Feedforward Neural Networks and Objective Functions

Danna Gurari
University of Colorado Boulder
Spring 2025

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2024/

Review

- Last lecture:
  - Supervised learning: approach to develop a model
  - Artificial neuron model: basic unit of neural networks
  - Evaluating classification models

- Assignments (Canvas):
  - Problem set 1 due in one week

- Questions?

Today's Topics

- Motivation for neural networks: need non-linear models
- Neural networks' basic ingredients: hidden layers and activation units
- Neural networks' support for diverse problems: output units
- Objective function: what a model should learn
- Programming tutorial

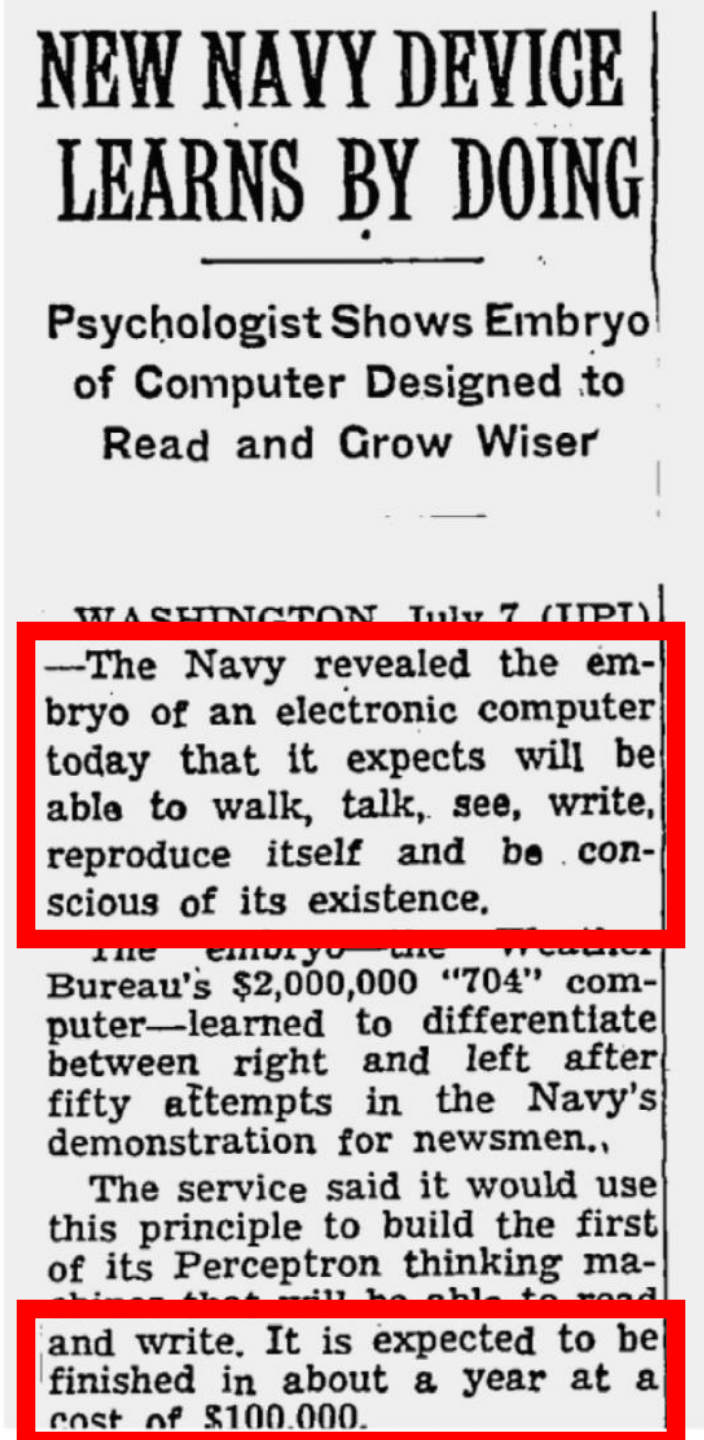
Today's Topics

- Motivation for neural networks: need non-linear models
- Neural networks' basic ingredients: hidden layers and activation units
- Neural networks' support for diverse problems: output units
- Objective function: what a model should learn
- Programming tutorial

Recall: Vision for Perceptrons

New York Times article, July 8, 1958 :

https://www.nytimes.com/1958/07/08/archives/new-navy-device-learns-by-doing-psychologist-shows-embryo-of.html



NEW NAVY DEVICE LEARNS BY DOING

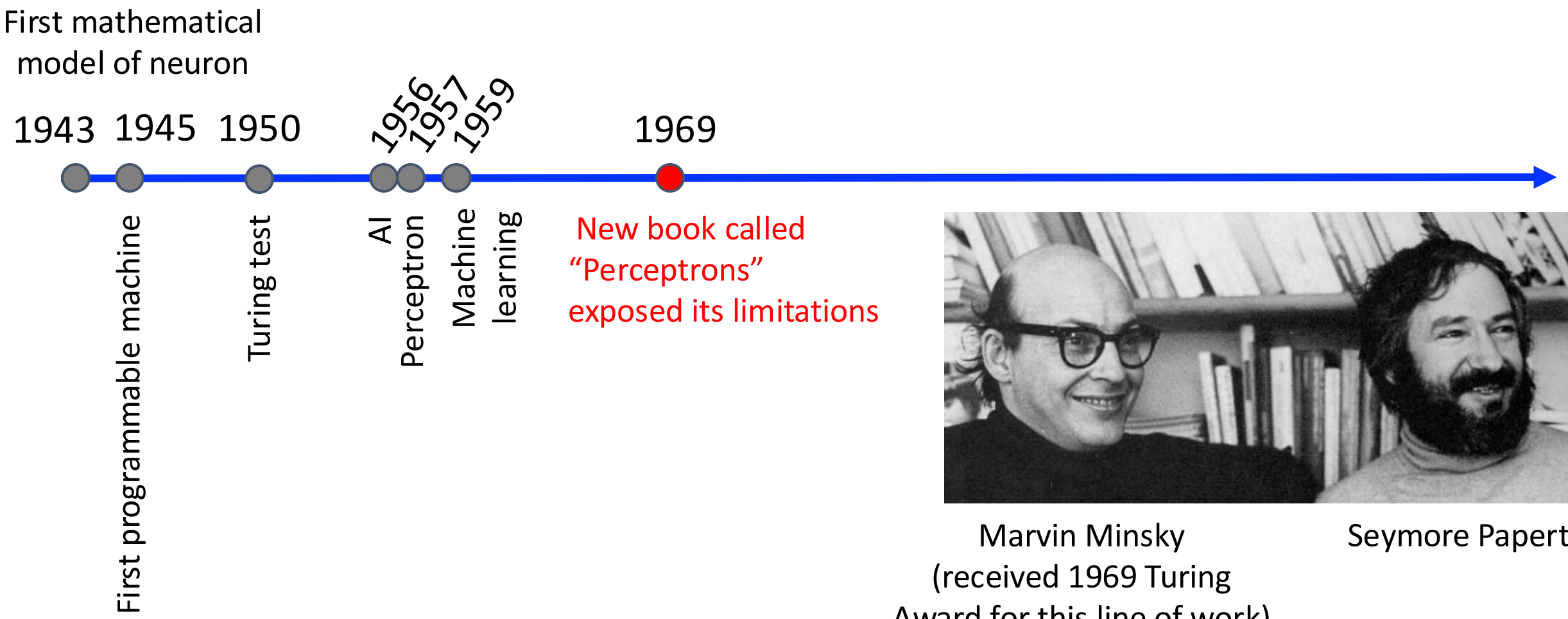
Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Historical Context: Artificial Neurons' Downfall



