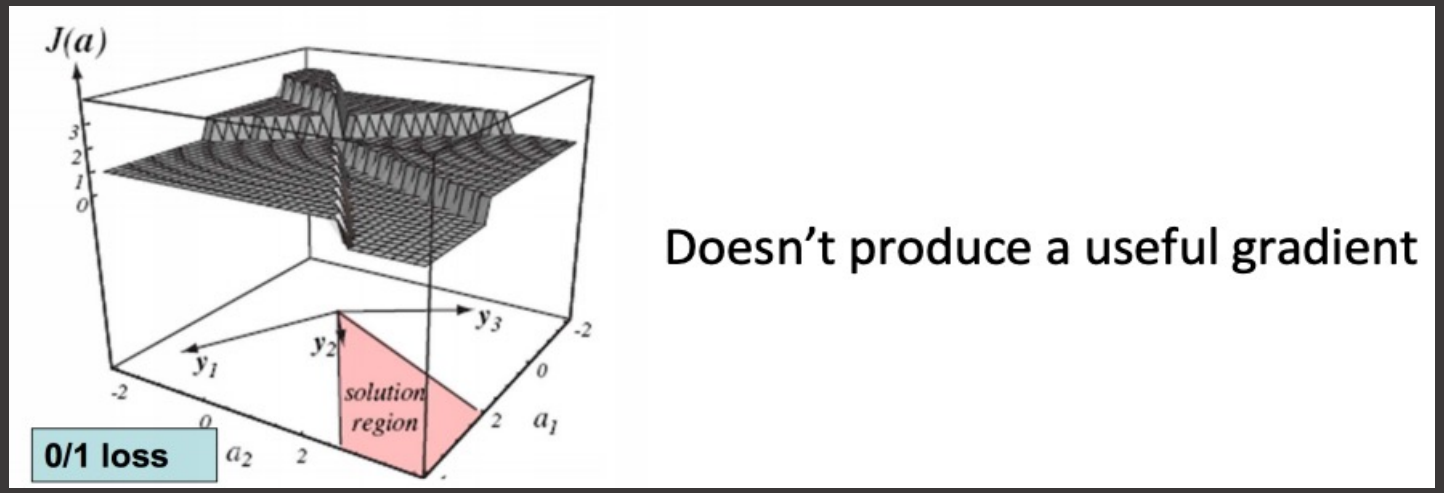
- The solution weight vector is not unique. There are infinite possible solutions and decision boundaries
- If the problem is not linearly separable then modifications need to be made

Perceptron


When an error is made, moves the weight in a direction that corrects the error
Decision boundary 1
Red points are labeled +
Blue points are labeled -
w1
w2
w3
Decision boundary 2
Decision boundary 3

Perceptron Cost Function

- Prediction is correct if $y^{(i)} w^T x^{(i)} > 0$
- Use 0/1 loss: $J_{0-1}(w) = \frac{1}{n} \sum_{i=1}^{n} l(sign(w^T x^{(i)}), y^{(i)})$



