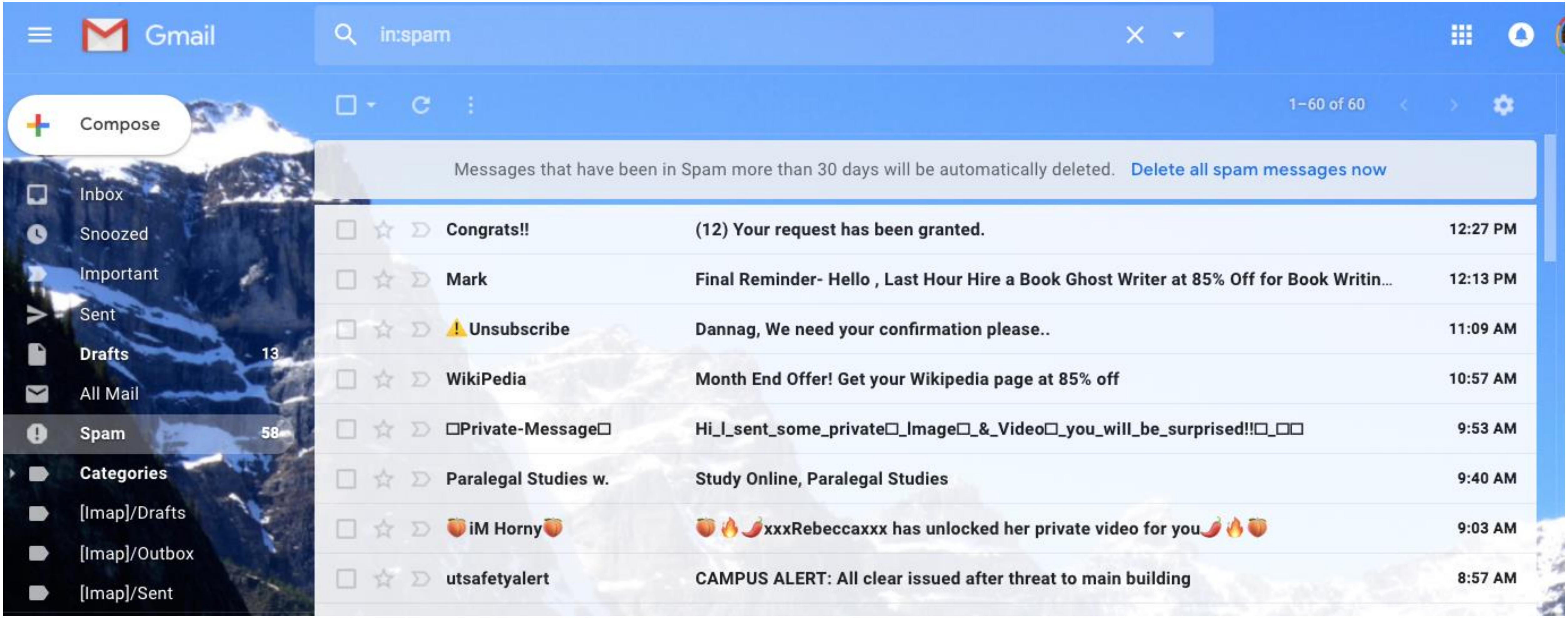
Artificial Neural Networks

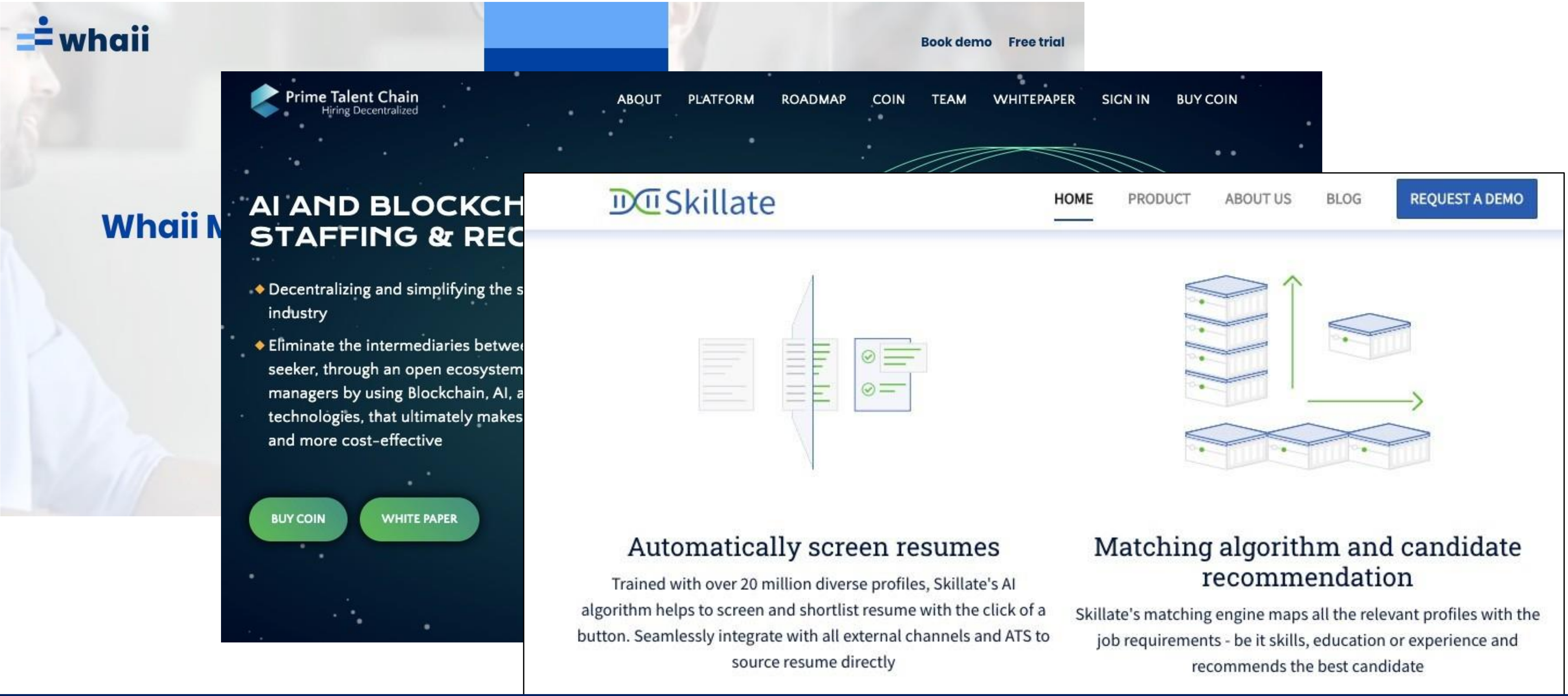
Binary Classification

Distinguish 2 classes

Binary Classification: Spam Detection



Binary Classification: Resume Pre-Screening



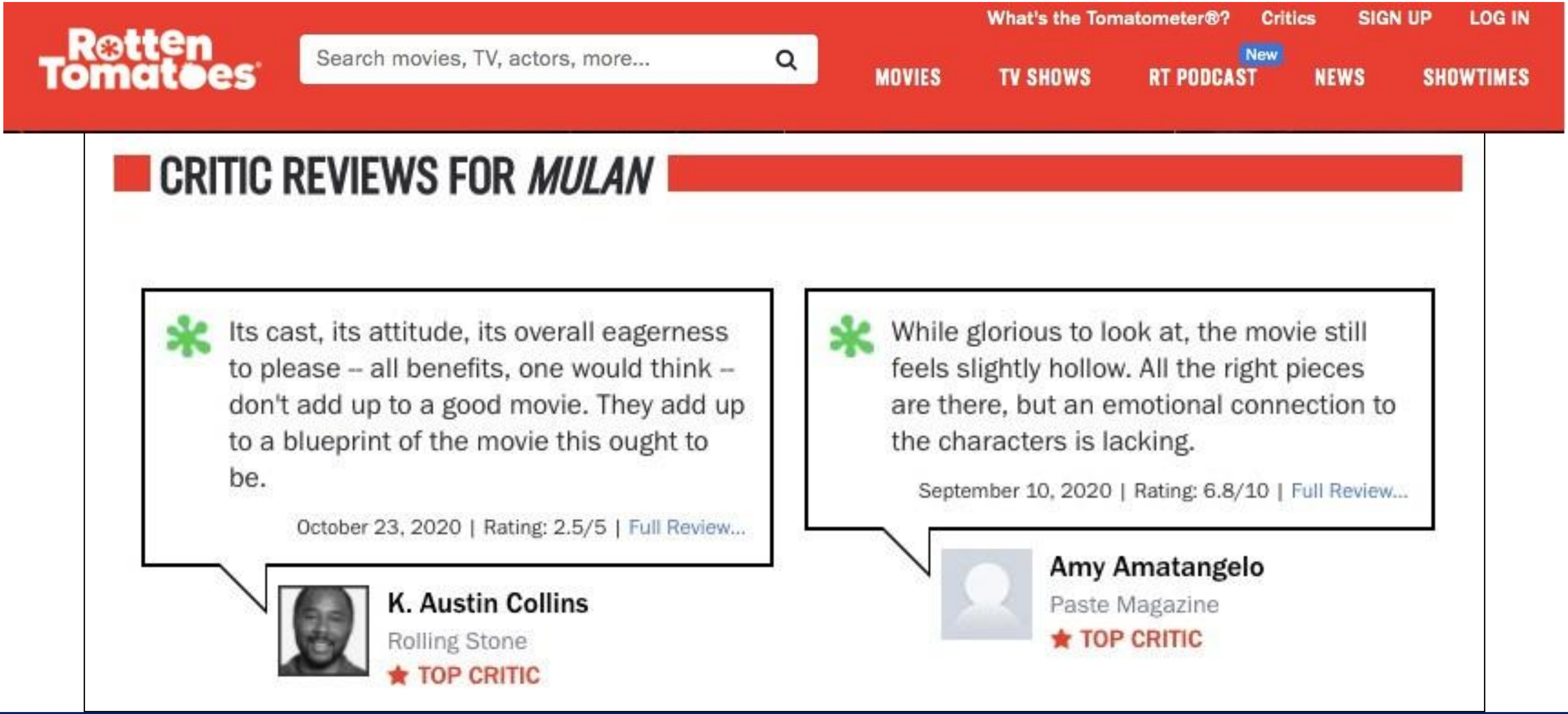
Binary Classification: Cancer Diagnosis

Binary Classification: Cognitive Impairment Recognition by Apple App Usage

Image Credit: https://www.techradar.com/news/the-10-best-phones-for-seniors

https://www.technologyreview.com/f/615032/the-apps-you-use-on-your-phone-could-help-diagnose-your-cognitive-

Binary Classification: Sentiment Analysis



Binary Classification: Food Quality Control

Machine Learning: Using Algorithms to Sort Fruit

Demo: https://www.youtube.com/watch?v=Bl3XzBWpZbY

Goal: Design Models that Generalize Well to New, Previously Unseen Examples

Example:

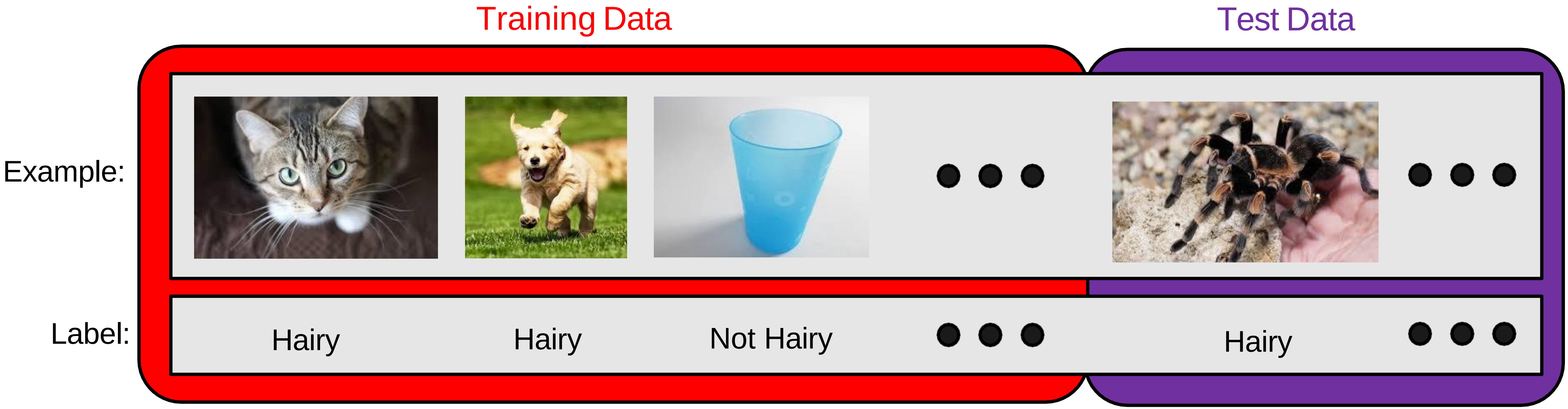
● ● ●

● ● ●

Label: Hairy Hairy Not Hairy ● ● ● Hairy ● ● ●

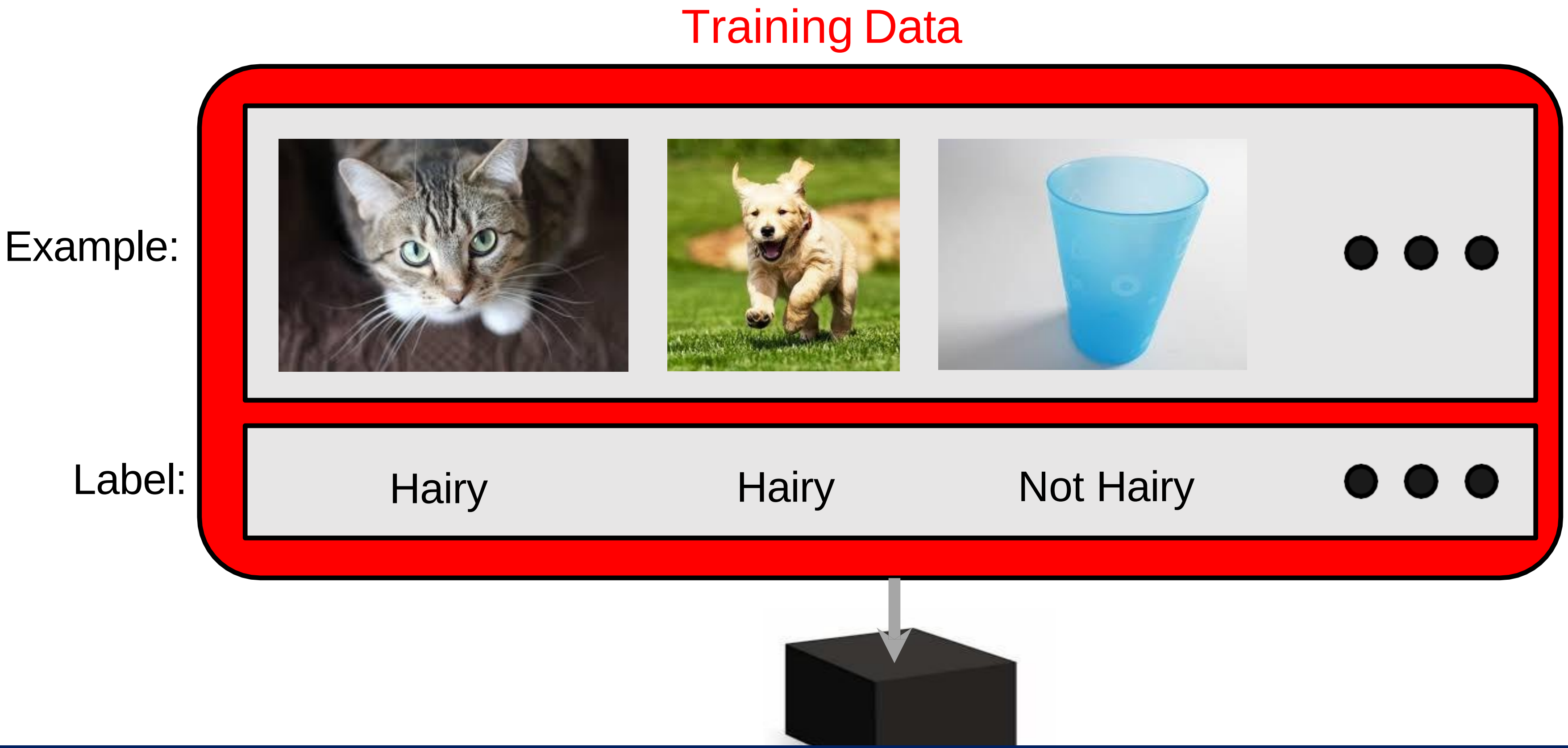
Goal: Design Models that Generalize Well to New, Previously Unseen Examples

1. Split data into a "training set" and "test set"



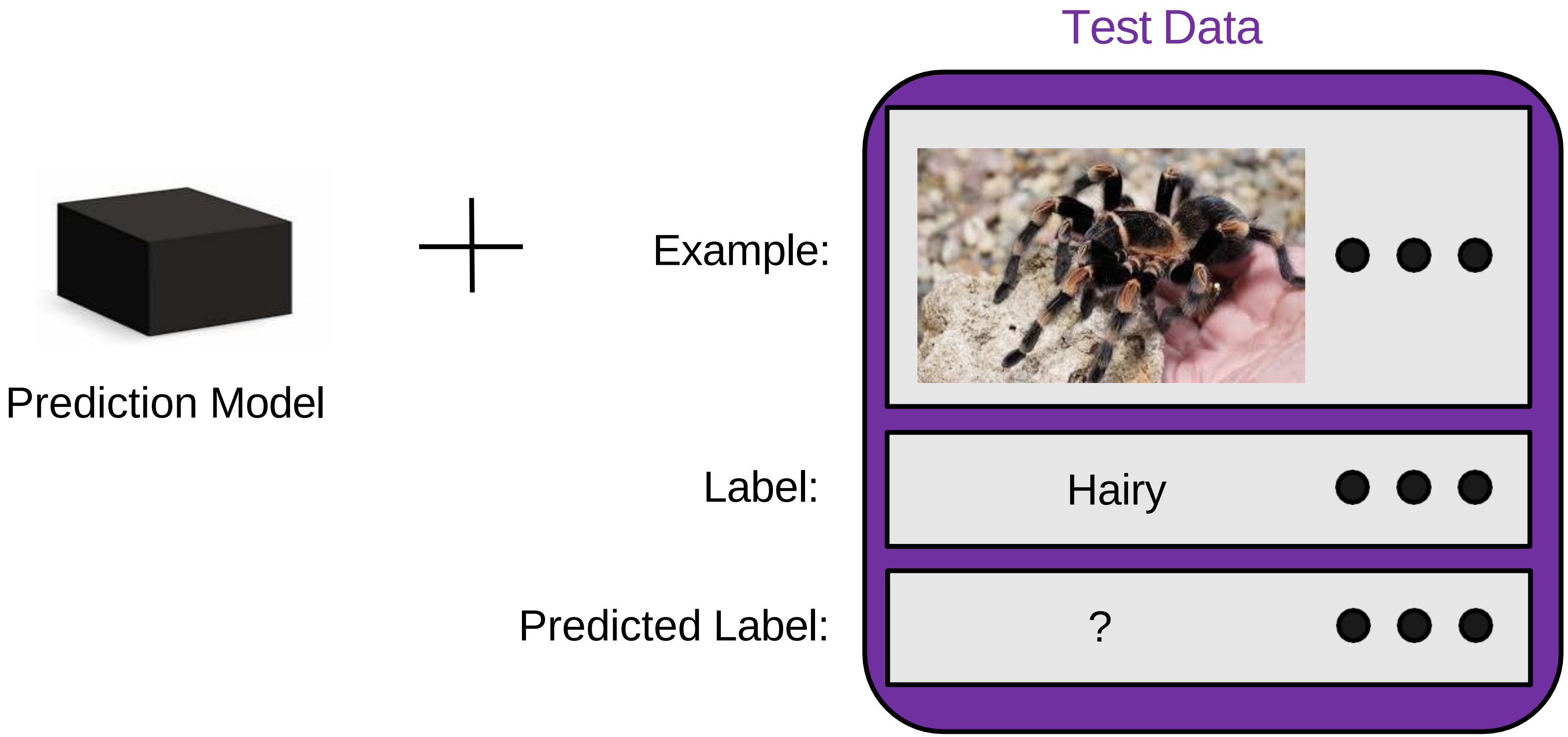
Goal: Design Models that Generalize Well to New, Previously Unseen Examples

2. Train model on “training set” to try to minimize prediction error on it



Goal: Design Models that Generalize Well to New, Previously Unseen Examples

3. Apply trained model on "test set" to measure generalization error



Goal: Design Models that Generalize Well to New, Previously Unseen Examples

3. Apply trained model on "test set" to measure generalization error

Prediction Model

+

Test Data

Example:  ● ● ●

Label: Hairy ● ● ●

Predicted Label: Not Hairy ● ● ●

Goal: Design Models that Generalize Well to New, Previously Unseen Examples

3. Apply trained model on “test set” to measure generalization error

Prediction Model

+

Test Data

Example:

● ● ●

Label: Hairy ● ● ●

Predicted Label: Not Hairy ● ● ●

Inspiration: Animal's Computing Machinery

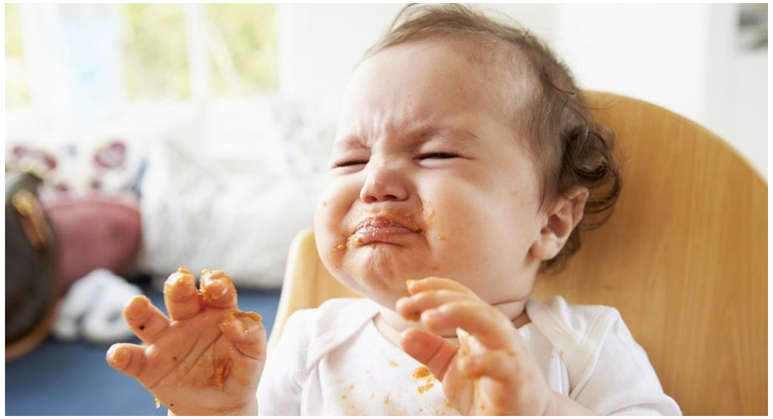
Neuron

- basic unit in the nervous system for receiving, processing, and transmitting information; e.g., messages such as...