Marvin Minsky (received 1969 Turing Award for this line of work)

Seymore Papert

https://www.wyso.org/2016-08-05/remembering-a-thinker-who-thought-about-thinking

Fall of Perceptron (Artificial Neuron)

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	?

Fall of Perceptron (Artificial Neuron)

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	?

Fall of Perceptron (Artificial Neuron)

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
0	1	?
1	0	?
1	1	?

Fall of Perceptron (Artificial Neuron)

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
0	1	1
1	0	?
1	1	?

Fall of Perceptron (Artificial Neuron)

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
0	1	1
1	0	1
1	1	?

Fall of Perceptron (Artificial Neuron)

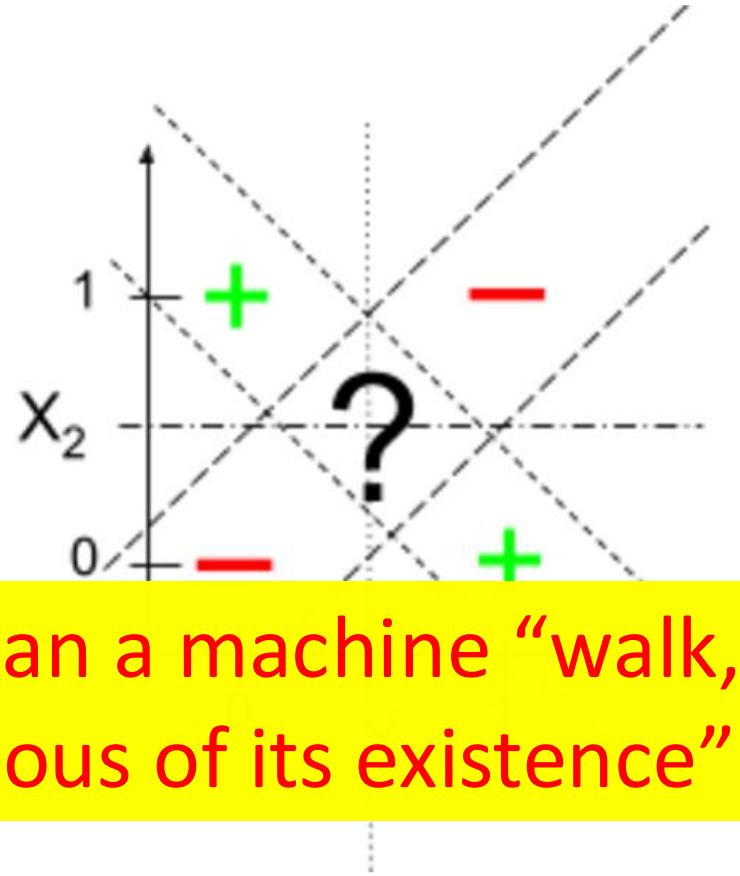
XOR = “Exclusive Or”

- 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
0	1	1
1	0	1
1	1	0

Fall of Perceptron (Artificial Neuron)

A Perceptron is a linear function, and so cannot solve XOR (by separating 1s from 0s):



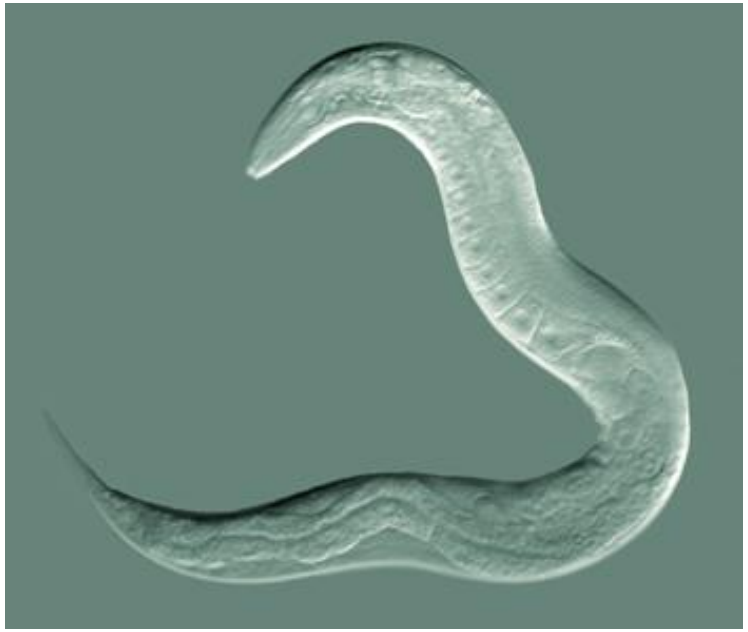
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

How can a machine “walk, talk, see, write, reproduce itself and be conscious of its existence” when it can’t solve the XOR problem?