Doesn't produce a useful gradient

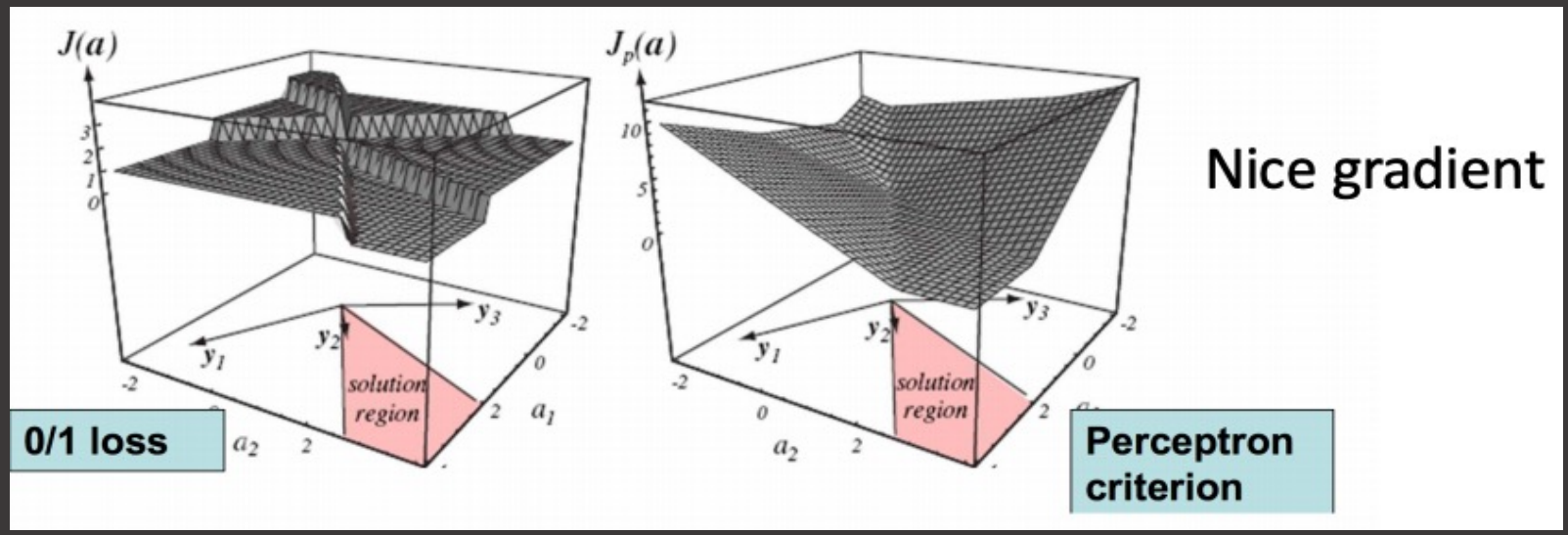
Slide credit: Alan Fern

Perceptron Cost Function

- The perceptron uses the following cost function:

$$J_p(w) = \frac{1}{n}\sum_{i=1}^{n} \max(0, -y^{(i)} w^T x^{(i)})$$

  - $\max(0, -y^{(i)} w^T x^{(i)})$ is 0 if the prediction is correct
  - Otherwise, it is the confidence in the misprediction



Slide credit: Alan Fern

Online Perceptron Algorithm

Let $\mathbf{w} \leftarrow [0,0,..,0]$
Repeat:
  Receive training example $(x^{(i)}, y^{(i)})$
  If $y^{(i)} x^{(i)} w \leq 0$
    $\mathbf{w} \leftarrow \mathbf{w} + y^{(i)} x^{(i)}$

Prediction is incorrect

Online learning – the learning mode where the model update is performed Each time a single observation is received

Batch learning – the learning mode where the model update is performed after observing the entire training set

Slide credit: Alan Fern

Batch Perceptron

Given training data $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$
Let w $\leftarrow [0,0,..,0]$
Repeat:
    Let $\Delta \leftarrow [0,0,..,0]$
    for $i = 1, \ldots, n$, do
        if $y^{(i)} x^{(i)} w \leq 0$
            $\Delta \leftarrow \Delta + y^{(i)} x^{(i)}$
    $\Delta \leftarrow \frac{\Delta}{n}$
    w $\leftarrow w + \alpha \Delta$
Until $||\Delta||_2 < \epsilon$

- Guaranteed to find a separating hyperplane if it exists.

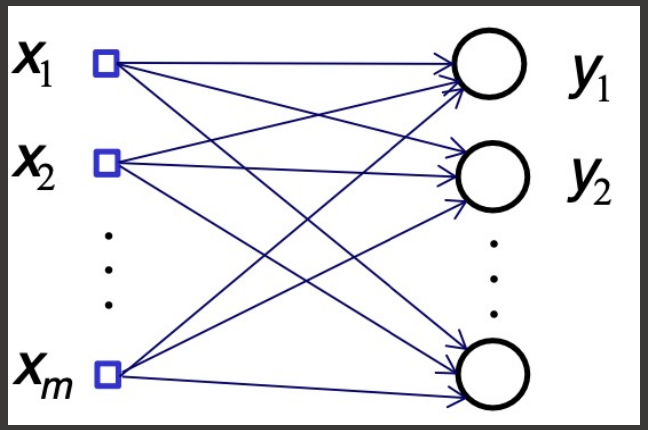
Slide credit: Alan Fern

Perceptron Improvement

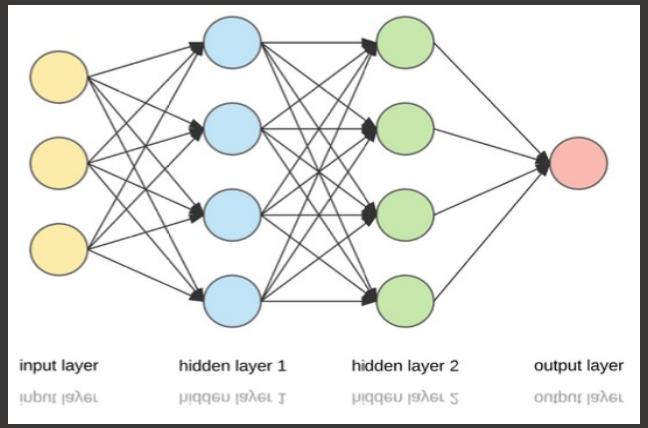
- **During training**
  - The perceptron produces many ws
- **During testing**
  - Only the final w is used
  - Some other w may be correct on $1{,}000$ consecutive examples, but one mistake ruins it
- ***Idea 1***: use a combination of multiple perceptrons
  - (e.g., neural networks)
- ***Idea 2***: Use the intermediate w's (e.g., voted / averaged perceptron)

Perceptrons for Multi-class classification

- Multiple perceptrons can be used when performing Multi-class classification (one for each class)
  - Perceptron output is 1 when input is from the corresponding class
  - Its output is 0 otherwise



- When these perceptons have the same inputs (but With different weights), this stacking of perceptrons Is called a ***layer***



- When the outputs of one layer become the input to another, we call this a ***multi-layer perceptron (MLP)*** or ***neural network***. ***Deep neural networks*** have three or more layers.