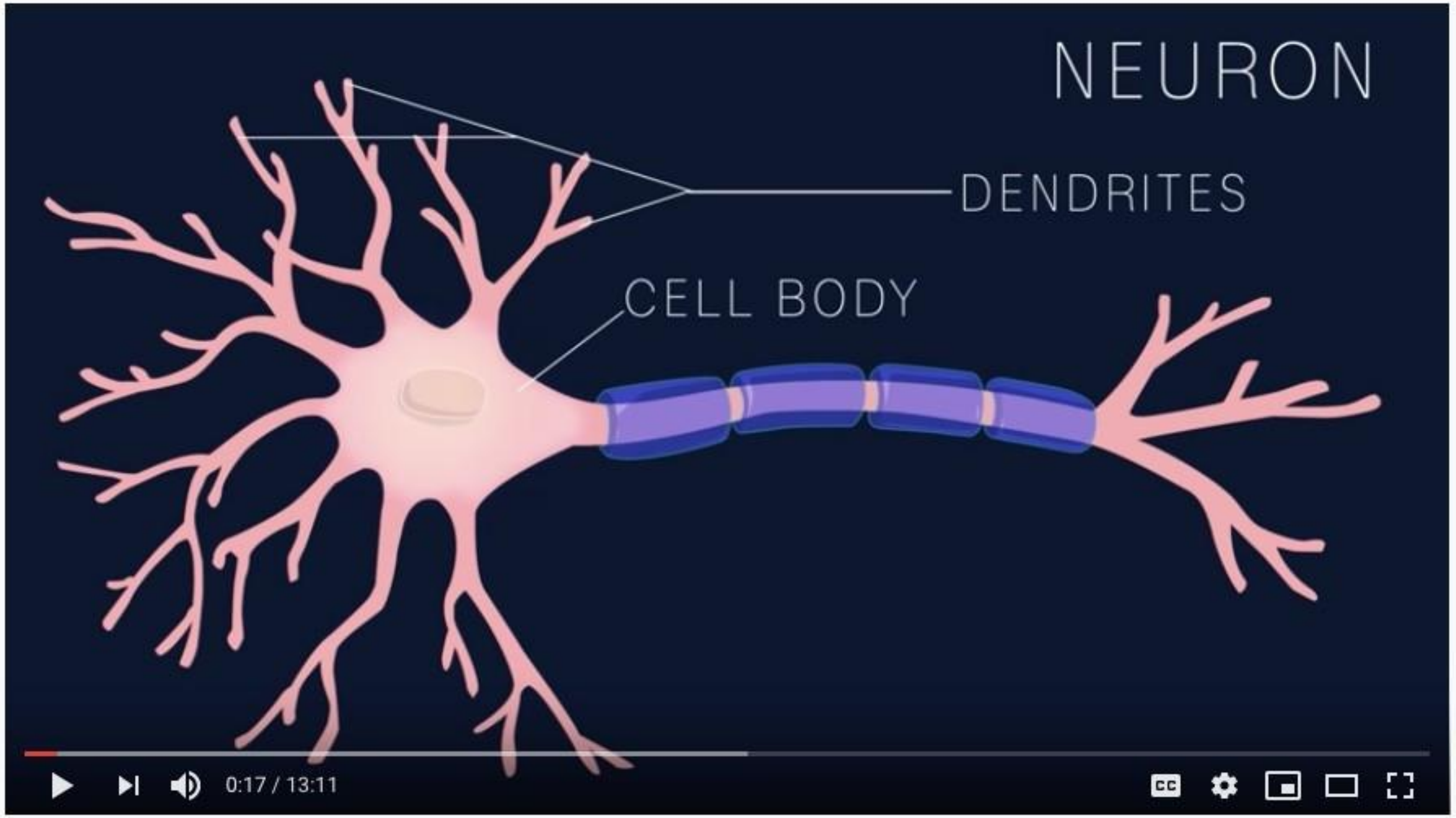
"hot"

"loud"

"spicy"

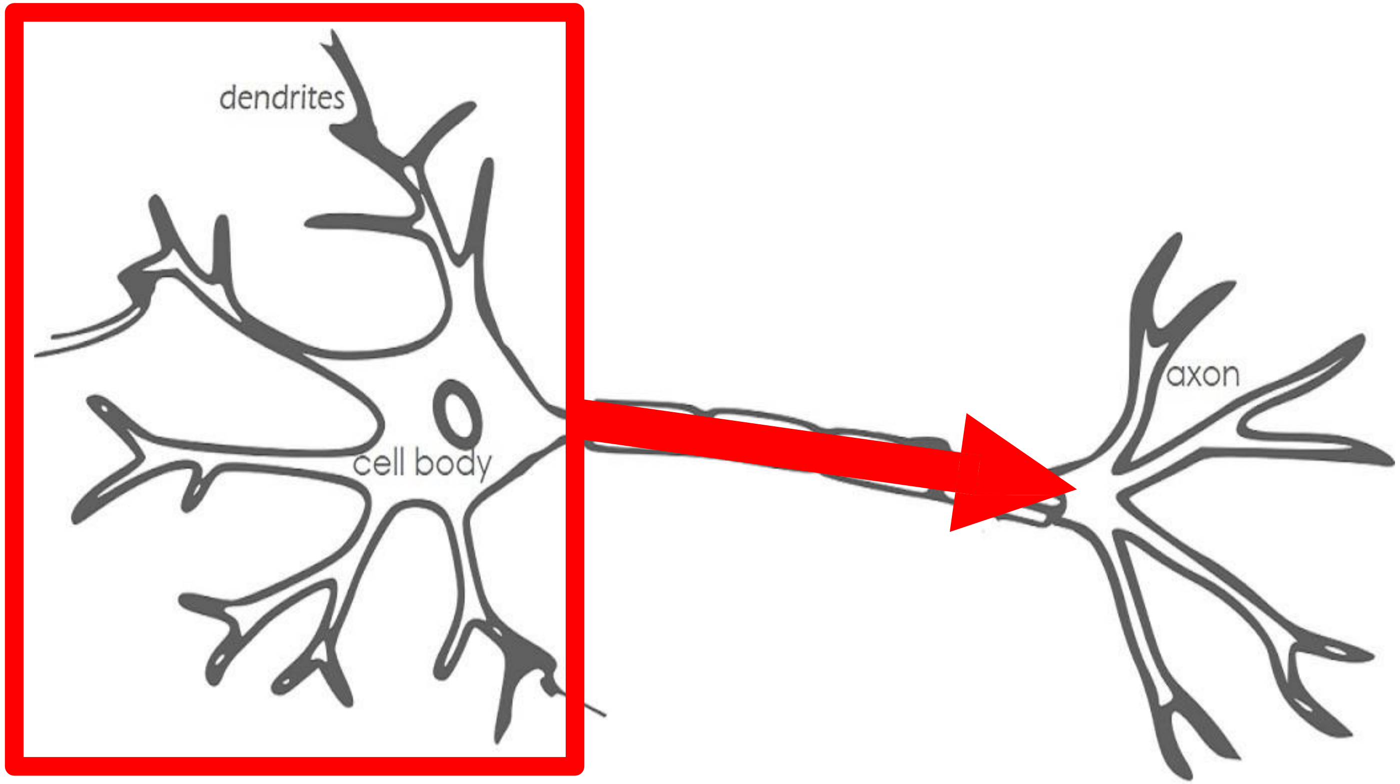


Inspiration: Animal's Computing Machinery



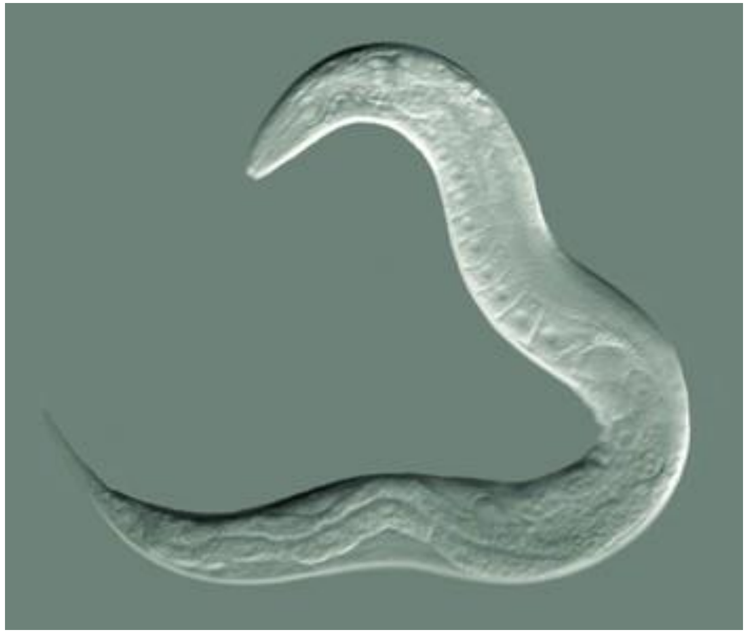
Action Potential in the Neuron

Inspiration: Neuron “Firing”

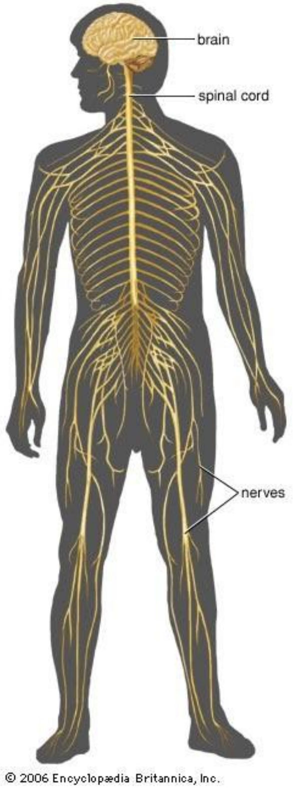


- When the input signals exceed a certain threshold within a short period of time, a neuron “fires”
- Neuron “firing” (outputs signal) is an “all-or-none” process

Inspiration: Animal's Computing Machinery



https://en.wikipedia.org/wiki/Nematode#/media/File:CelegansGoldsteinLabUNC.jpg

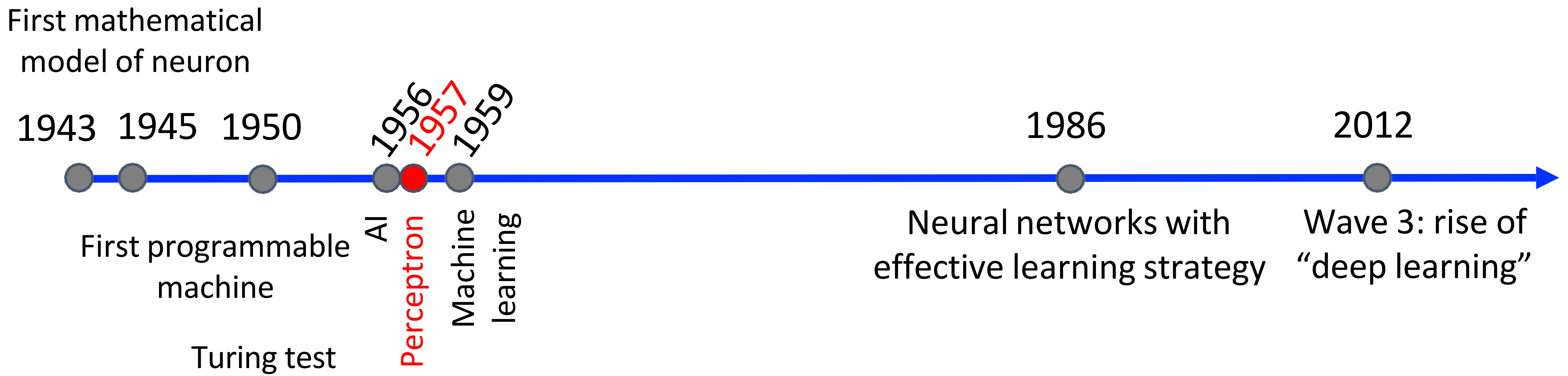


https://www.britannica.com/science/human-nervous-system

Nematode worm: 302 neurons

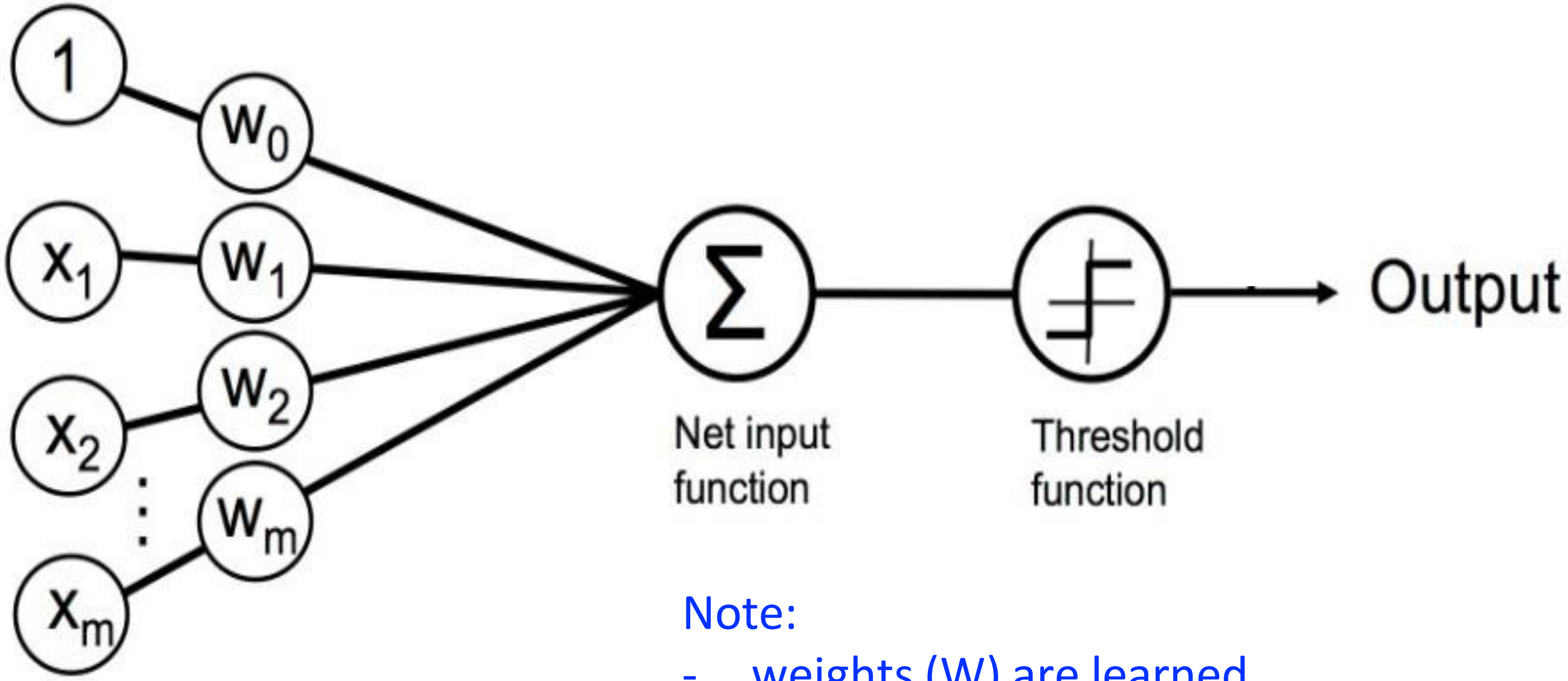
Human: ~Billions neurons

Historical Context: Artificial Neurons



Modern deep learning algorithms rely on techniques developed over the past 65 years.