Today's Topics

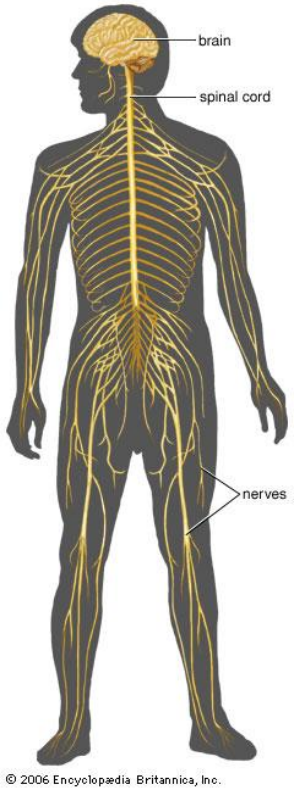
- Motivation for neural networks: need non-linear models
- Neural networks' basic ingredients: hidden layers and activation units
- Neural networks' support for diverse problems: output units
- Objective function: what a model should learn
- Programming tutorial

Solution: Solve Non-Linear Problems with Connected Neurons (i.e., Neural Networks)



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

Nematode worm: 302 neurons

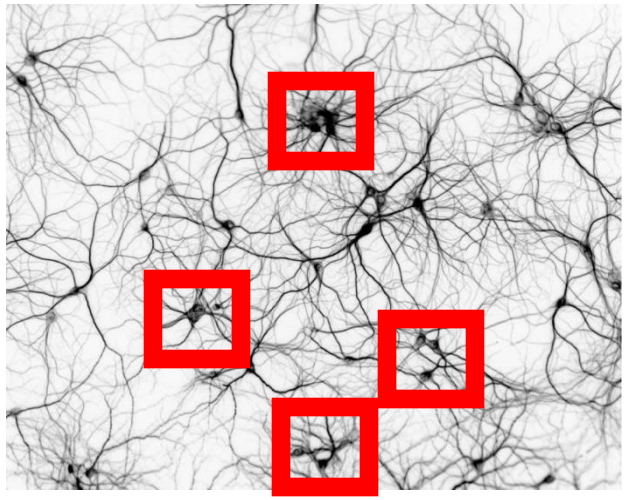


https://www.britannica.com/science/human-nervous-system

Human: ~100,000,000,000 neurons

Solution: Solve Non-Linear Problems with Connected Neurons (i.e., Neural Networks)

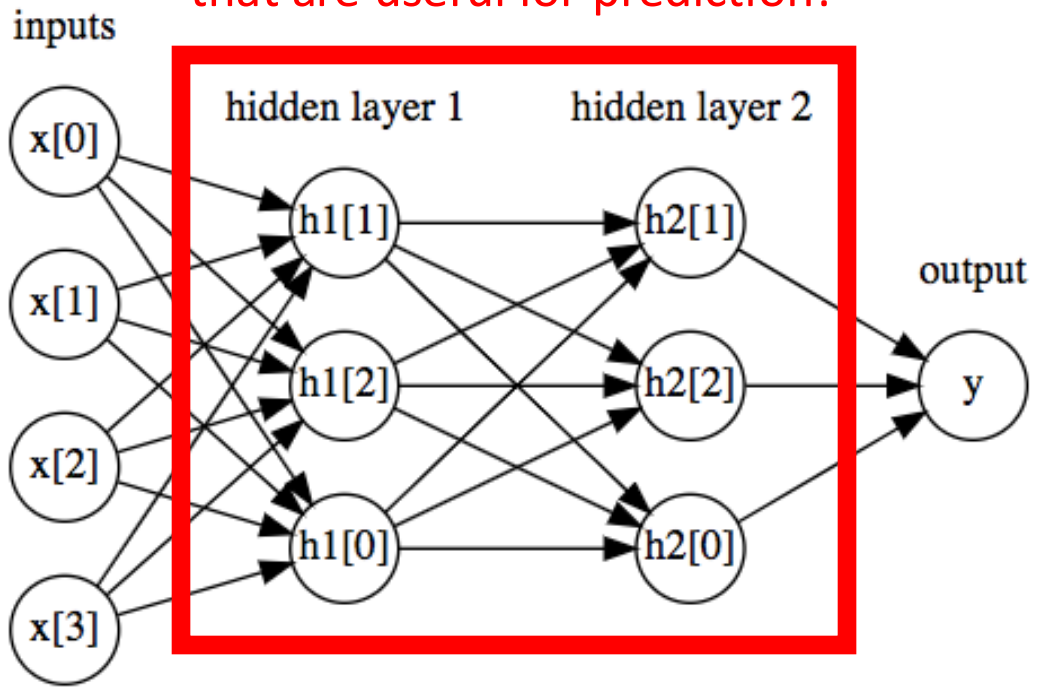
Biological Neural Network:



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

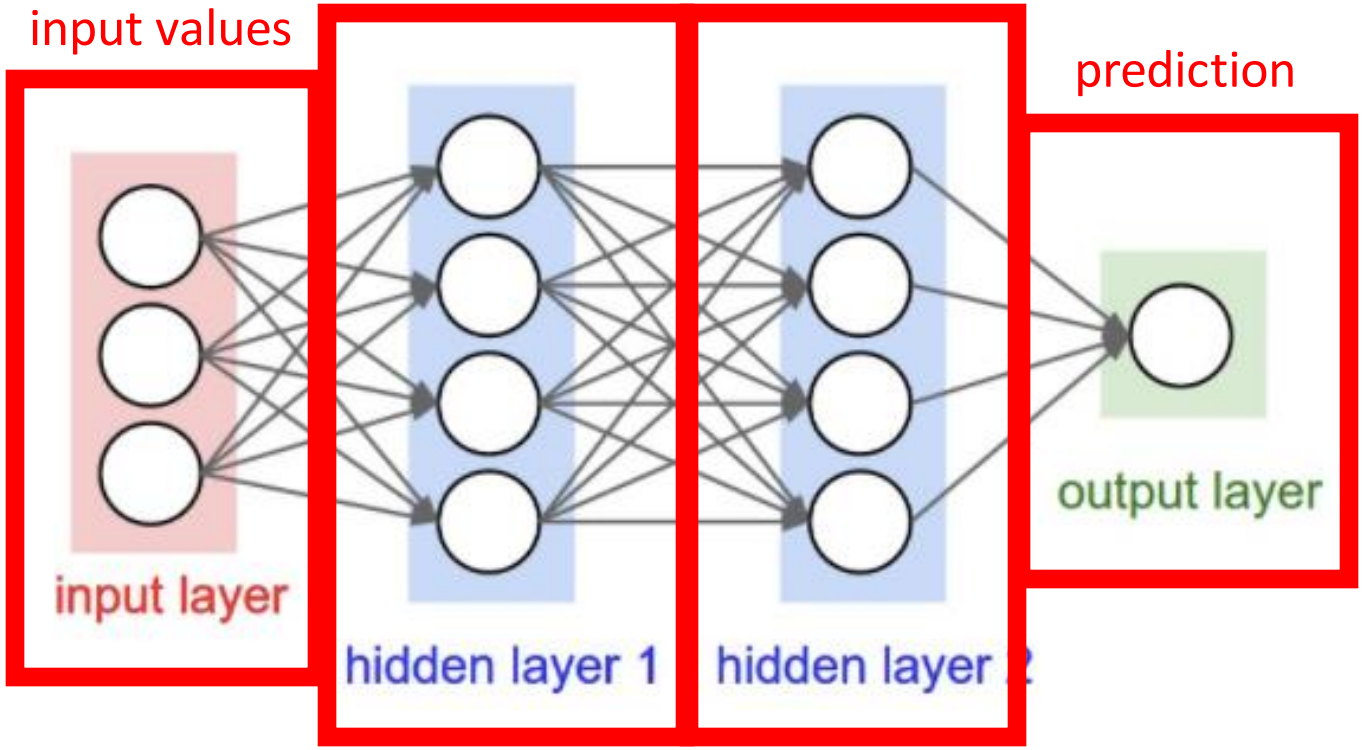
Artificial Neural Network:

Transforms input into features that are useful for prediction!