Perceptron: Model (Linear Threshold Unit)

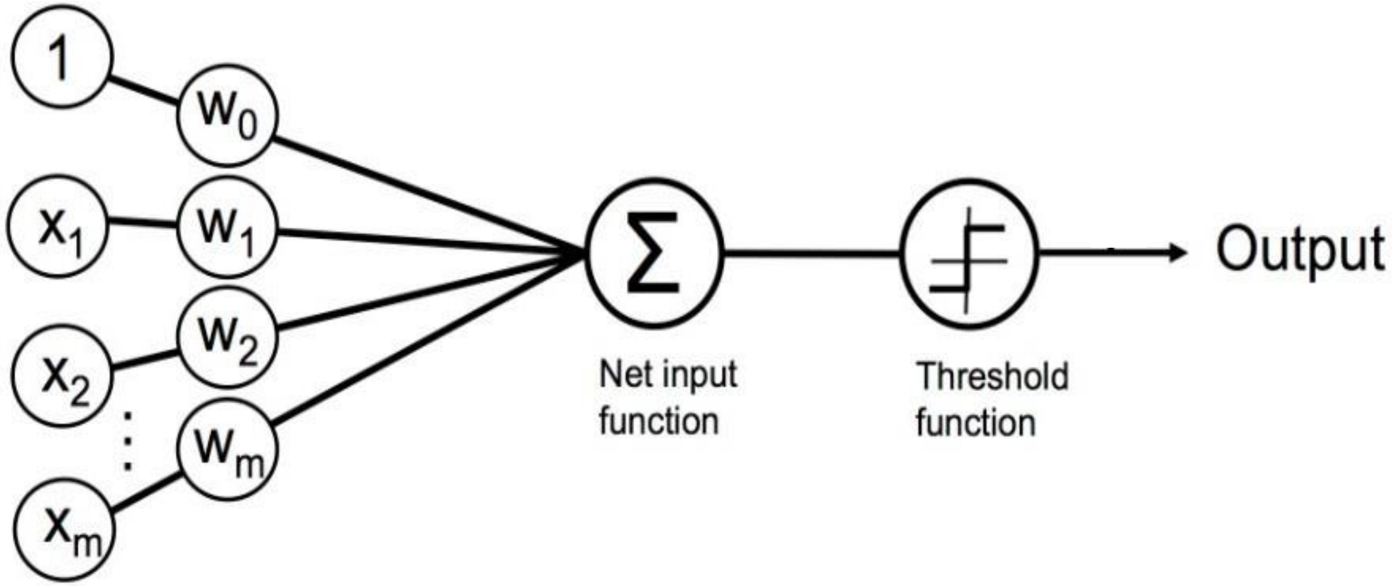


Note:
- weights (W) are learned
- inputs and weights can be any value
- fires when combined input exceeds threshold

Python Machine Learning; Raschka & Mirjalili

Frank Rosenblatt, The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory, 1957

Perceptron: Model (Linear Threshold Unit)



$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

Python Machine Learning; Raschka & Mirjalili

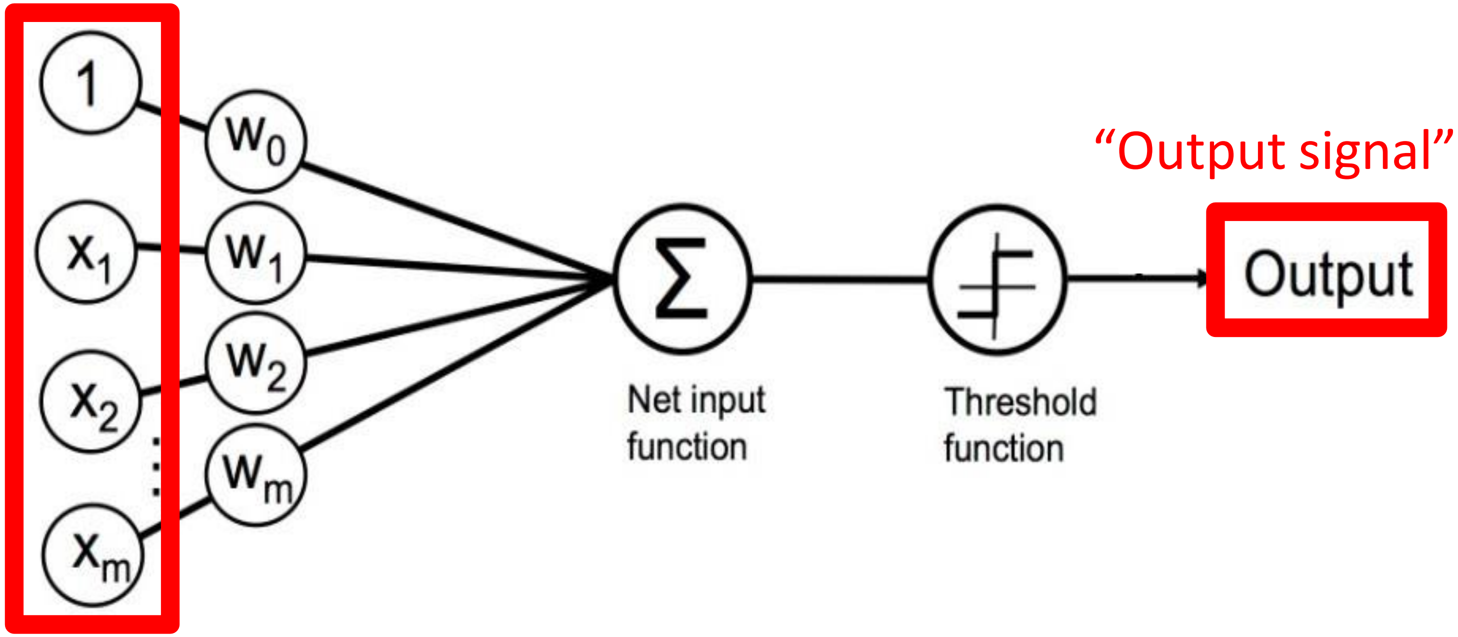
Frank Rosenblatt, The perceptron, a perceiving and recognizing automaton Project Para, Cornell Aeronautical Laboratory, 1957

Perceptron: Model (Linear Threshold Unit)

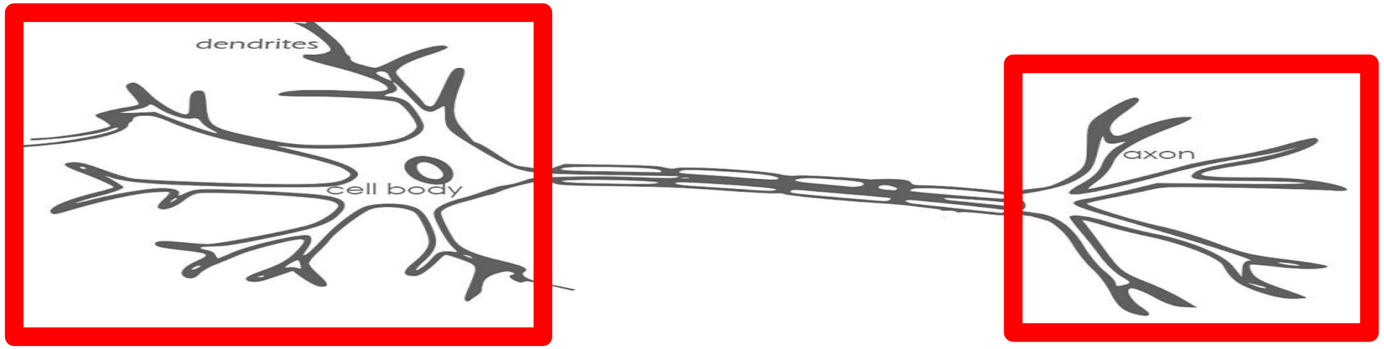
Artificial Neuron:

"Input signals"

"Output signal"

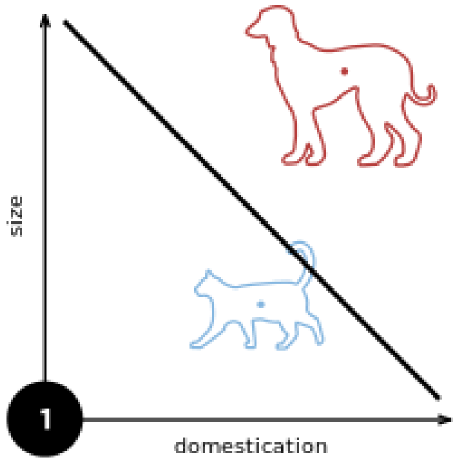


Biological Neuron:



Python Machine Learning; Raschka & Mirjalili

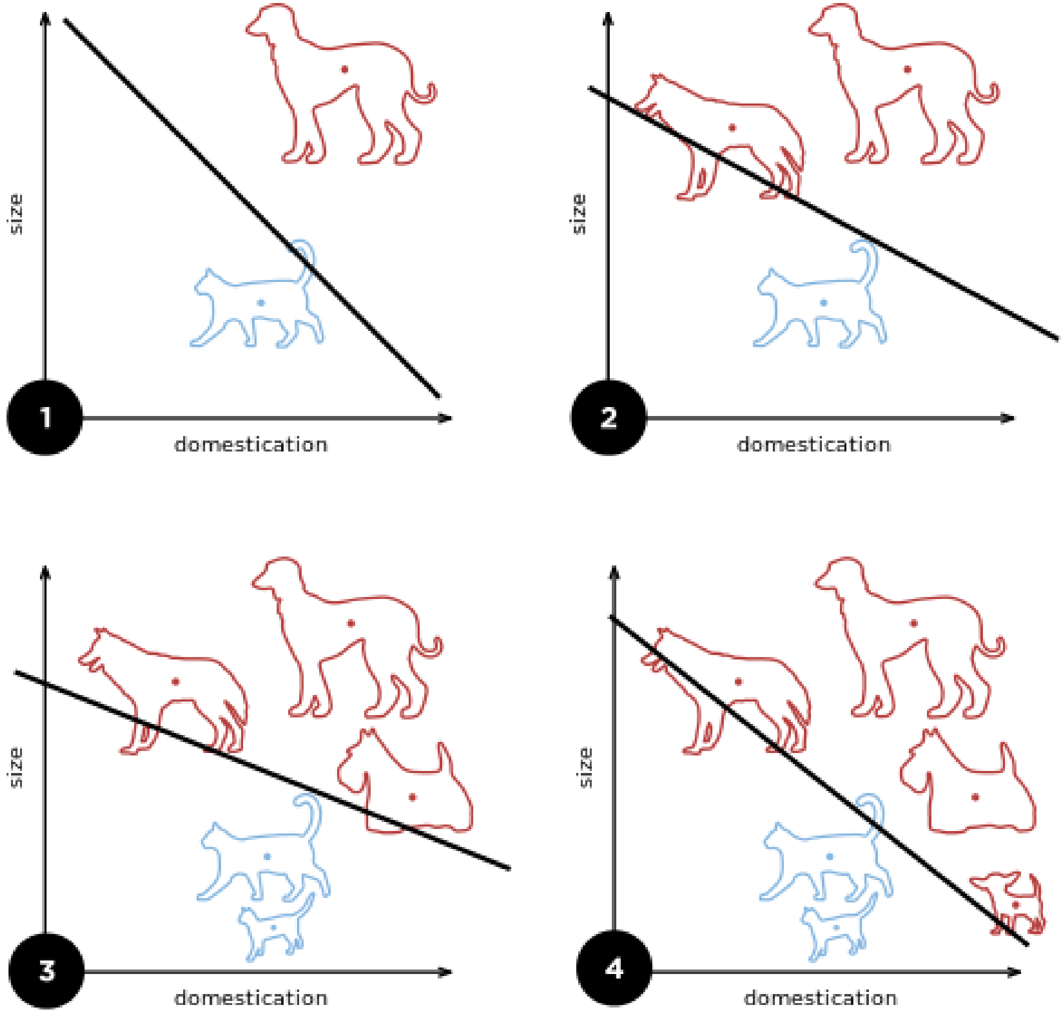
Perceptron: Learning Algorithm



Iteratively update linear boundary with observation of each additional example:

Perceptron: Learning Algorithm

Iteratively update linear boundary with observation of each additional example:



Perceptron: Learning Algorithm

1. Initialize weights to 0 or small random numbers
2. For each training sample:
    1. Compute output value: $\sum_{j=0}^{m} x_j w_j = w^T x$
    2. Update weights with the following definition: $w_j := w_j + \boxed{\Delta w_j}$

$$\boxed{\Delta w_j = \underbrace{\eta}_{\text{Learning Rate}} \left(\underbrace{\text{target}^{(i)}}_{\text{True Class Label}} - \underbrace{\text{output}^{(i)}}_{\text{Predicted Class Label}}\right) x_j^{(i)}}$$

Perceptron: Learning Algorithm - What Happens to Weights When It Predicts Correct Class Label?

1. Initialize weights to 0 or small random numbers
2. For each training sample:
   1. Compute output value: $\sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}$
   2. Update weights with the following definition: $w_j := w_j + \boxed{\Delta w_j}$

equals 0, so no weight update

$$\Delta w_j = \eta \boxed{\left(\text{target}^{(i)} - \text{output}^{(i)}\right)} x_j^{(i)}$$

Learning Rate

True Class Label

Predicted Class Label

Perceptron: Learning Algorithm - What Happens to Weights When It Predicts Wrong Class Label?

1. Initialize weights to 0 or small random numbers
2. For each training sample:
   1. Compute output value: $\sum_{j=0}^{m} x_j w_j = w^T x$
   2. Update weights with the following definition: $w_j := w_j + \boxed{\Delta w_j}$

Equals positive or negative value, so weights change

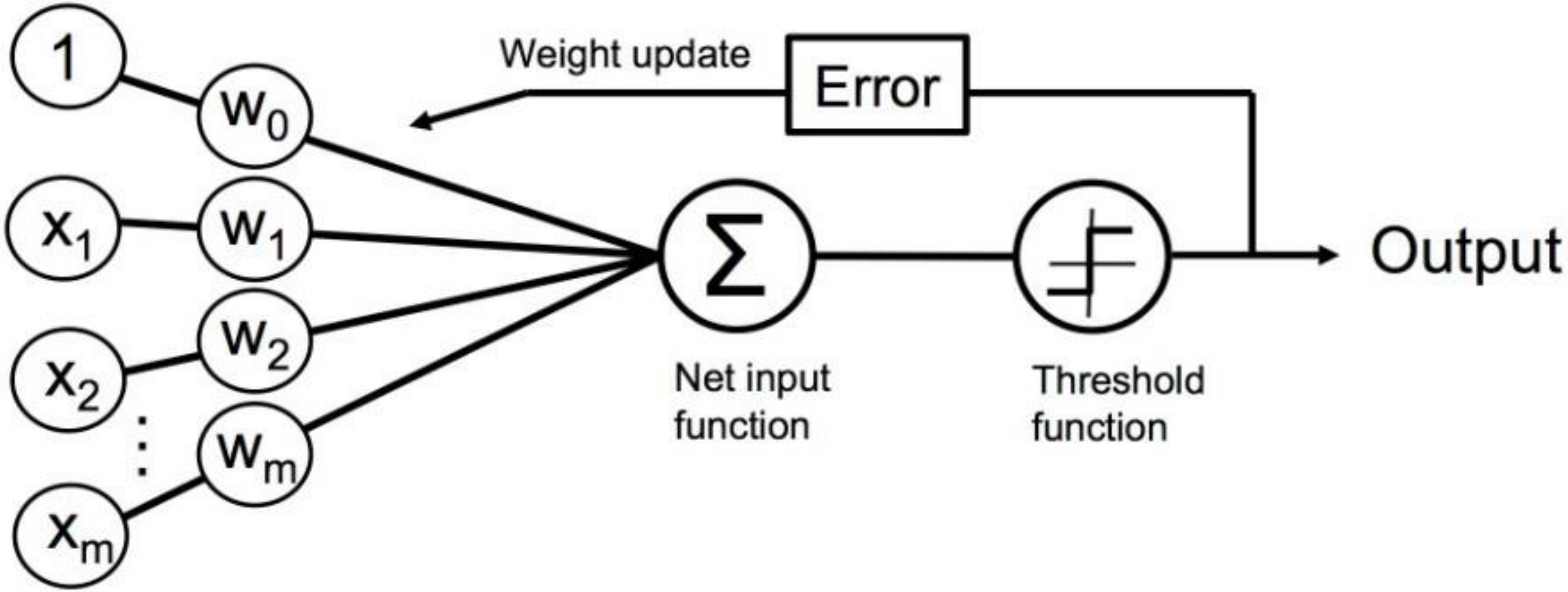
$$\Delta w_j = \eta \boxed{\left(\text{target}^{(i)} - \text{output}^{(i)}\right)} x_j^{(i)}$$

Learning Rate

True Class Label

Predicted Class Label

Perceptron: Learning Algorithm



Python Machine Learning: Raschka & Mirjalili

Perceptron: Learning Algorithm Choices

- Learning rate
- Number of epochs (passes over the dataset)

Perceptron Limitation: XOR Problem

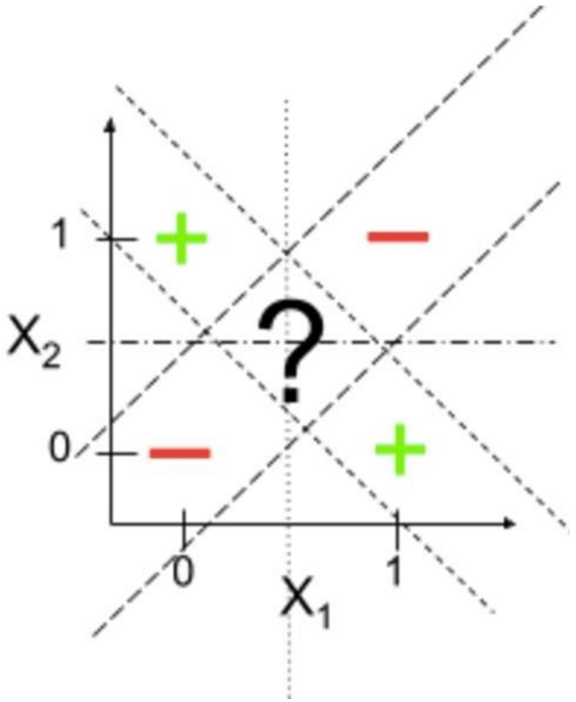
XOR = “Exclusive Or”

- Input: two binary values x_1 and x_2
- Output:
  - 1, when exactly one input equals 1
  - 0, otherwise

x_1	x_2	x_1 XOR x_2
0	0	?
0	1	?
1	0	?
1	1	?

Perceptron Limitation: XOR Problem

Cannot solve XOR problem and so separate 1s from 0s with a perceptron (linear function):



x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Perceptron Limitation: XOR Problem

Frank Rosenblatt

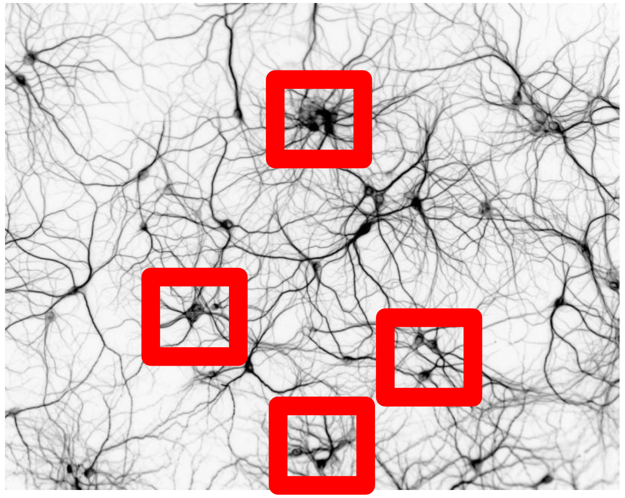
ychol

$100

How can a machine be "conscious" when it can't solve the XOR problem?

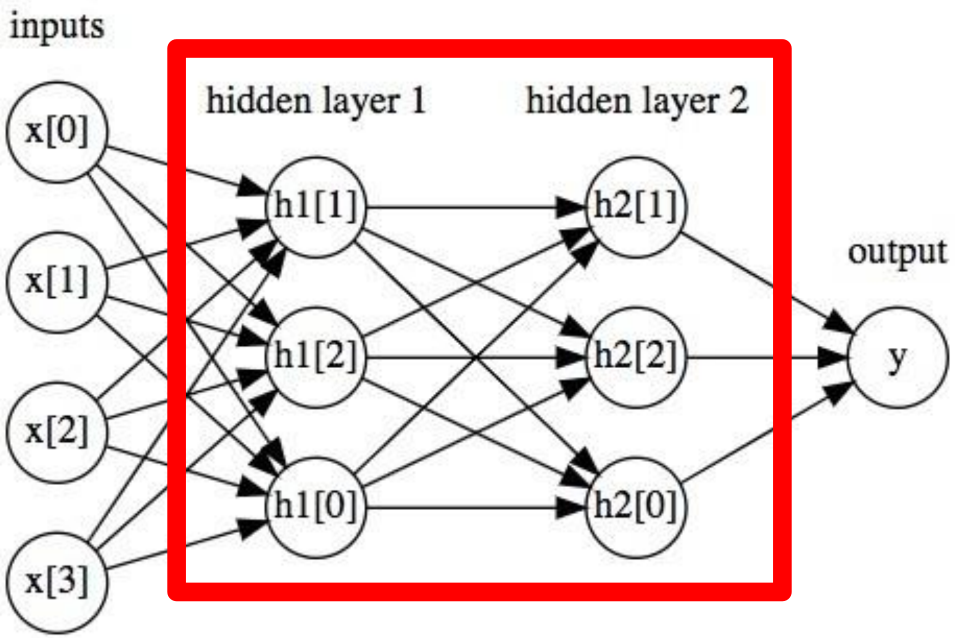
Neural Networks: Connected Neurons

Biological Neural Network:

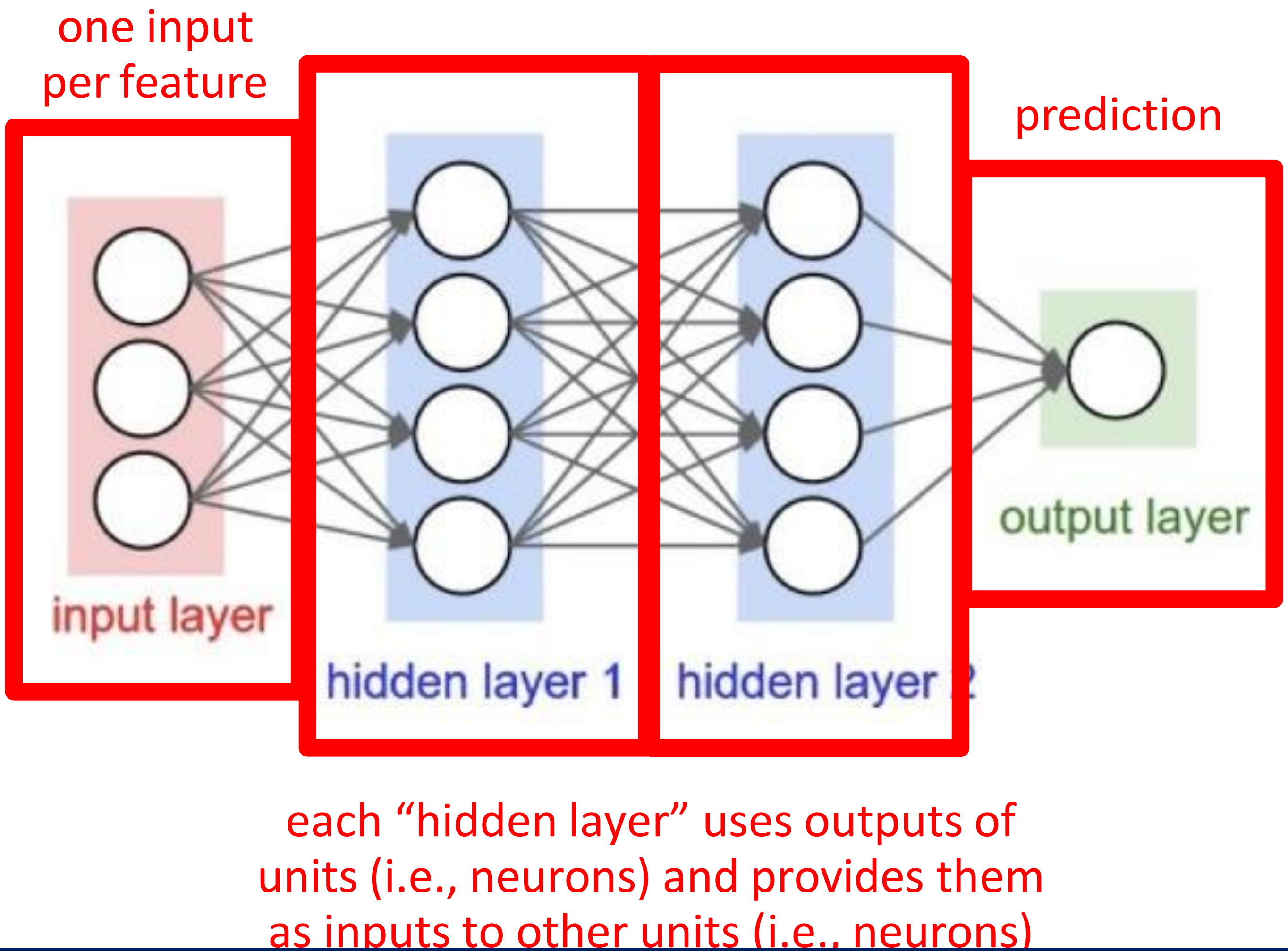


http://www.rzagabe.com/2014/11/03/an-introduction-to-artificial-neural-networks.html

Artificial Neural Network:



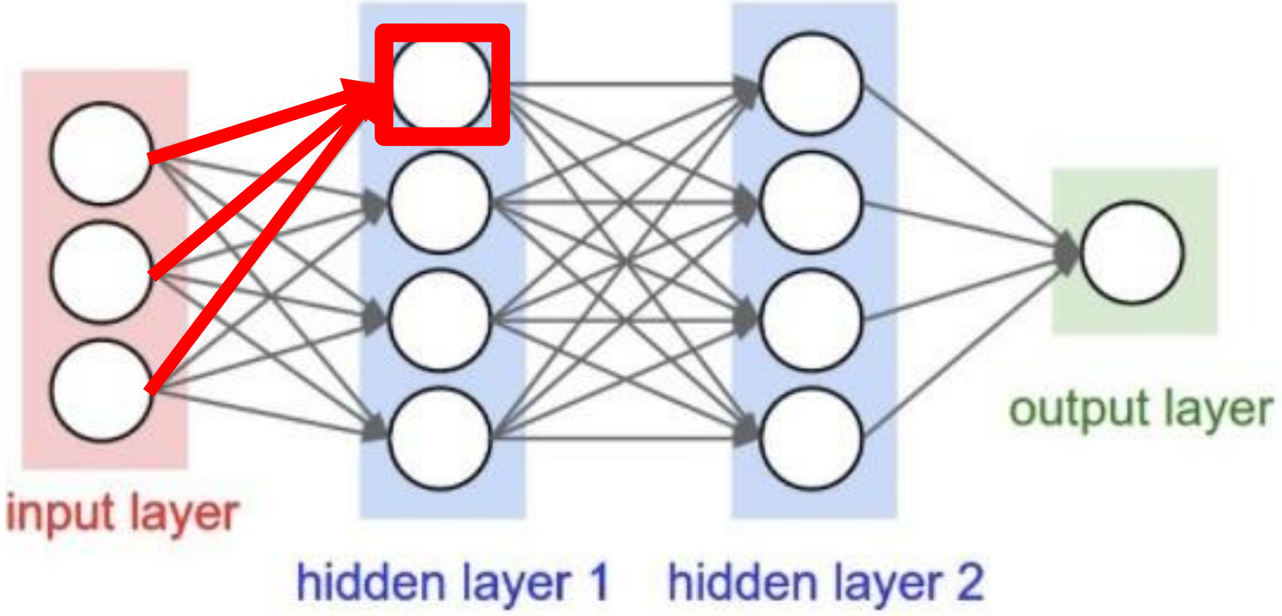
Neural Network



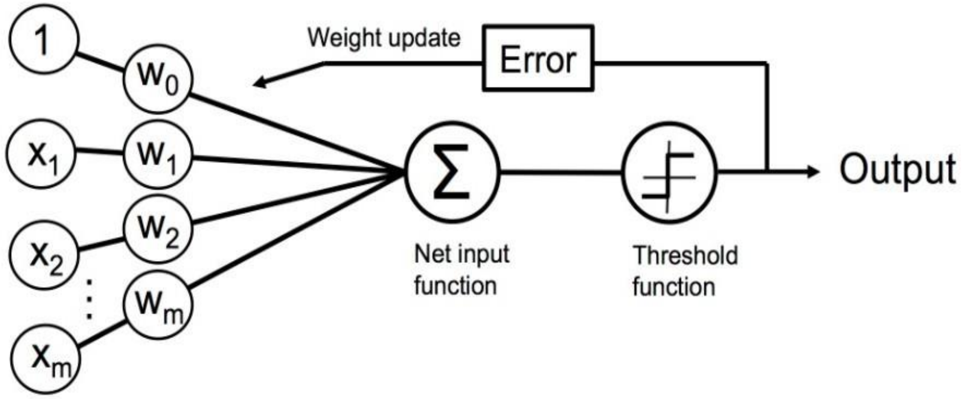
This is a 3-layer neural network (i.e., count number of hidden layers plus output layer)

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

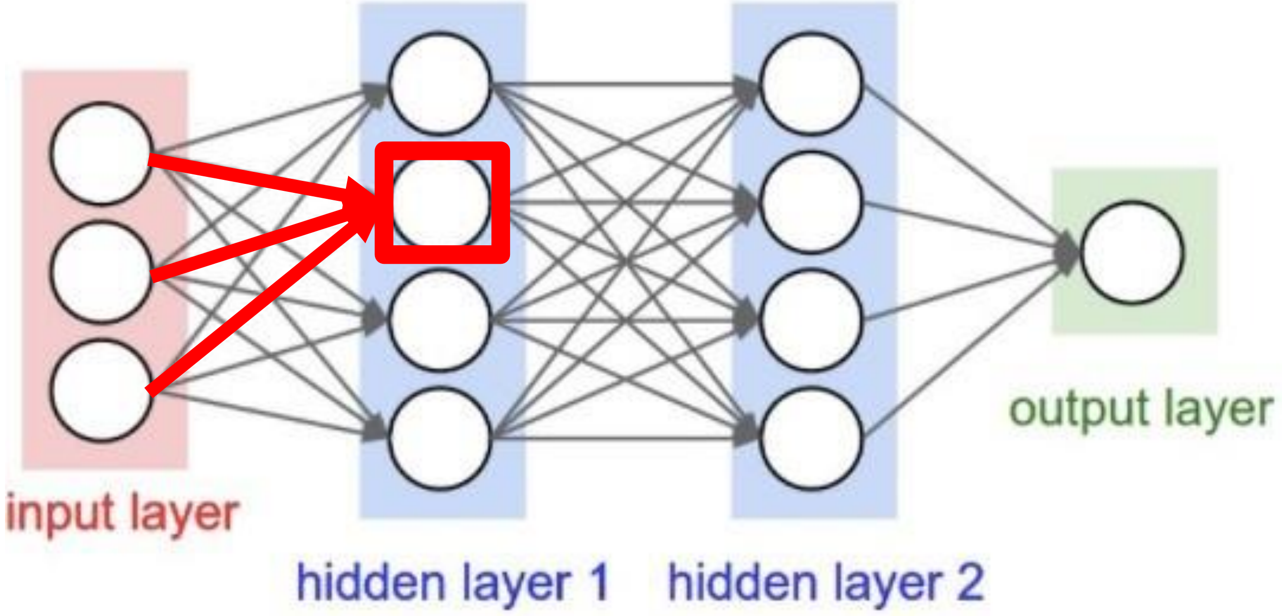


- Unit: takes as input a weighted sum and applies an activation function

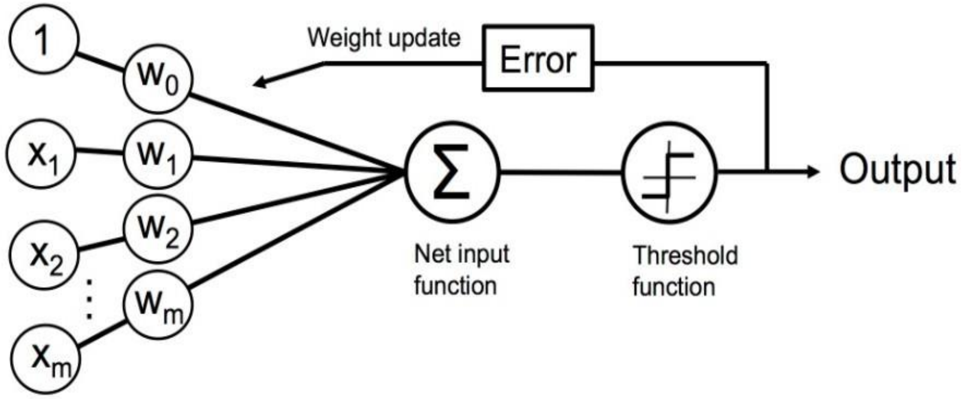
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

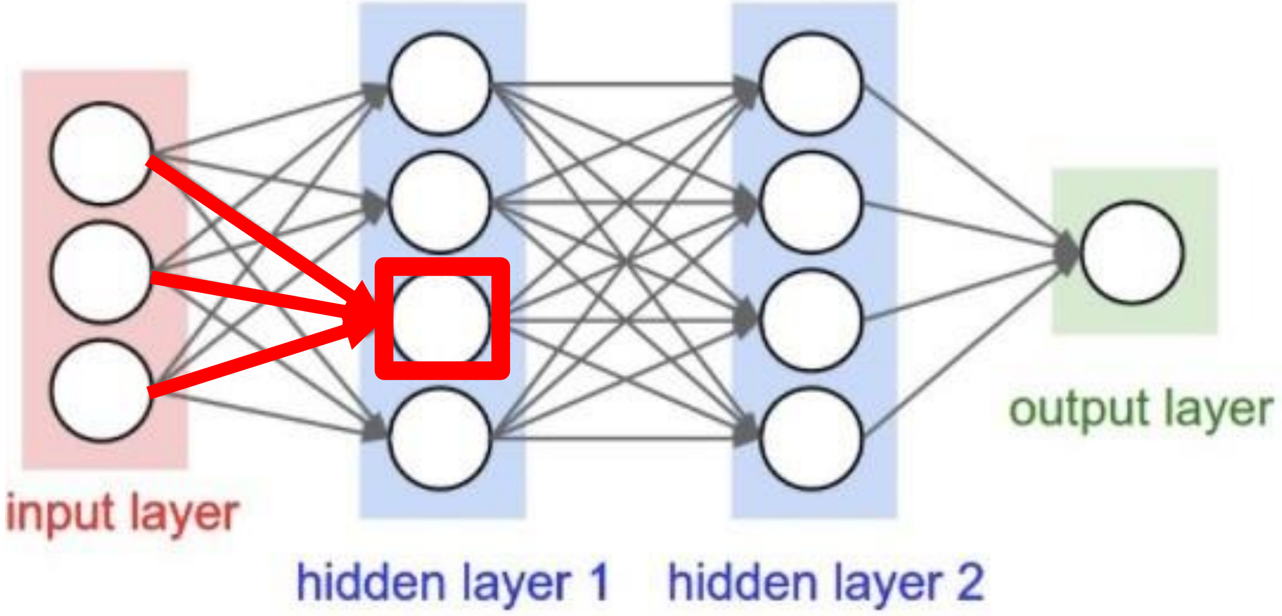


- Unit: takes as input a weighted sum and applies an activation function

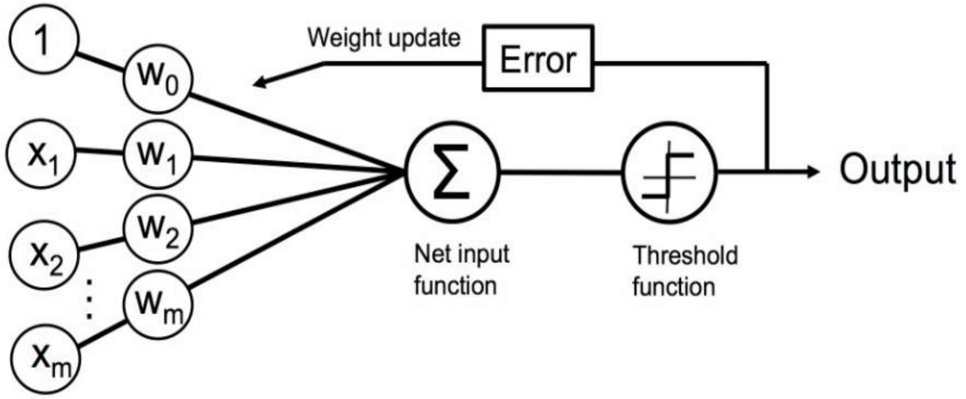
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

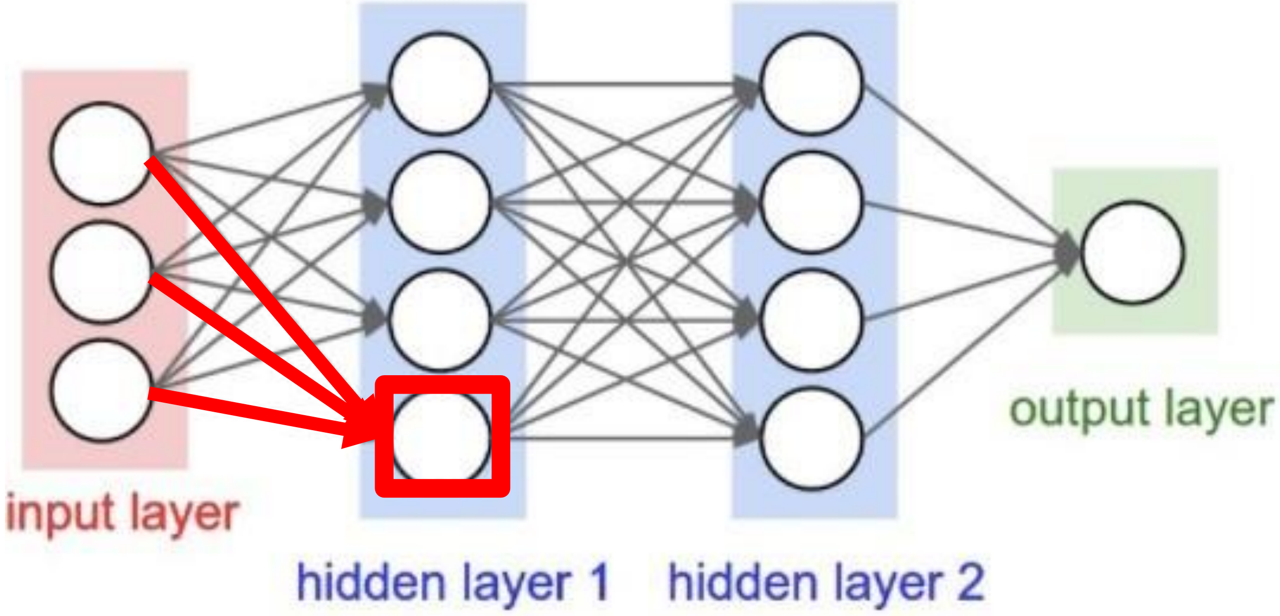


- Unit: takes as input a weighted sum and applies an activation function

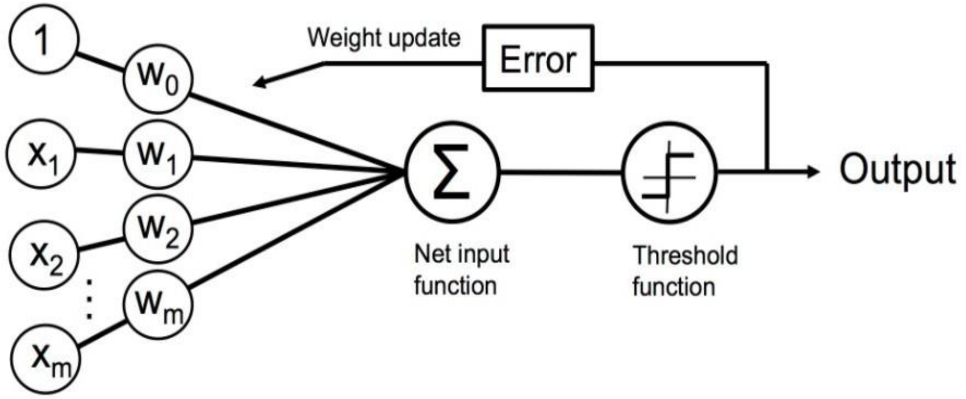
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

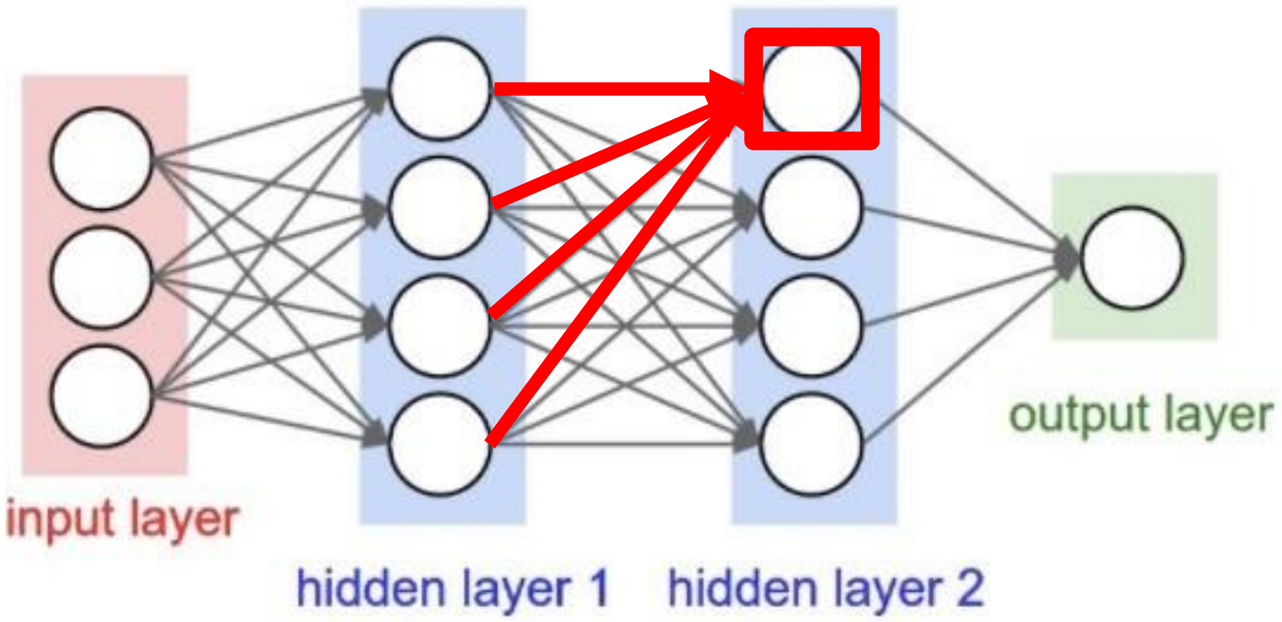


- Unit: takes as input a weighted sum and applies an activation function

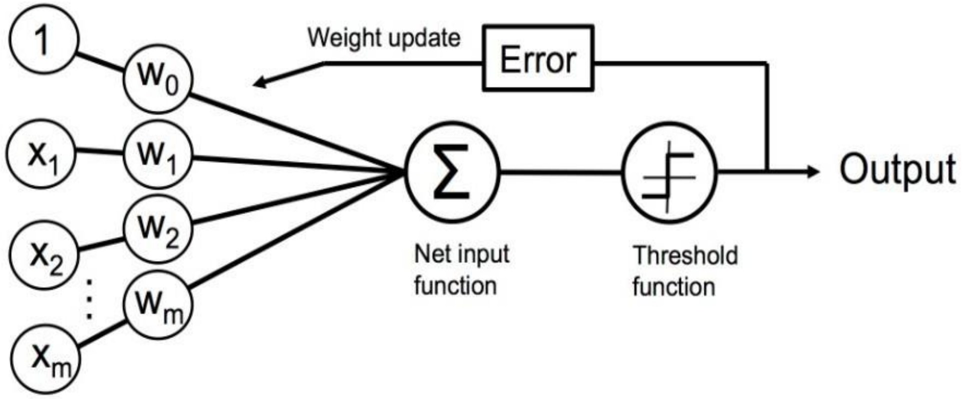
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