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

Neural Network: Hidden Layers

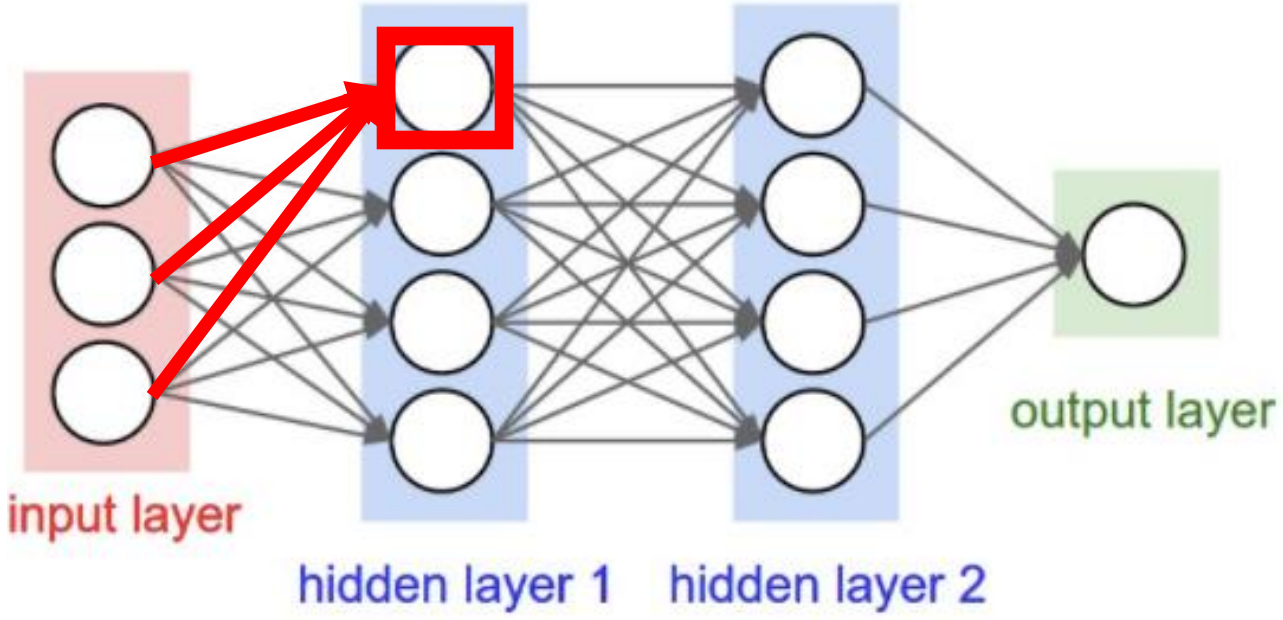


each “hidden layer” uses outputs of units (i.e., neurons) and provides them as inputs to other units (i.e., neurons)

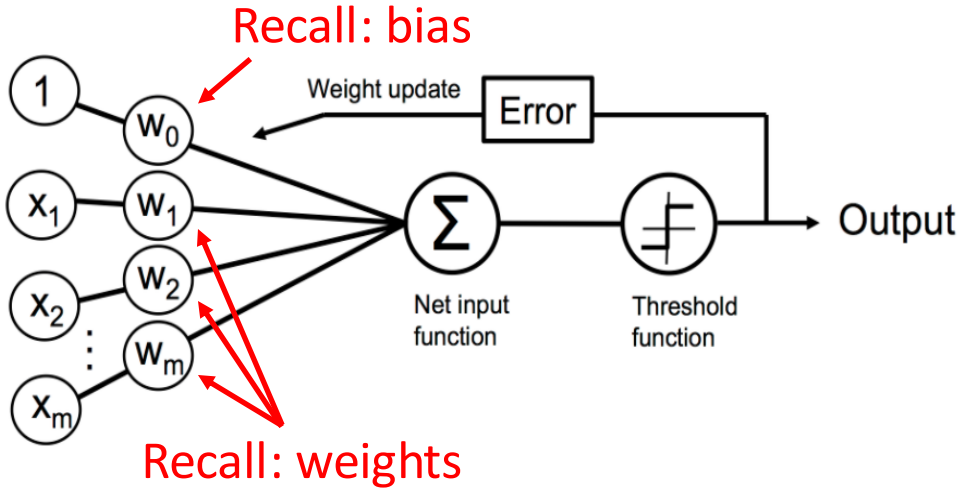
This is a 3-layer neural network (i.e., count number of hidden layers plus output layer)