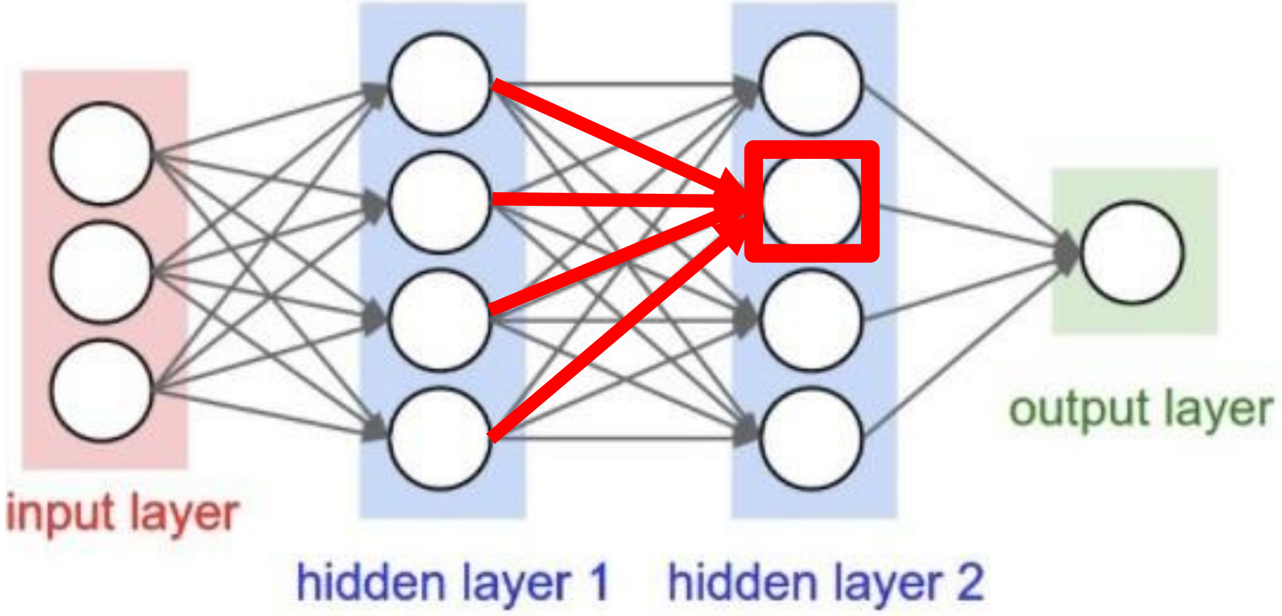


- Unit: takes as input a weighted sum and applies an activation function

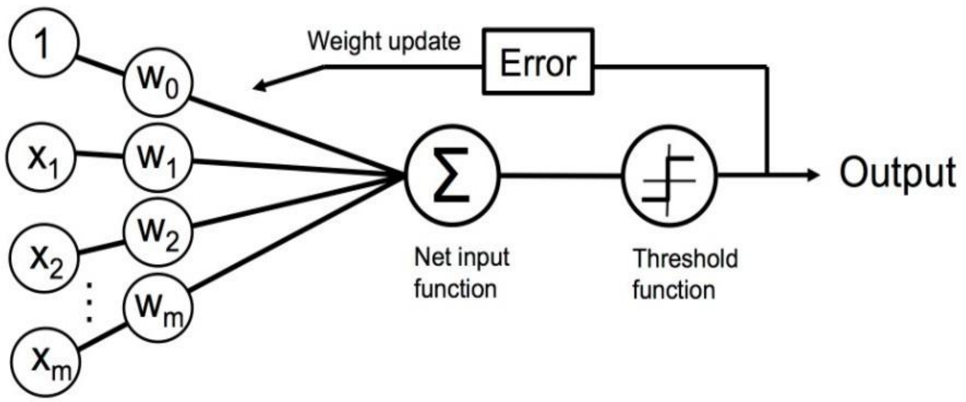
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

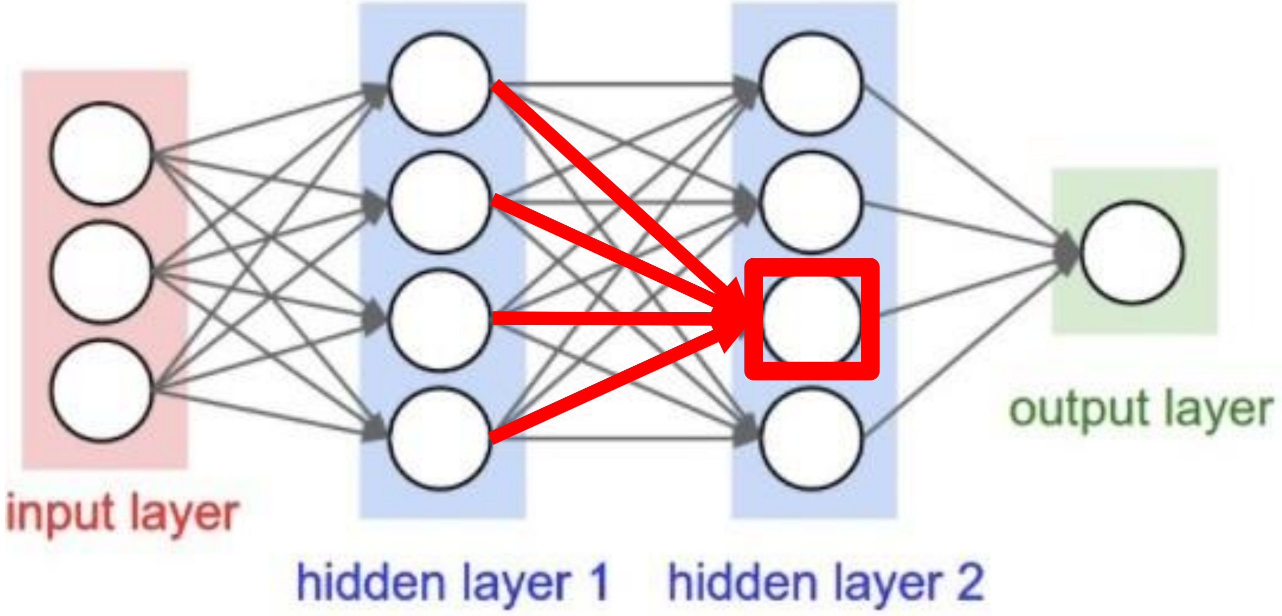


- Unit: takes as input a weighted sum and applies an activation function

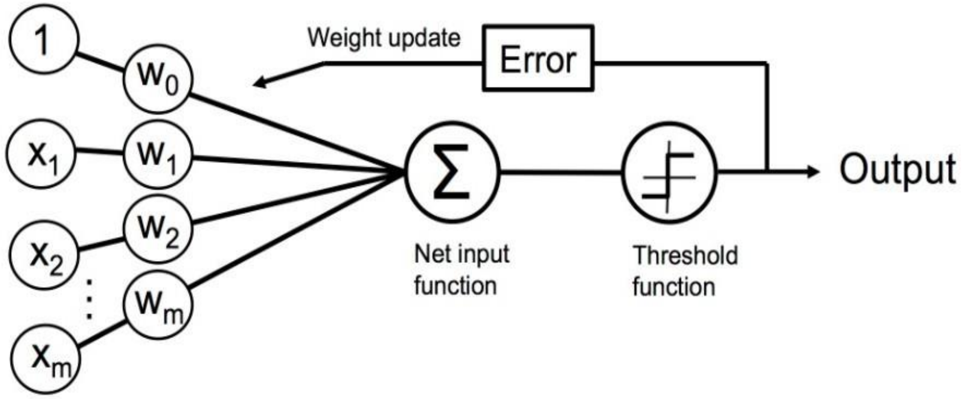
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

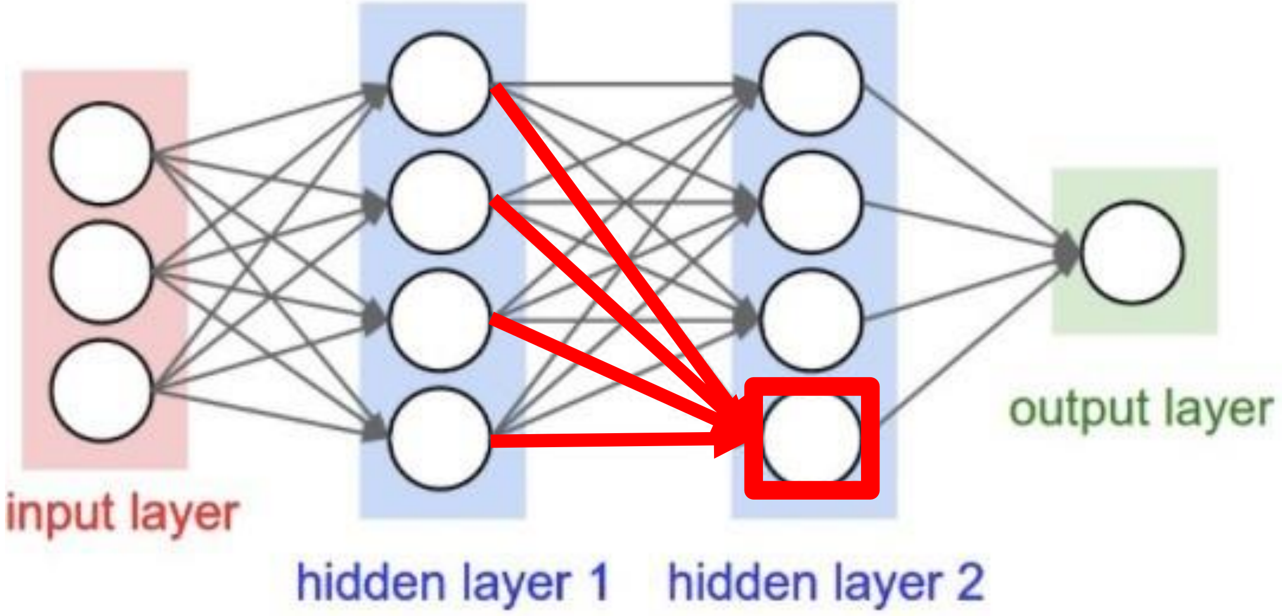


- Unit: takes as input a weighted sum and applies an activation function

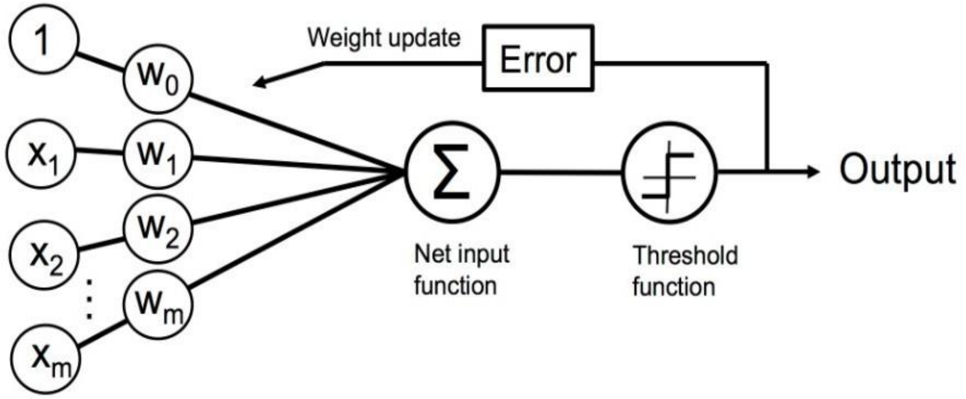
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

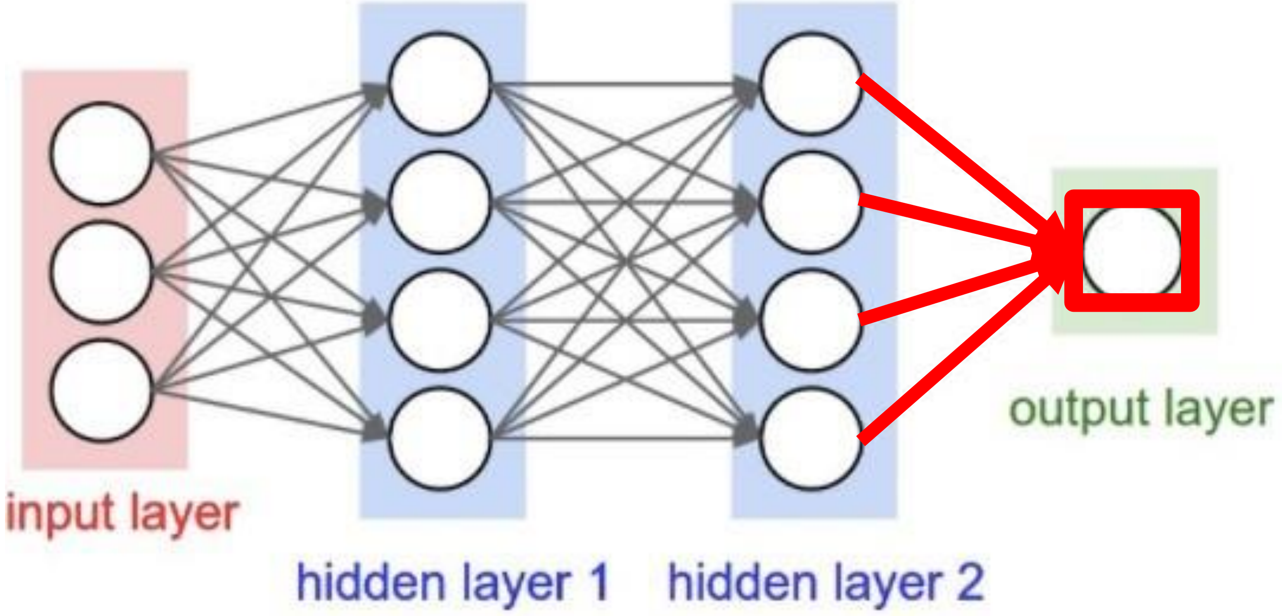


- Unit: takes as input a weighted sum and applies an activation function

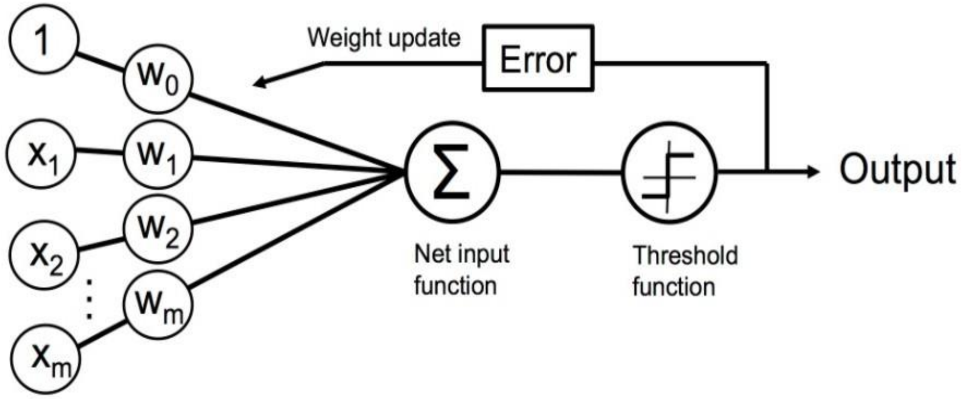
Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

Neural Network



- How does this relate to a perceptron?

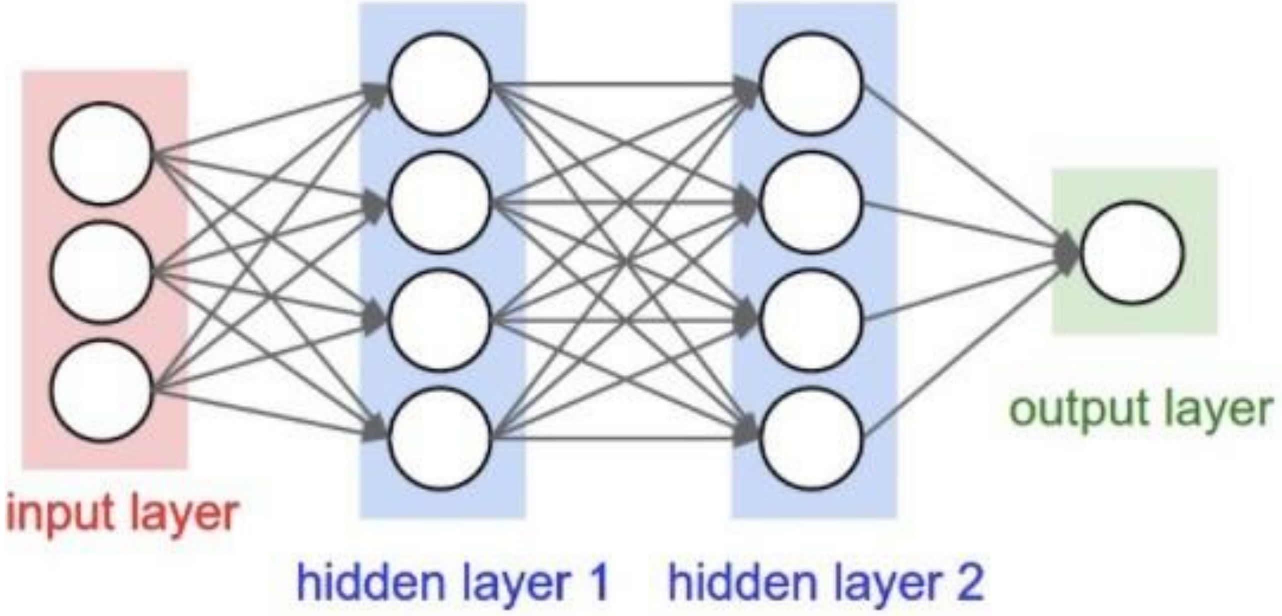


- Unit: takes as input a weighted sum and applies an activation function

Python Machine Learning; Raschka & Mirjalili

http://cs231n.github.io/neural-networks-1/

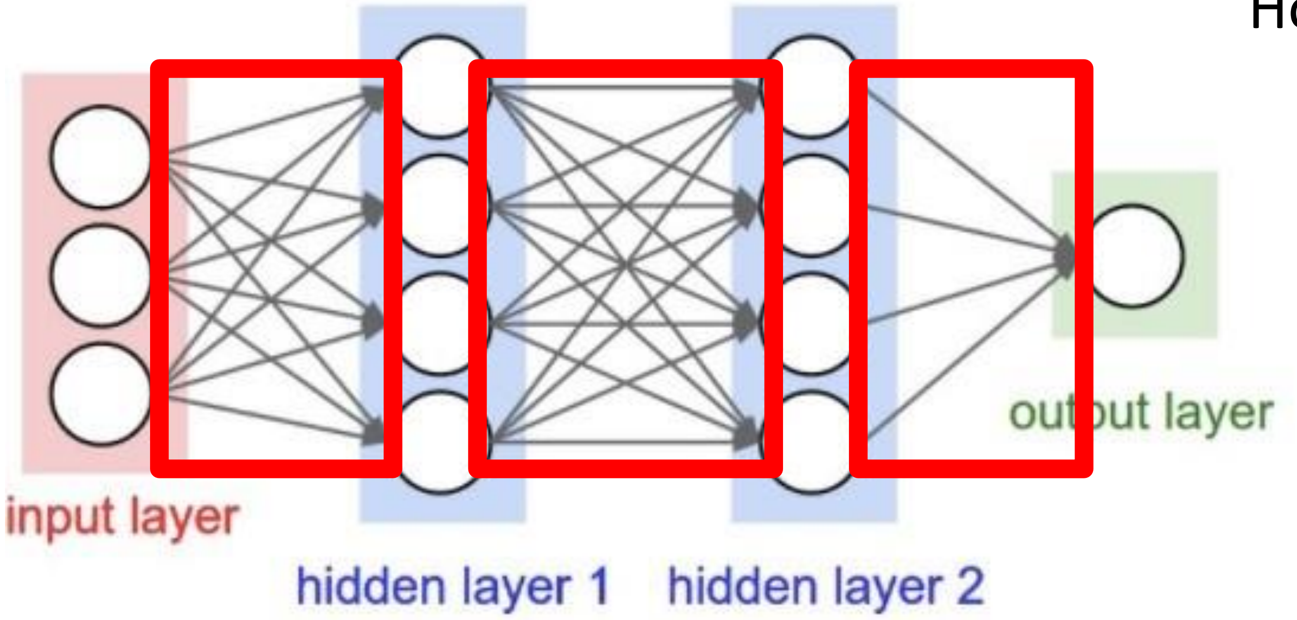
Neural Network



- **Training goal: learn model parameters**
- Layers are called "hidden" because algorithm decides how to use each layer to produce its output

http://cs231n.github.io/neural-networks-1/

Neural Network

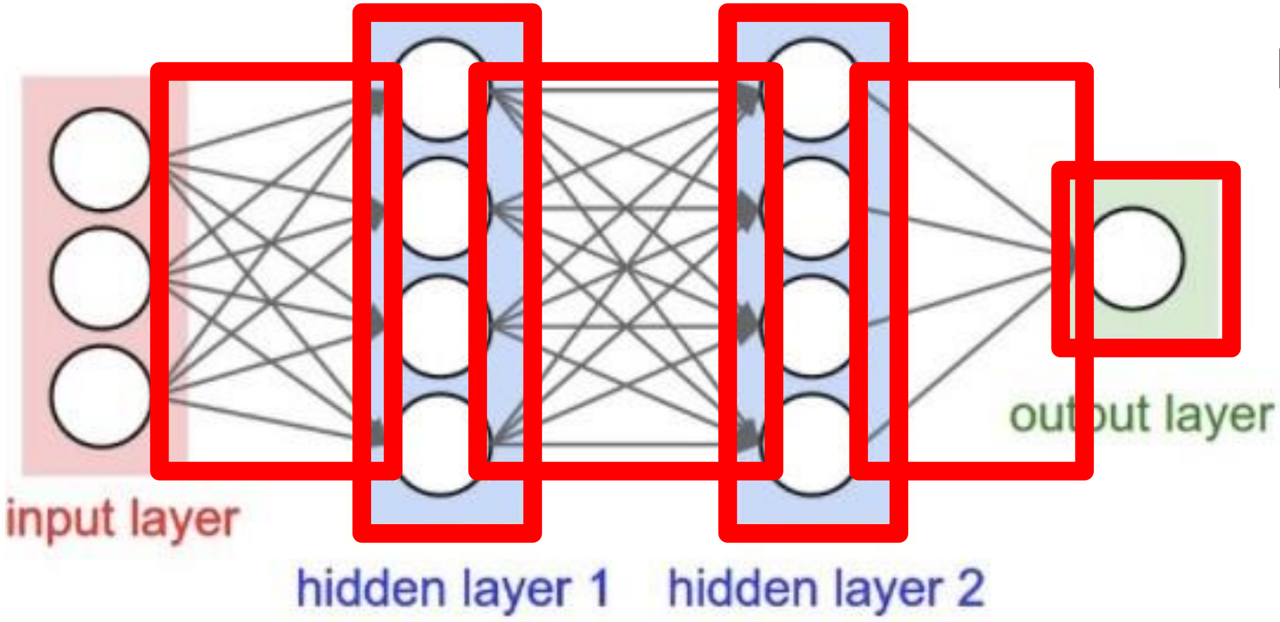


How many weights are in this model?

- Input to Hidden Layer 1:
  - 3x4 = 12
- Hidden Layer 1 to Hidden Layer 2:
  - 4x4 = 16
- Hidden Layer 2 to Output Layer
  - 4x1 = 4
- Total:
  - 12 + 16 + 4 = 32

http://cs231n.github.io/neural-networks-1/

Neural Network

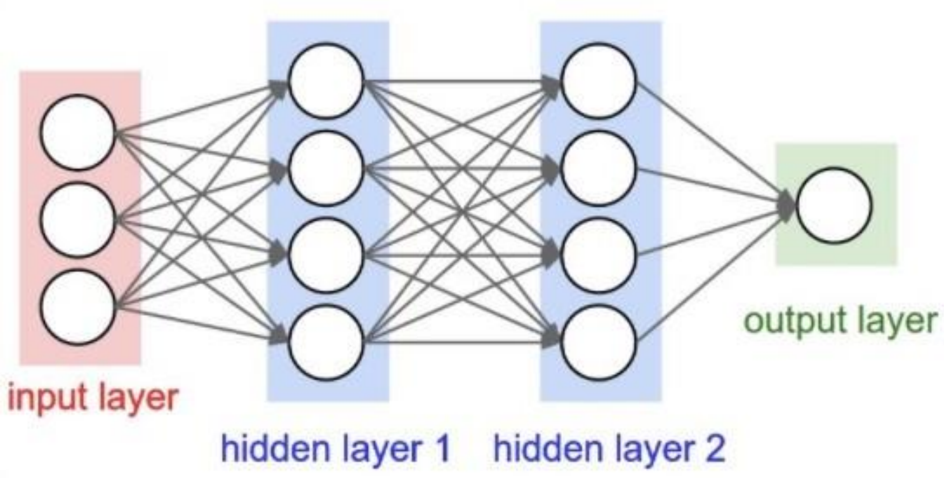


How many parameters are there to learn?

- Number of weights:
  - 32
- Number of biases:
  - 4 + 4 + 1 = 9
- Total
  - 41

http://cs231n.github.io/neural-networks-1/

Fully Connected, Feedforward Neural Networks



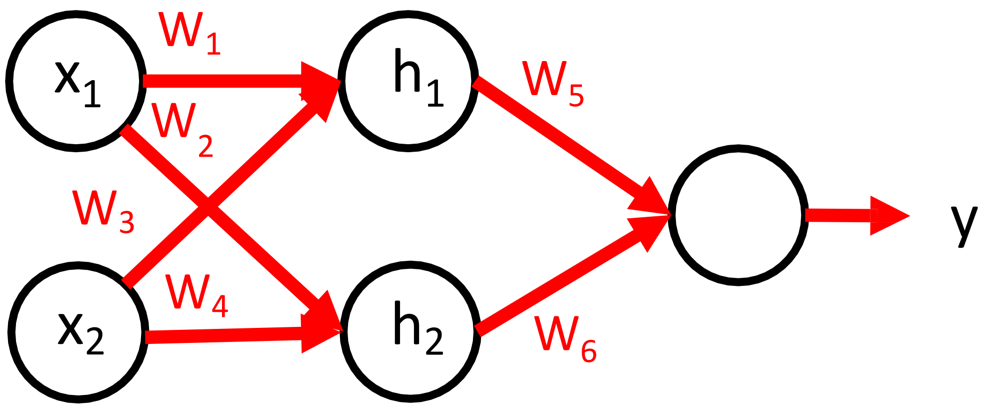
- What does it mean for a model to be fully connected?
  - Each unit provides input to each unit in the next layer
- What does it mean for a model to be feedforward?
  - Each layer serves as input to the next layer with no loops

http://cs231n.github.io/neural-networks-1/

Hidden Layers Alone Are NOT Enough to Model Non-Linear Functions

Key Observation: feedforward networks are just functions chained together

e.g.,



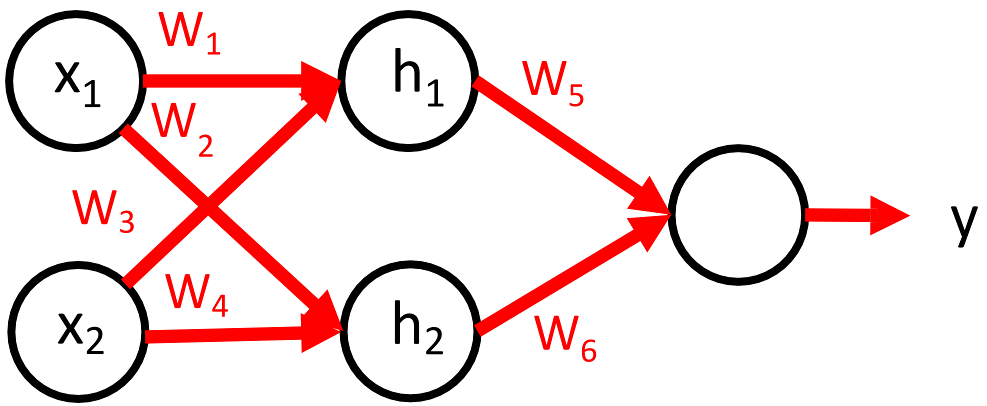
- What is function for h_1?
  - $h_1 = w_1x_1 + w_3x_2 + b_1$
- What is function for h_2?
  - $h_2 = w_2x_1 + w_4x_2 + b_2$
- What is function for y?
  - $y = h_1w_5 + h_2w_6 + b_3$
  - $y = (w_1x_1 + w_3x_2 + b_1)w_5 + (w_2x_1 + w_4x_2 + b_2)w_6 + b_3$
  - $\boxed{y = w_1w_5x_1 + w_3w_5x_2 + w_5b_1 + w_2w_6x_1 + w_4w_6x_2 + w_6b_2 + b_3}$

A chain of LINEAR functions at any depth is still a LINEAR function!

Hidden Layers Alone Are NOT Enough to Model Non-Linear Functions

Key Observation: feedforward networks are just functions chained together

e.g.,



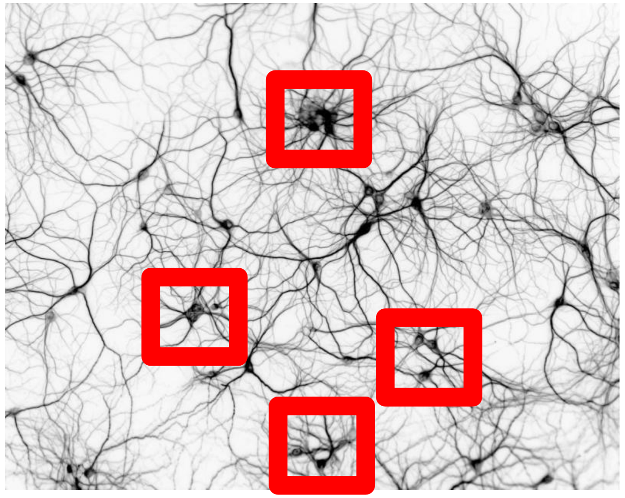
- What is function for h_1?
  - $h_1 = w_1x_1 + w_3x_2 + b_1$
- What is function for h_2?
  - $h_2 = w_2x_1 + w_4x_2 + b_2$
- What is function for y?
  - $y = \underbrace{h_1w_5} + \underbrace{h_2w_6} + b_3$

Constant x linear function = linear function

A chain of LINEAR functions at any depth is still a LINEAR function!

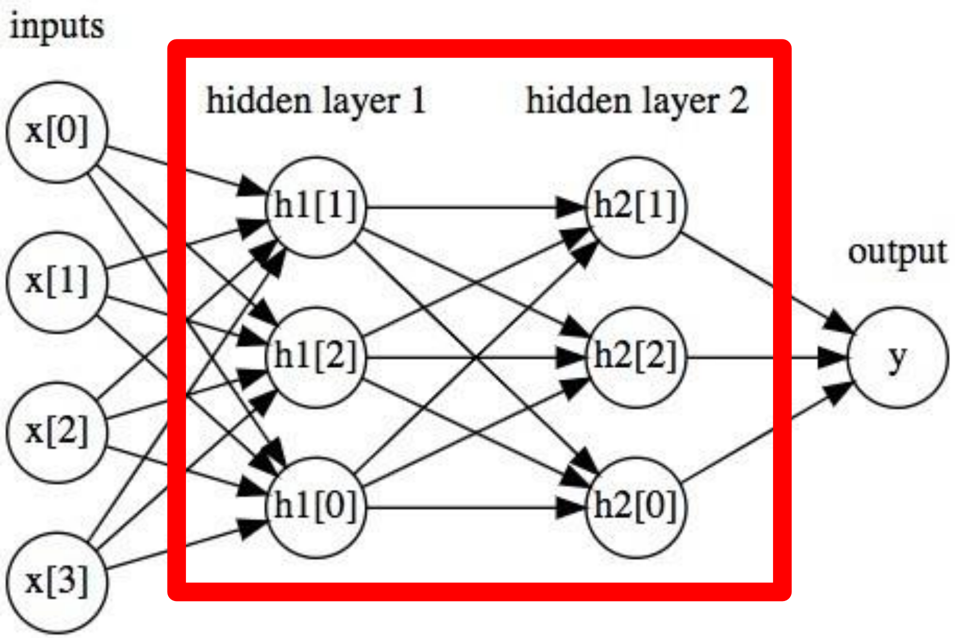
Key Idea: Use Connected Neurons to Non-linearly Transform Input into Useful Features for Predictions

Biological Neural Network:



http://www.rzagabe.com/2014/11/03/an-introduction-to-artificial-neural-networks.html

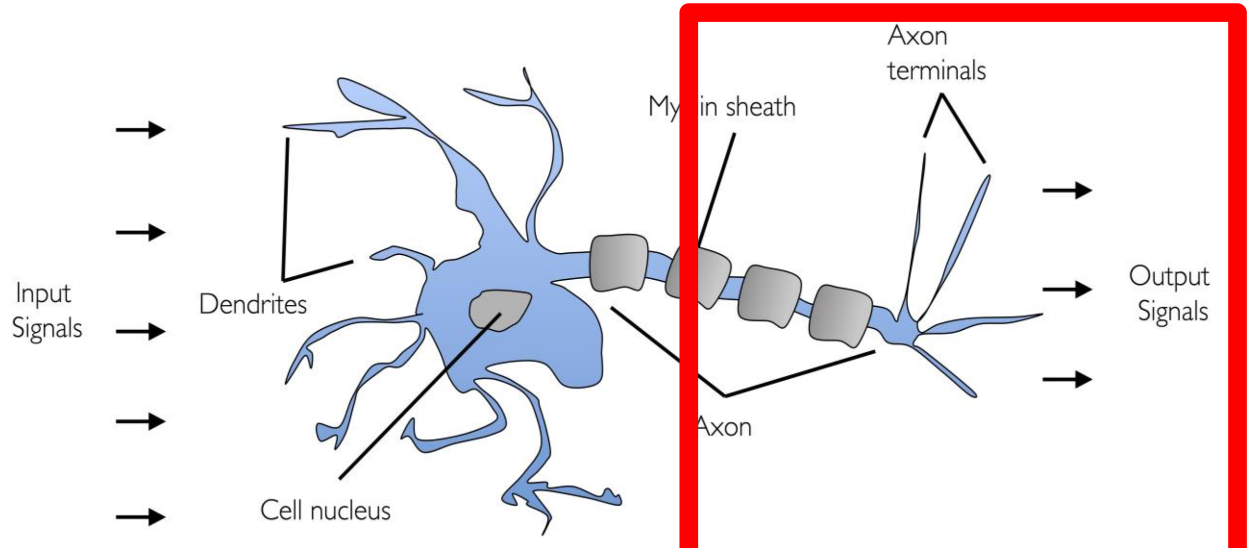
Artificial Neural Network:



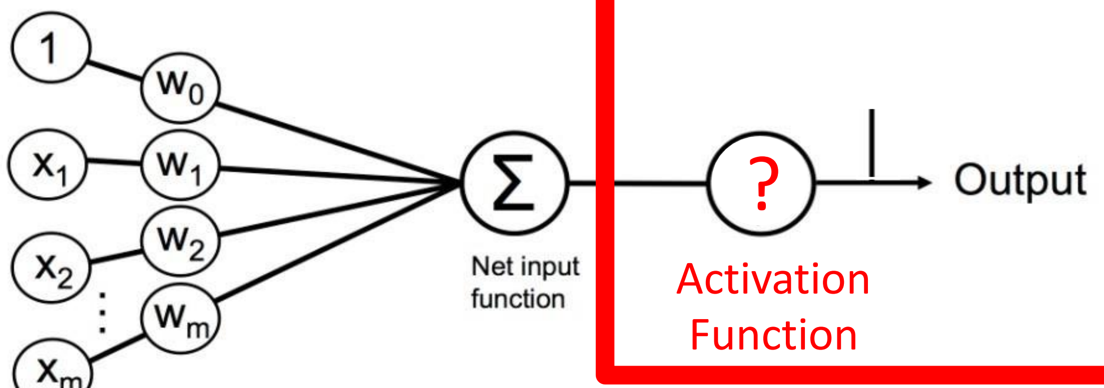
https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb

Key Idea: Use Connected Neurons to Non-linearly Transform Input into Useful Features for Predictions

Biological Neuron:



Artificial Neurons
(e.g., Perceptron):



Mimic a neuron firing, by having each unit apply a non-linear "activation" function to the weighted input

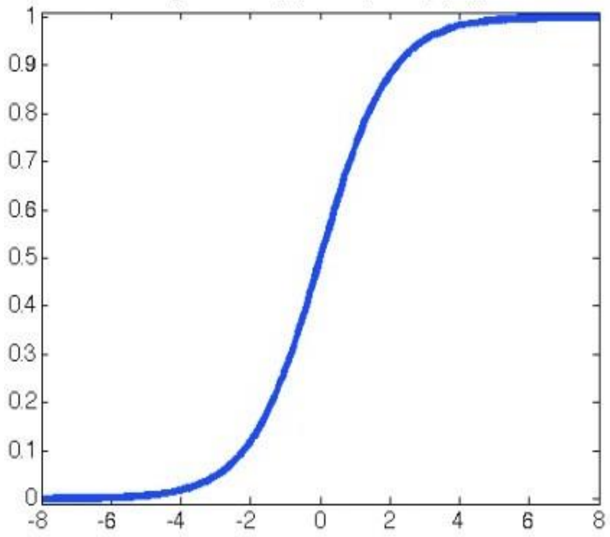
Python Machine Learning; Raschka & Mirjalili

Non-Linear Activation Functions

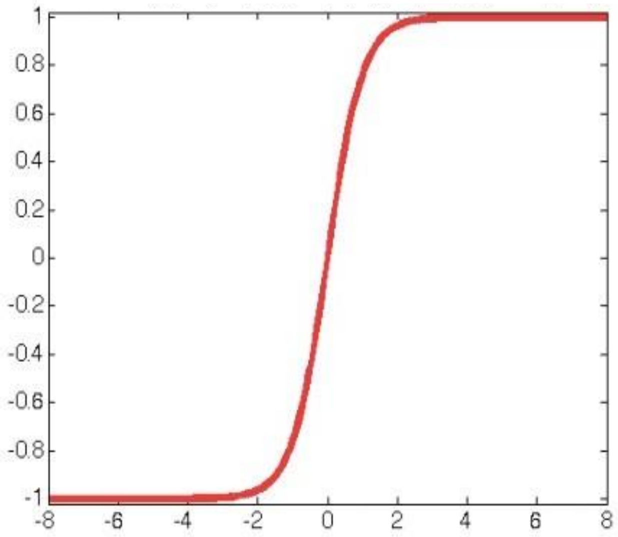
- Each unit applies a non-linear "activation" function to the weighted input to mimic a neuron firing