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

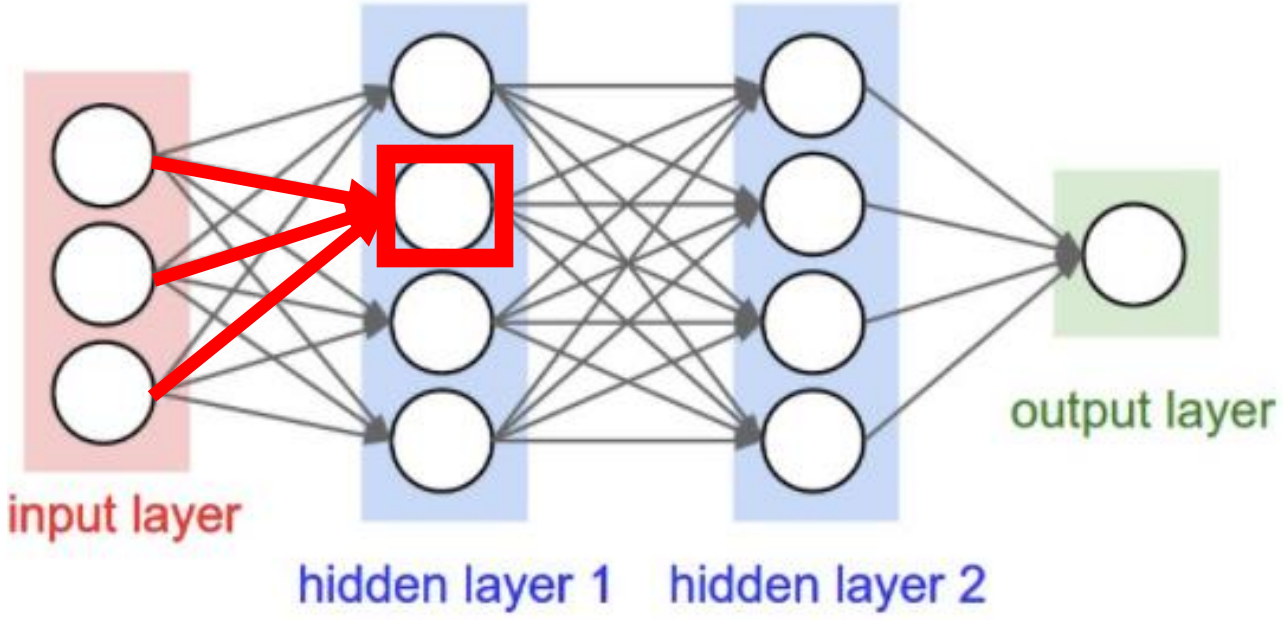


- Unit: computes a weighted sum and applies an activation function

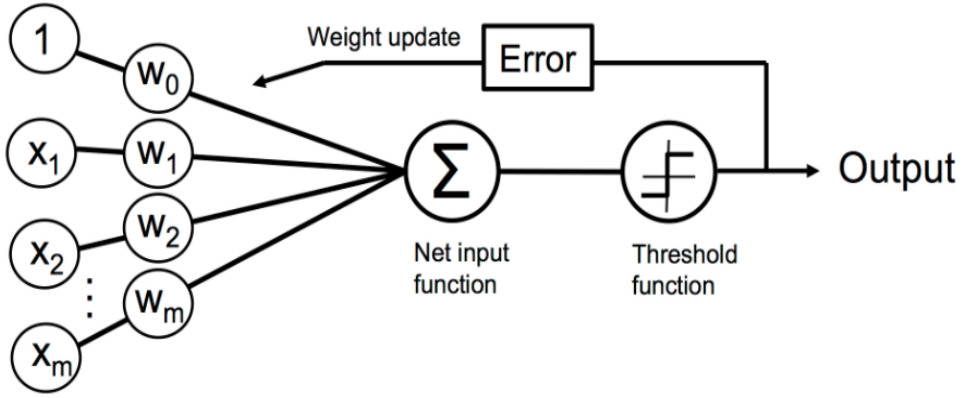
Python Machine Learning; Raschka & Mirjalili

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

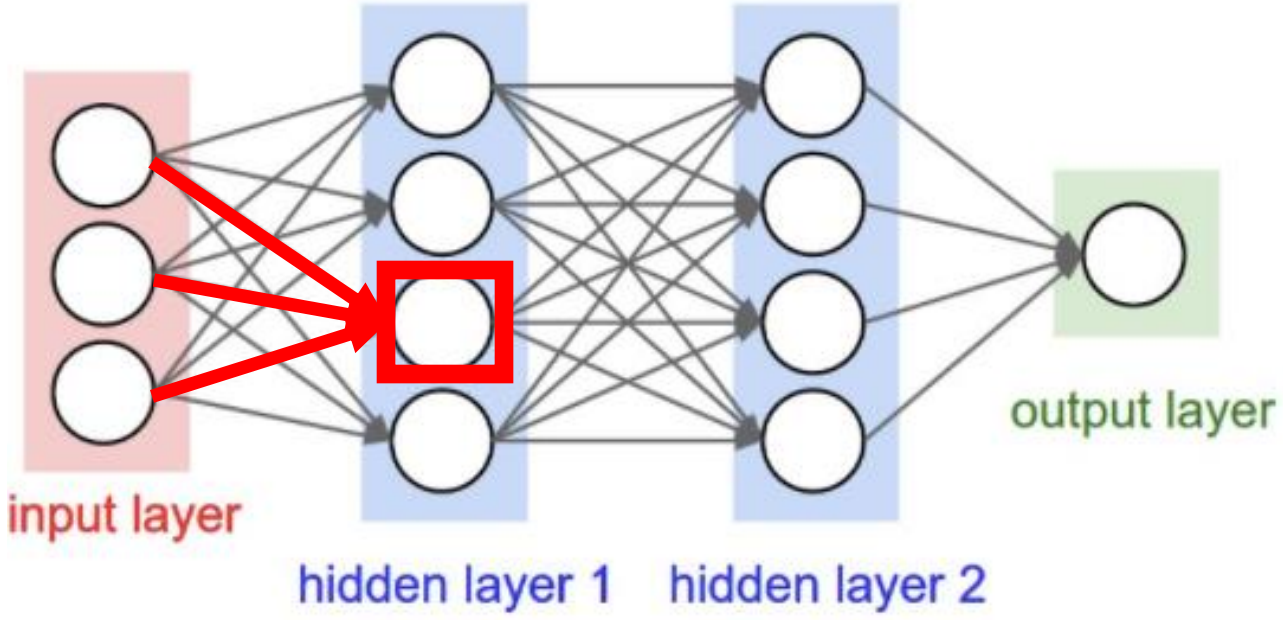


- Unit: computes a weighted sum and applies an activation function

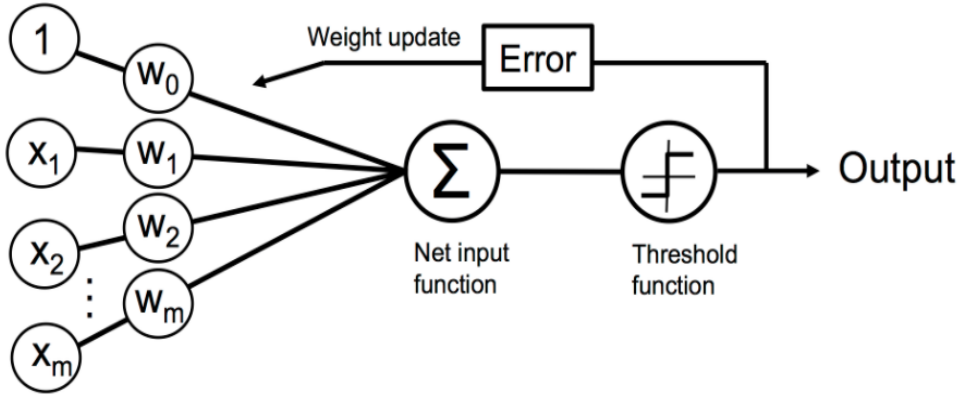
Python Machine Learning; Raschka & Mirjalili