Computationally faster

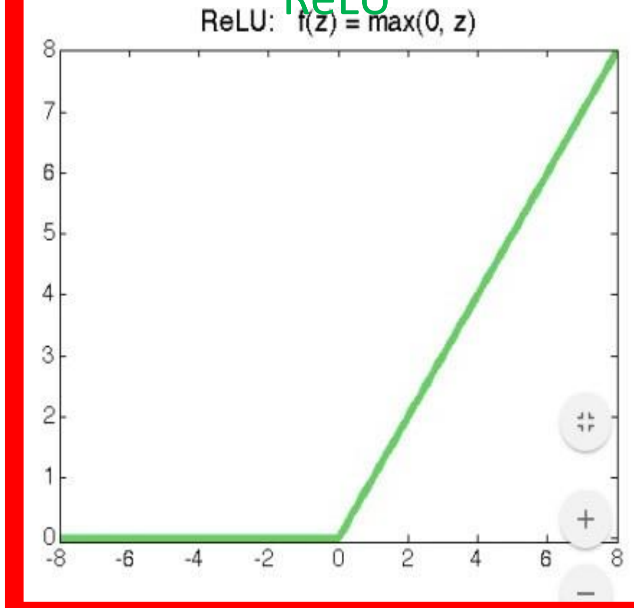
Sigmoid



Tanh



ReLU

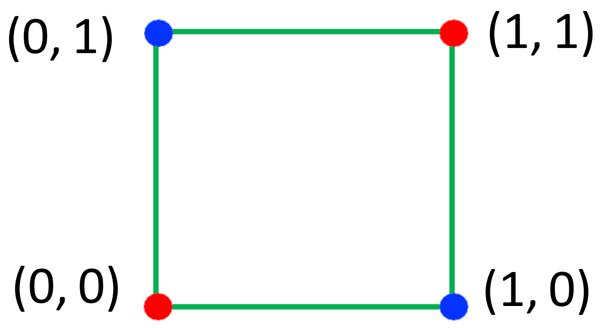


$$\sigma(z) = \frac{1}{1+\exp(-z)} \qquad \tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)} \qquad \mathrm{ReLU}(z) = \max(0, z)$$

http://www.cs.utoronto.ca/~fidler/teaching/2015/slides/CSC411/10_nn1.pdf

Non-Linear Example: Revisiting XOR problem

- Non-linear function: separate 1s from 0s:

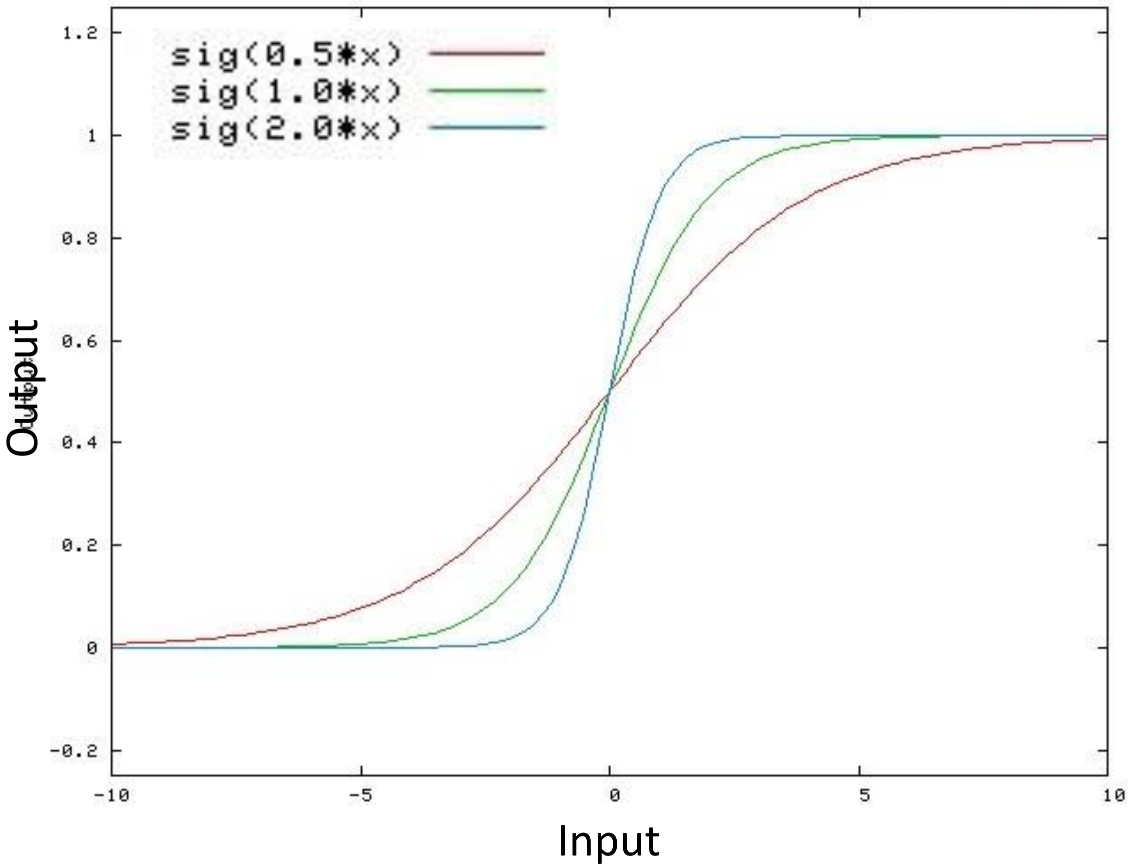


- Approach: ReLU activation function ($\mathrm{ReLU}(z) = \max(0, z)$) with these parameters:

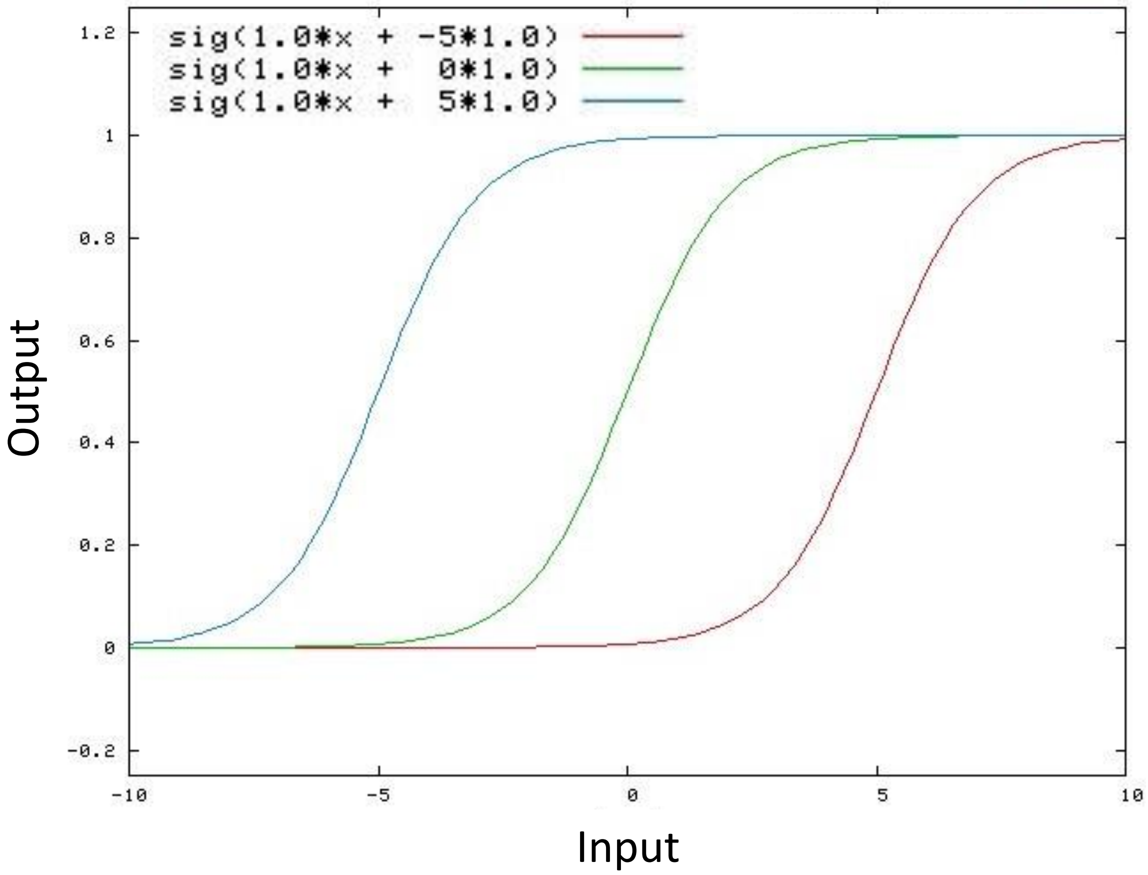
Neural networks can solve XOR problem...
and so model non-linear functions!

Activation Functions and Model Parameters (e.g., Sigmoid)

Weights determine curvature:

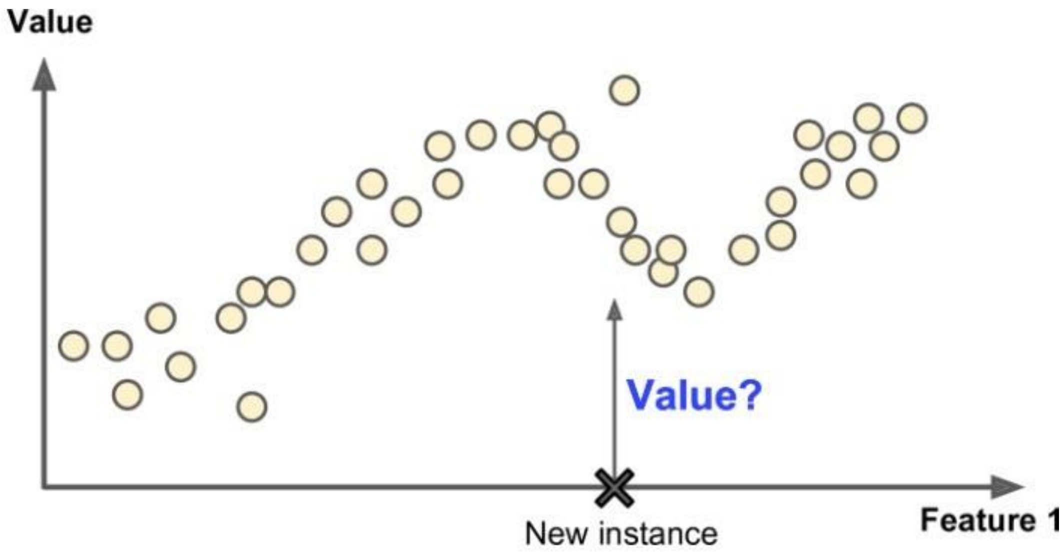


Biases determine shifted position:

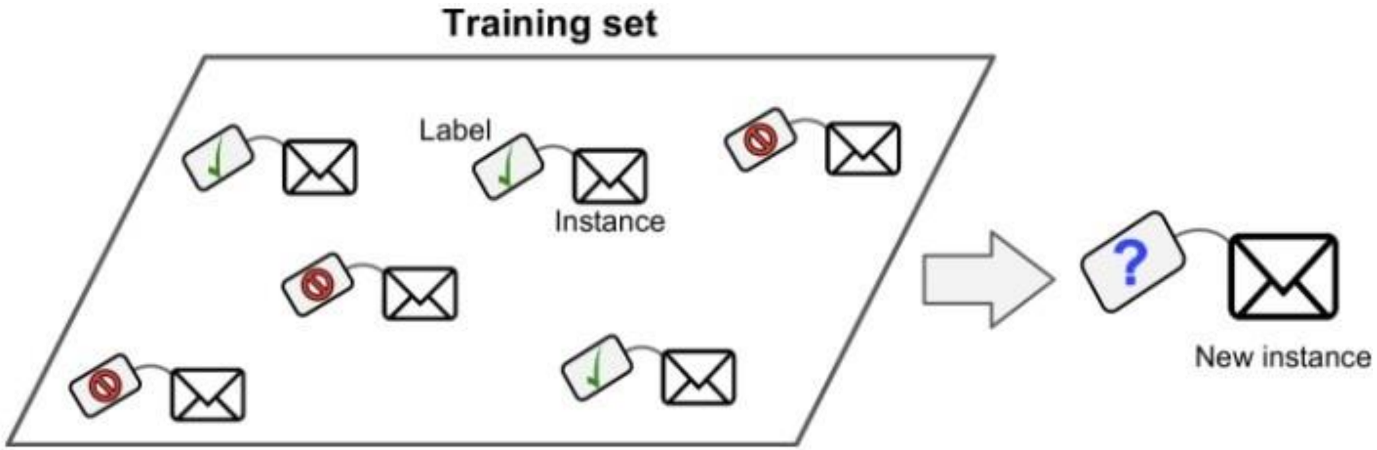


Desired Output Driven by Task

Regression
(predict **continuous** value)

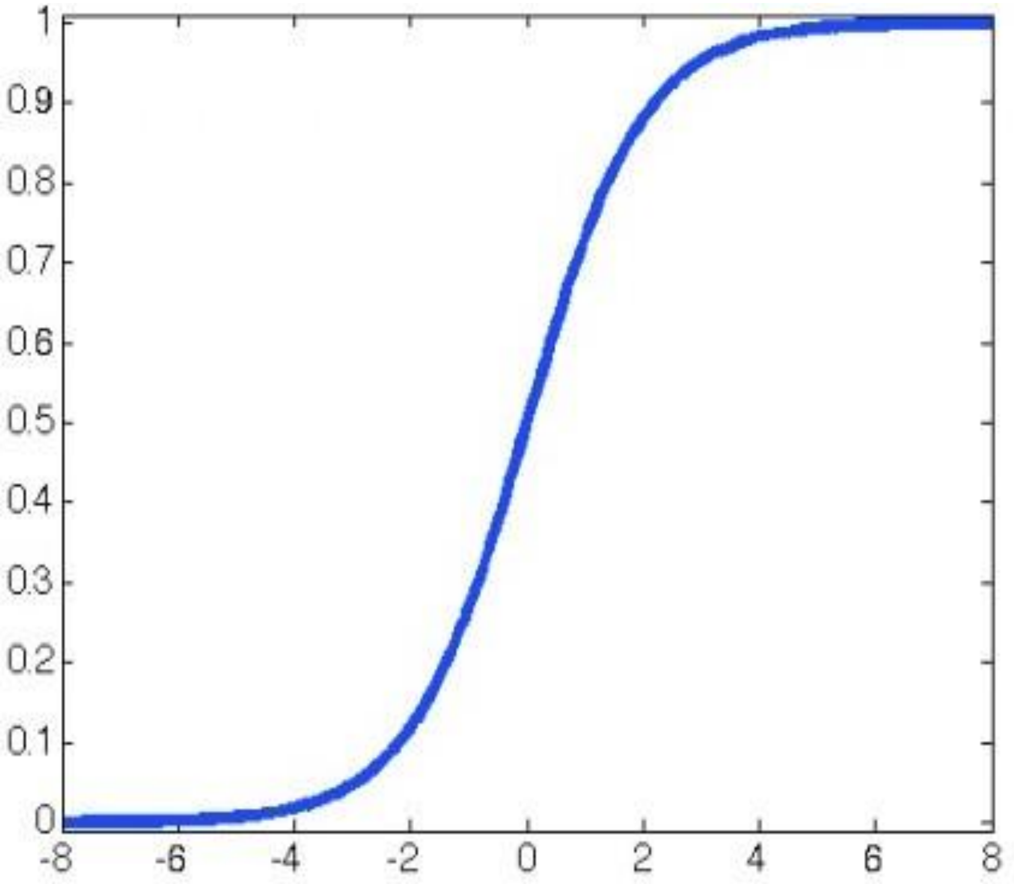


Classification
(predict **discrete** value)



Hands-on Machine Learning with Scikit-Learn & TensorFlow, Aurelien Geron

Sigmoid (for Binary Classification)



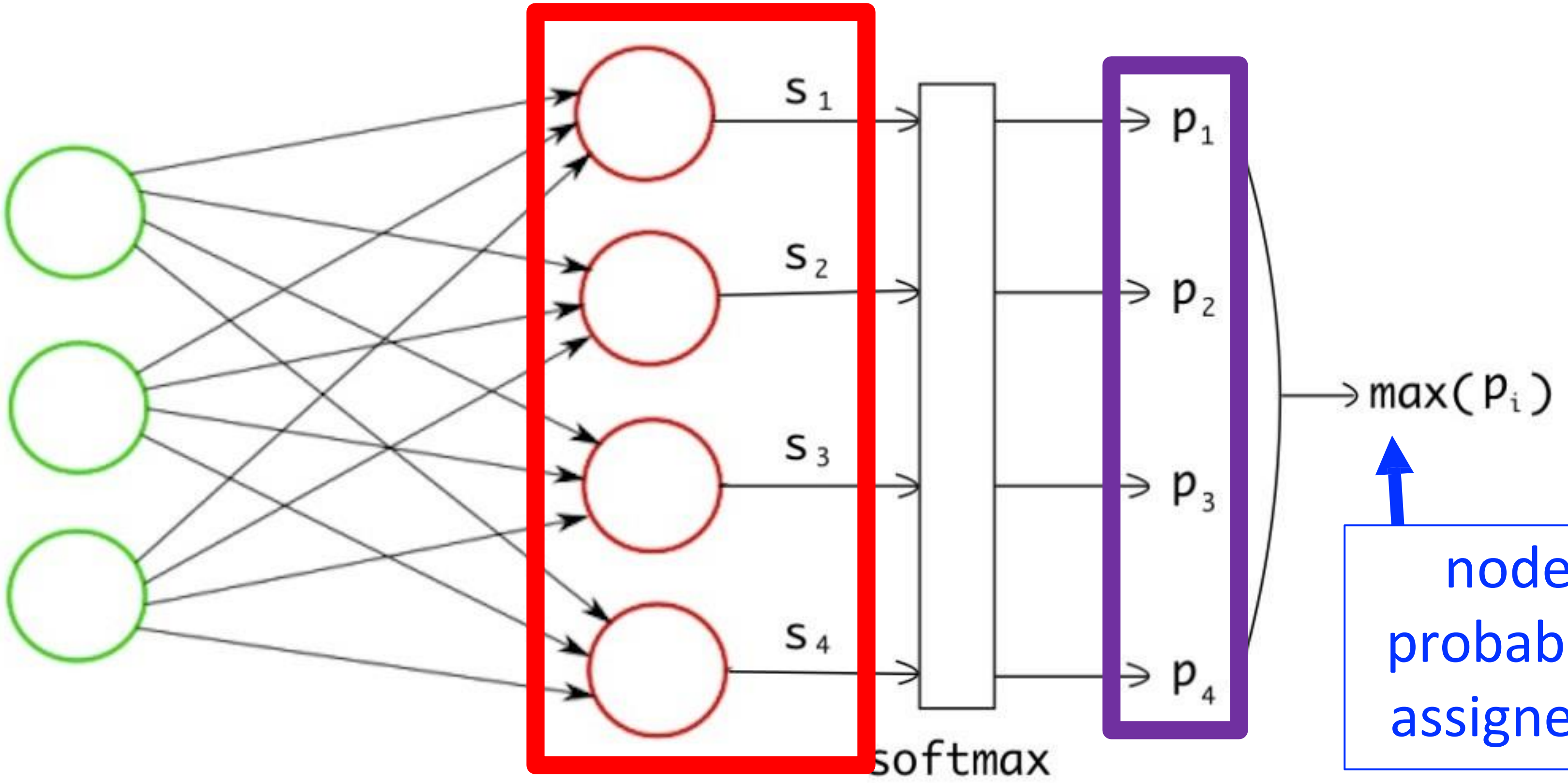
$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

If >= 0.5, output 1;

Else, outputs 0

Softmax (for Multiclass Classification)