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

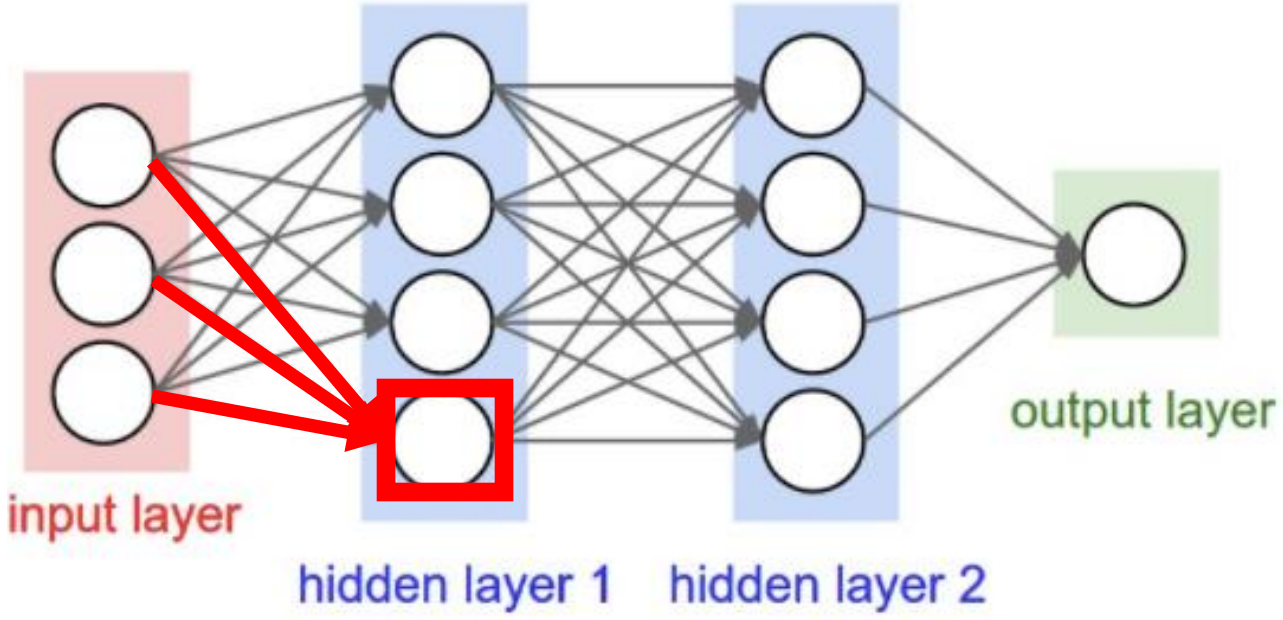


- Unit: computes a weighted sum and applies an activation function

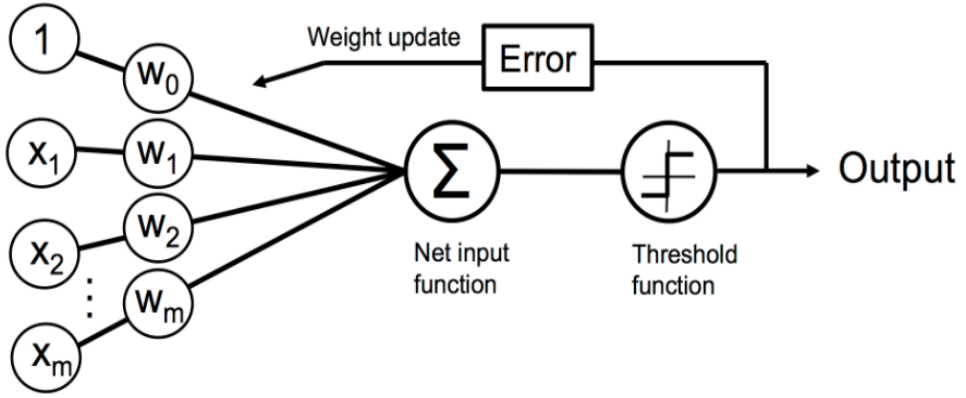
Python Machine Learning; Raschka & Mirjalili

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

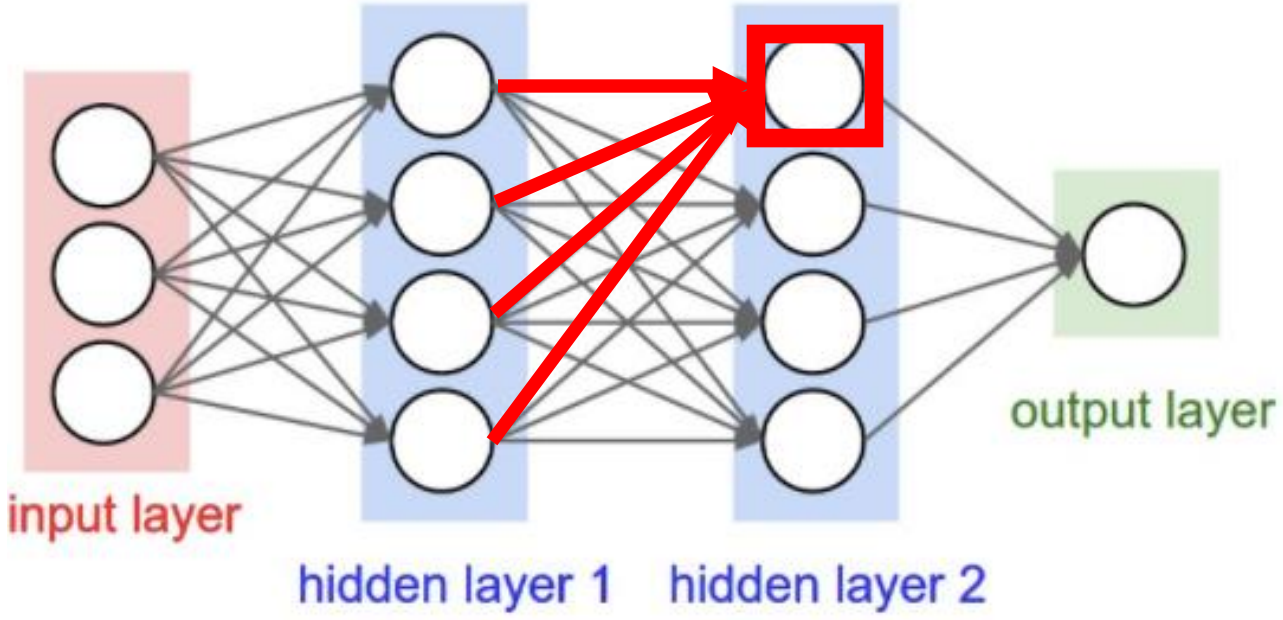


- Unit: computes a weighted sum and applies an activation function

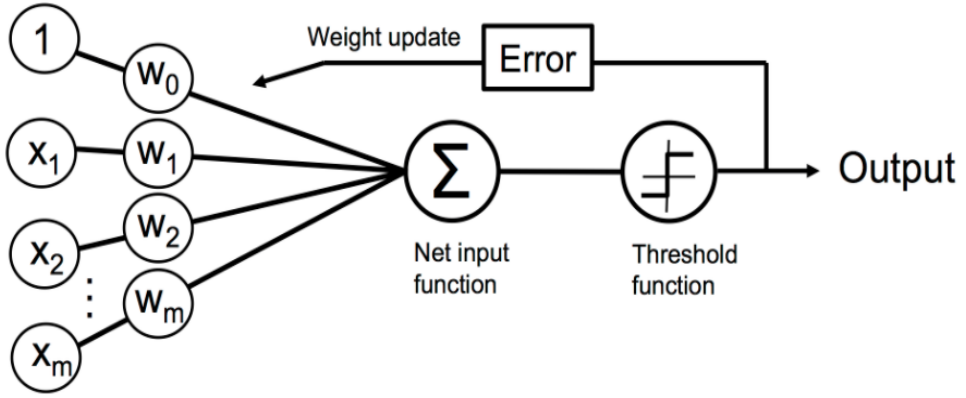
Python Machine Learning; Raschka & Mirjalili

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

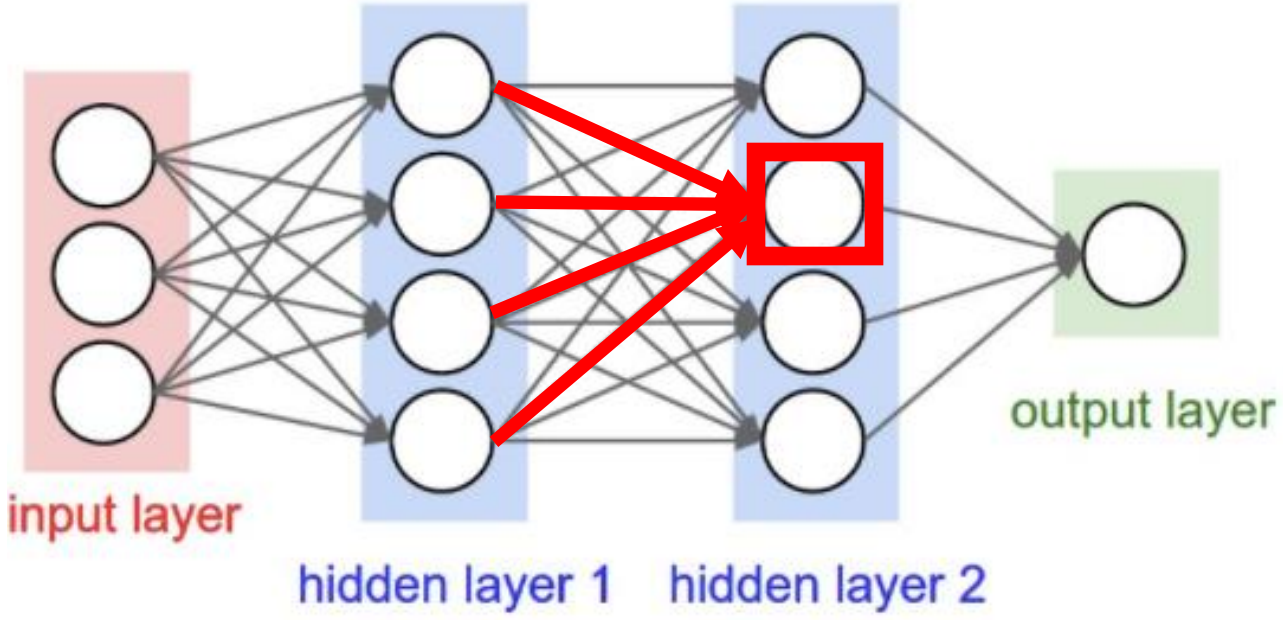


- Unit: computes a weighted sum and applies an activation function

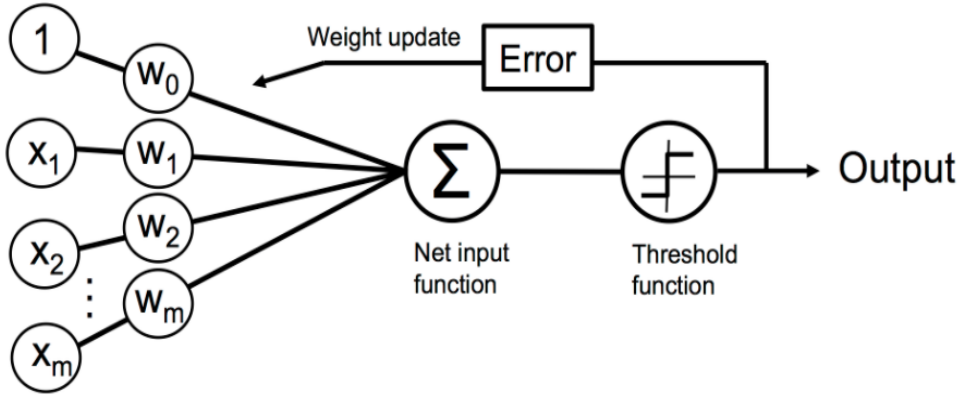
Python Machine Learning; Raschka & Mirjalili