Converts vector of scores into a probability distribution that sums to 1; e.g.,

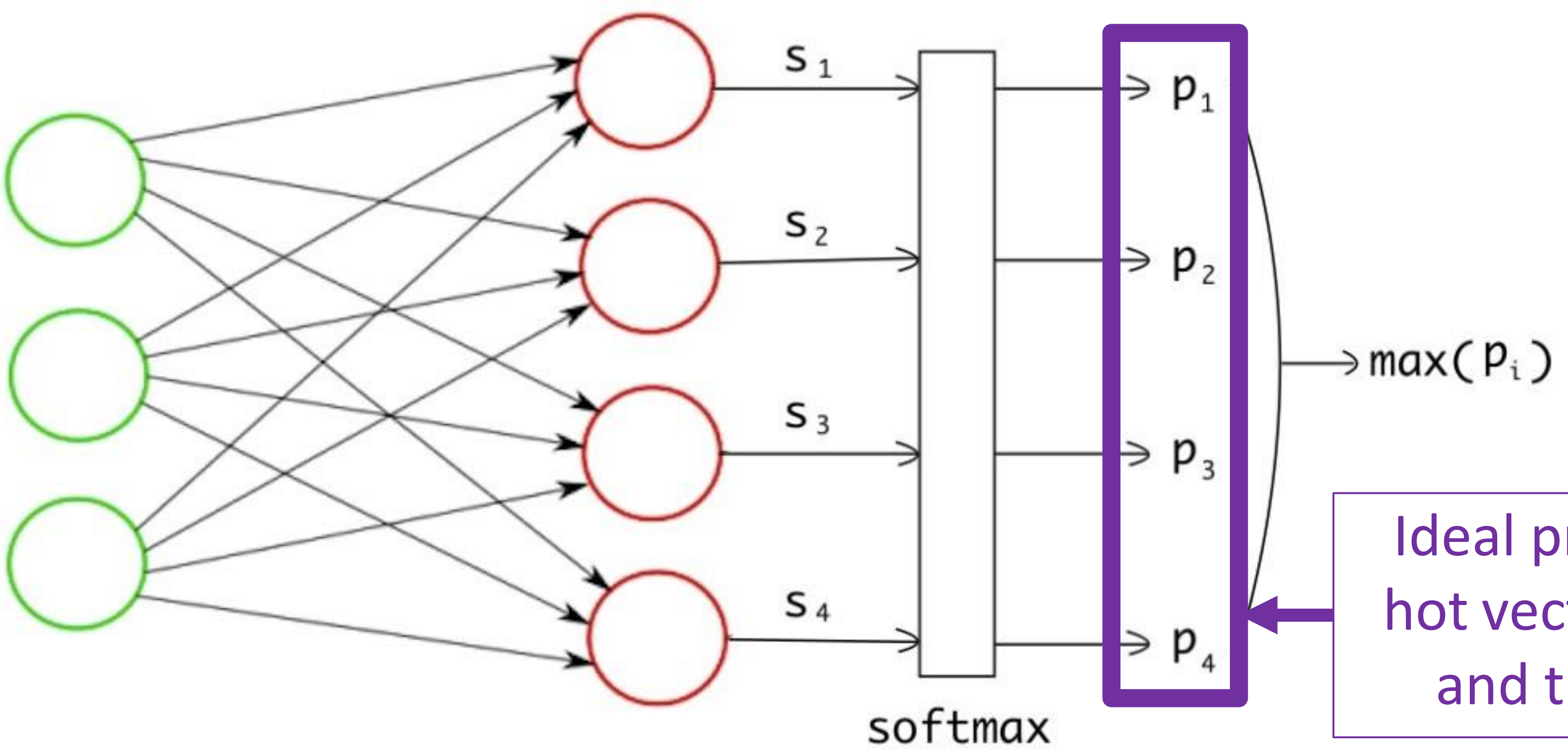


node with largest probability is the class assigned to the image

Figure Source: https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede

Softmax (for Multiclass Classification)

Converts vector of scores into a probability distribution that sums to 1; e.g.,



Ideal prediction; 1 hot vector of one 1 and the rest 0s

Figure Source: https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede

Softmax (for Multiclass Classification)

Converts vector of scores into a probability distribution that sums to 1

Get rid of negative values while preserving original order of scores; e causes negative scores to become slightly larger than 0 while positive values grow exponentially (choosing *e* rather than another exponent base simplifies math during training)

$$\sigma(\mathbf{z})_i = \frac{\boxed{e}^{\boxed{z_i}}}{\boxed{\sum_{j=1}^{K} \boxed{e}^{z_j}}}$$

i = 1, …, K

Number of classes

Want to divide each node's score by sum of all entries to make them sum to 1 (normalization)

Useful tutorial: https://towardsdatascience.com/exploring-the-softmax-function-578c8b0fb15

Softmax (for Multiclass Classification)

Converts vector of scores into a probability distribution that sums to 1; e.g.,

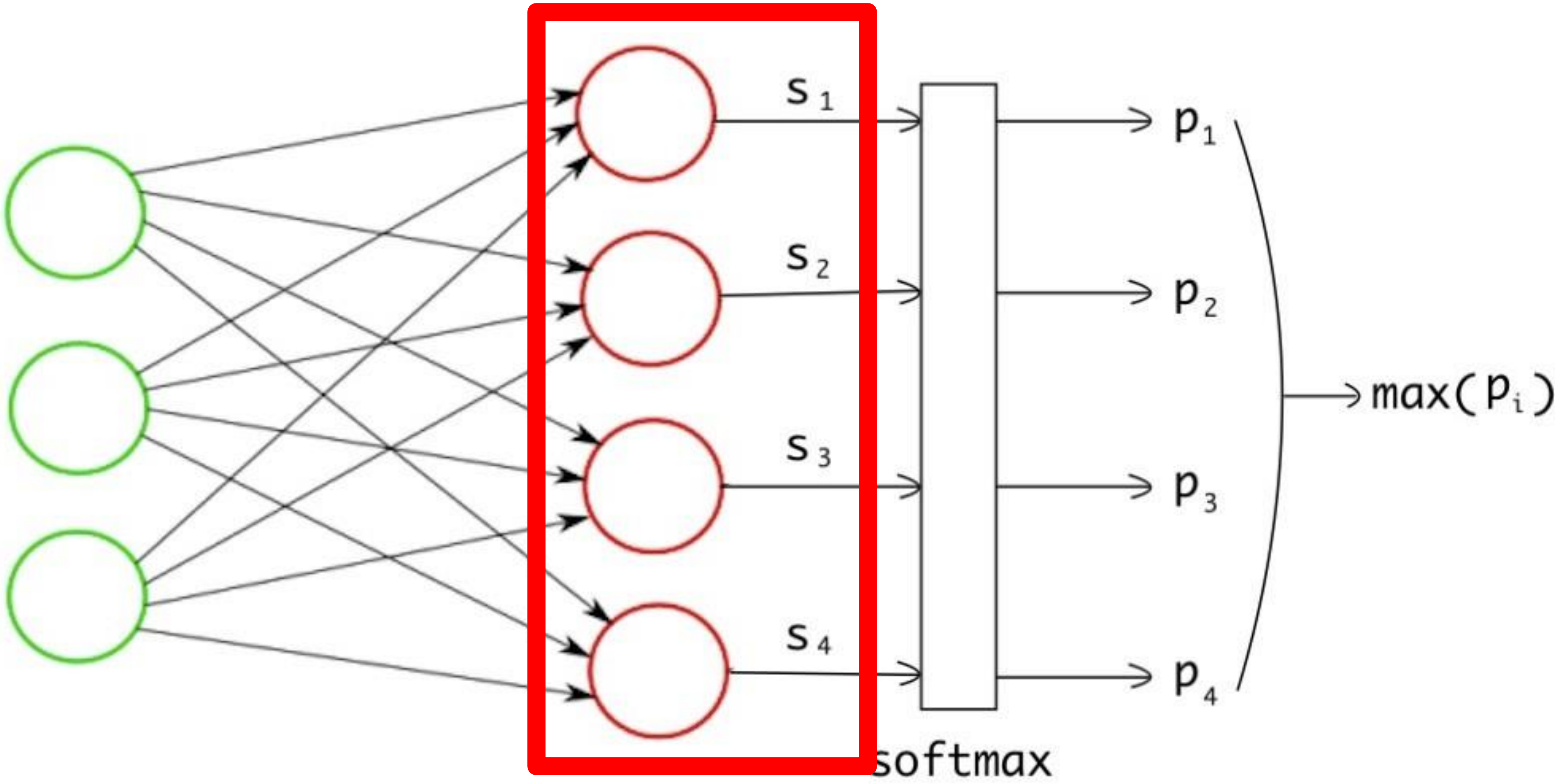


Figure Source: https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede

Softmax (for Multiclass Classification)

Converts vector of scores into a probability distribution that sums to 1; e.g.,

	Scoring Function
Dog	-3.44
Cat	1.16
Boat	-0.81
Airplane	3.91

Softmax (for Multiclass Classification)

Converts vector of scores into a probability distribution that sums to 1; e.g.,

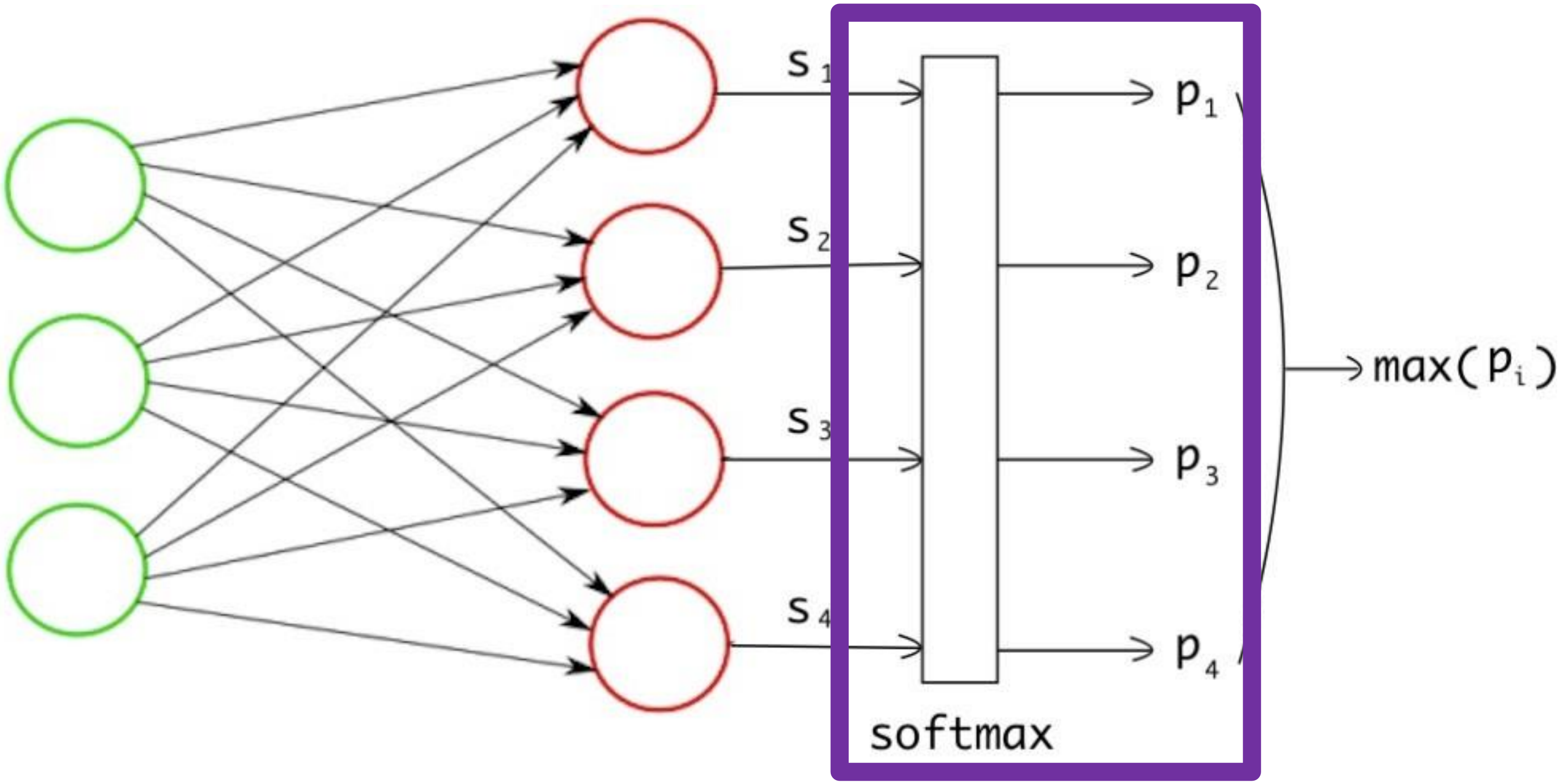


Figure Source: https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede

Softmax (for Multiclass Classification)

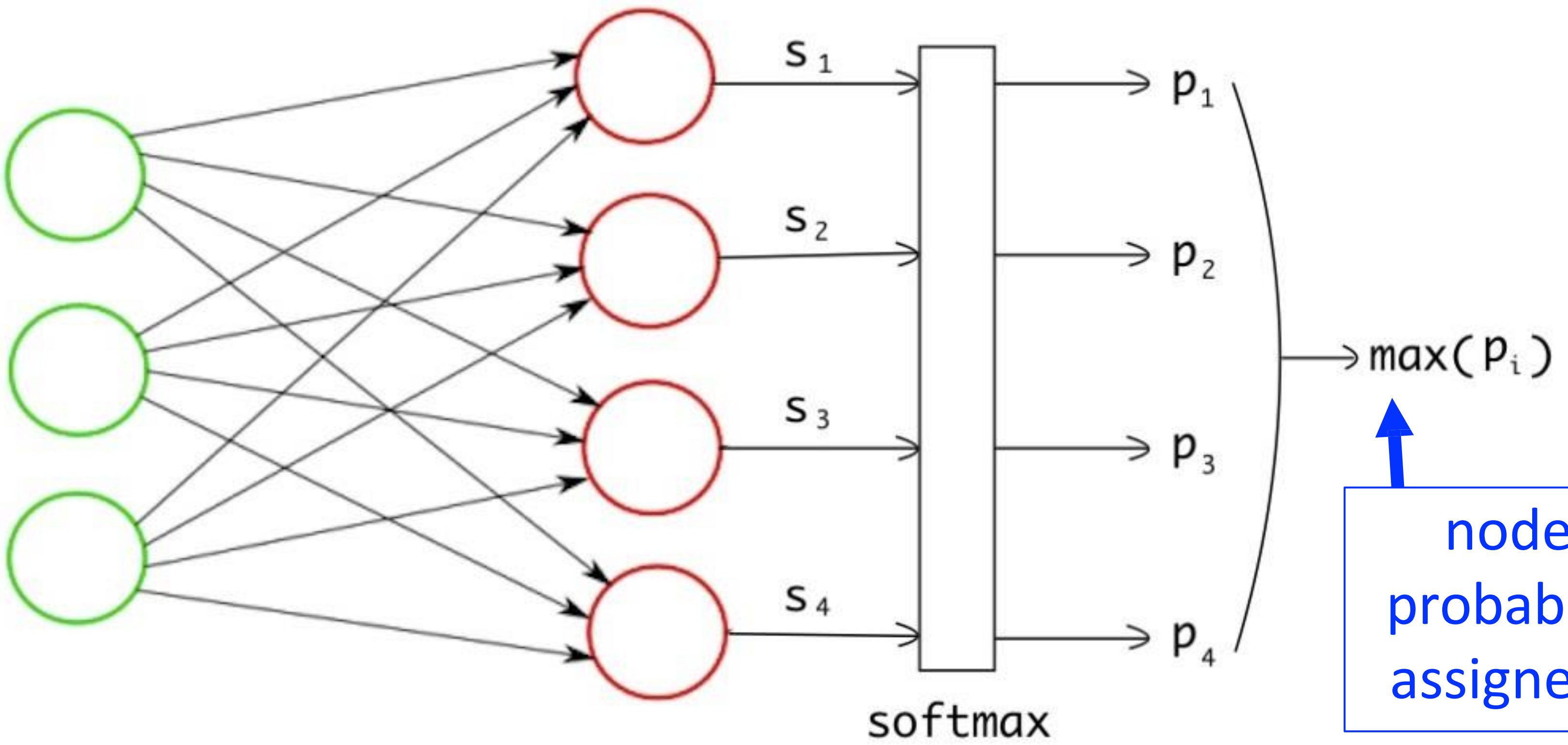
Converts vector of scores into a probability distribution that sums to 1; e.g.,

$$e^{z_i} \qquad \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

	Scoring Function	Unnormalized Probabilities	Normalized Probabilities
Dog	-3.44	0.0321	0.0006
Cat	1.16	3.1899	0.0596
Boat	-0.81	0.4449	0.0083
Airplane	3.91	49.8990	0.9315

Softmax (for Multiclass Classification)

Converts vector of scores into a probability distribution that sums to 1; e.g.,



node with largest probability is the class assigned to the image

Figure Source: https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede

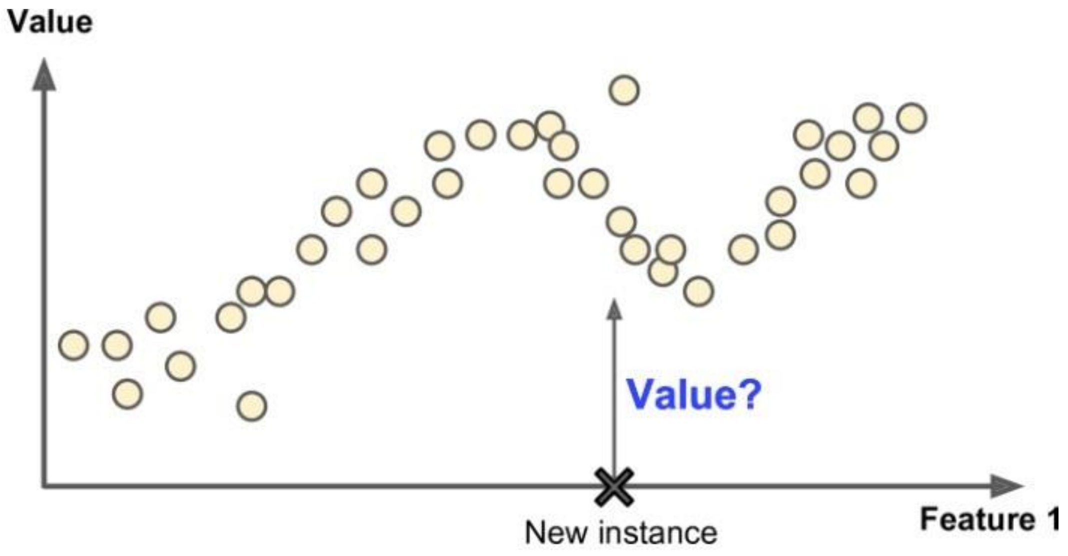
Softmax (for Multiclass Classification)

Converts vector of scores into a probability distribution that sums to 1; e.g.,

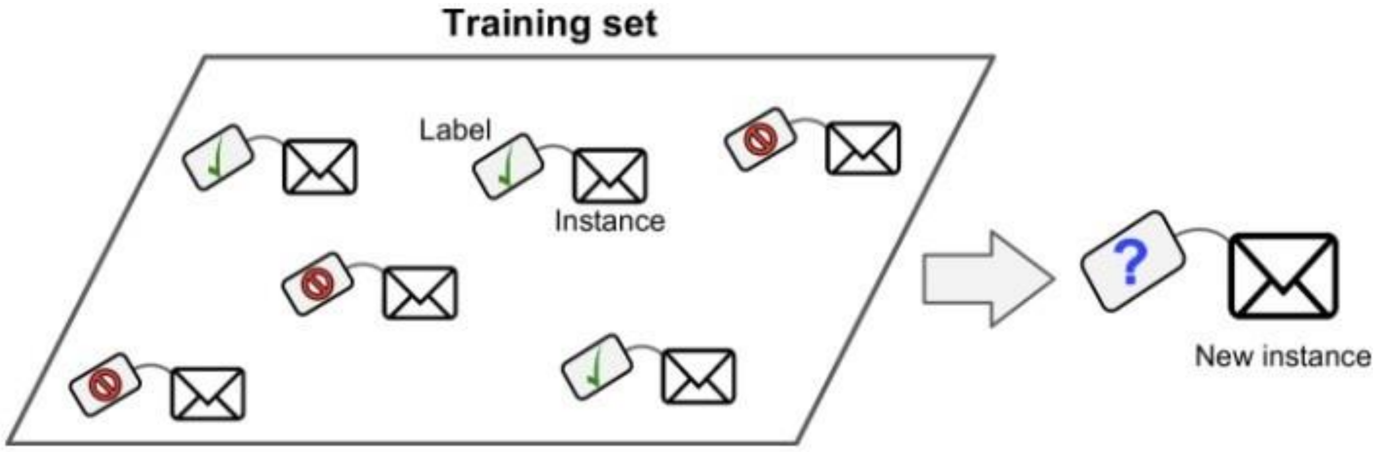
	Scoring Function	Unnormalized Probabilities	Normalized Probabilities
Dog	-3.44	0.0321	0.0006
Cat	1.16	3.1899	0.0596
Boat	-0.81	0.4449	0.0083
Airplane	3.91	49.8990	0.9315

Desired Output Driven by Task

Regression
(predict **continuous** value)



Classification
(predict **discrete** value)



Desired Output Driven by Task