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

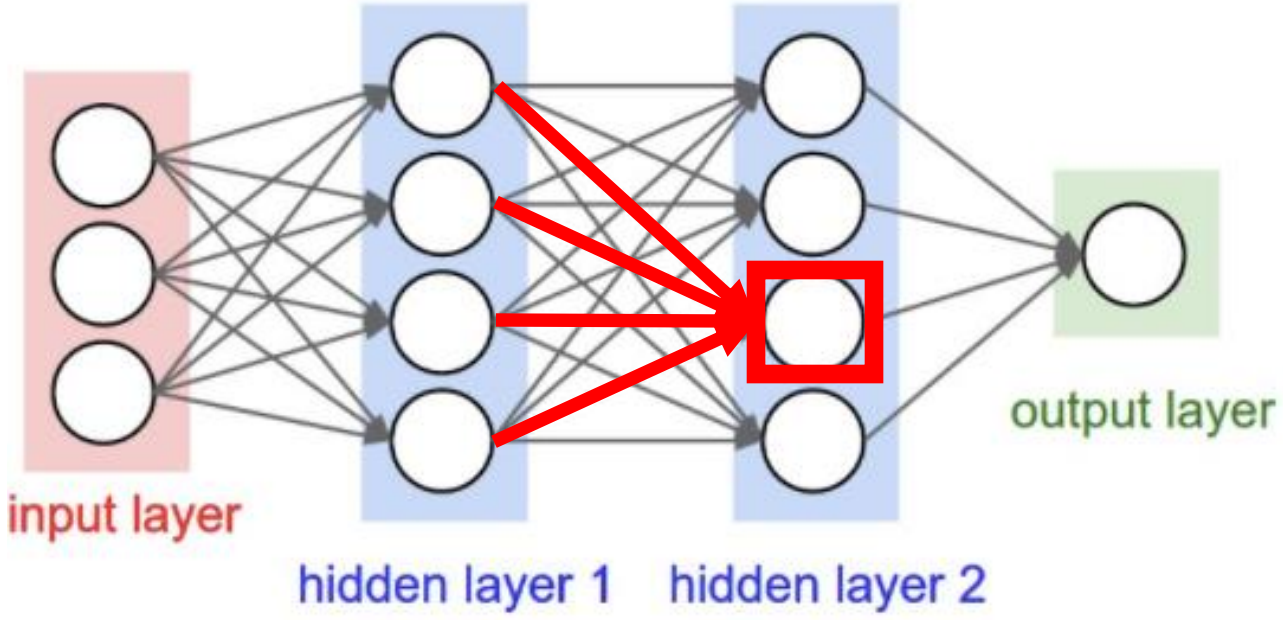


- Unit: computes a weighted sum and applies an activation function

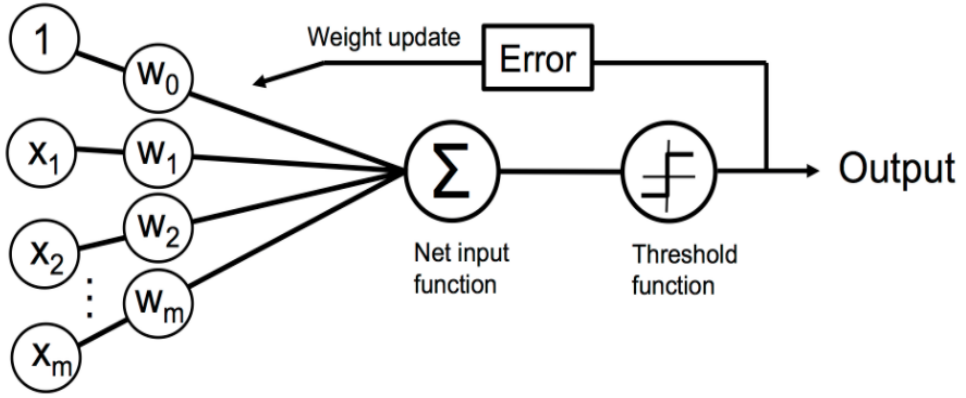
Python Machine Learning; Raschka & Mirjalili

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

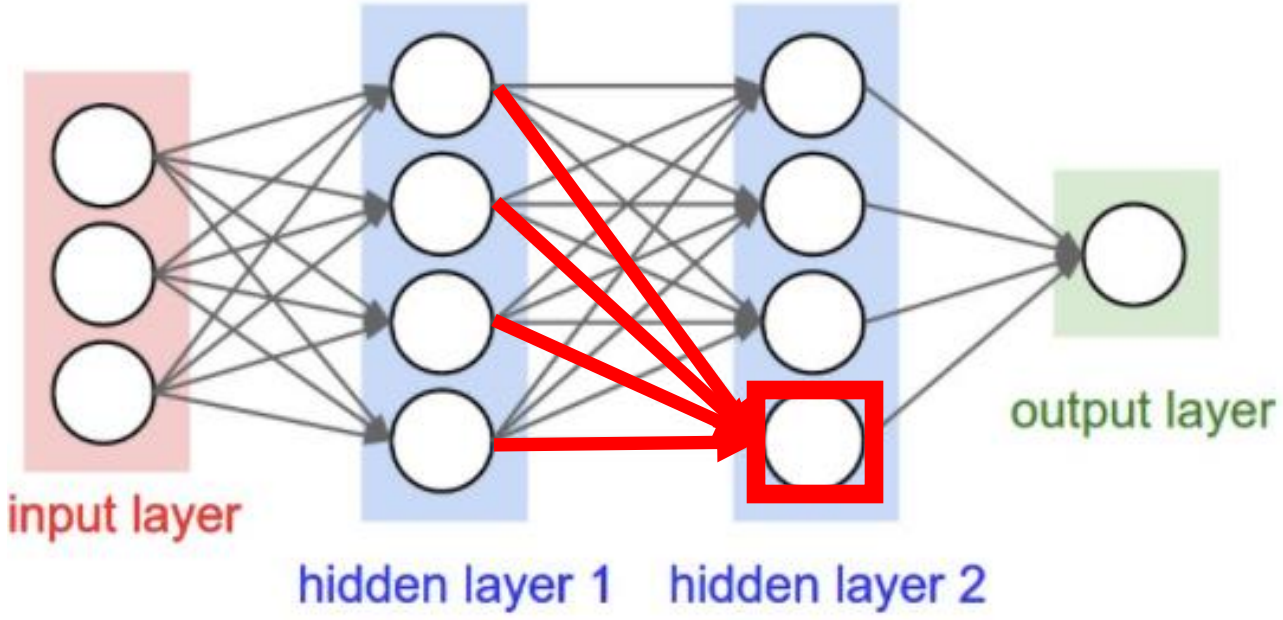


- Unit: computes a weighted sum and applies an activation function

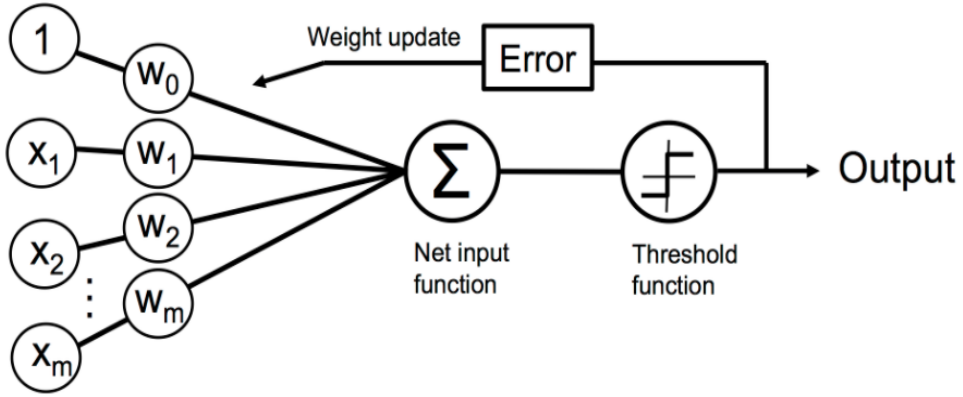
Python Machine Learning; Raschka & Mirjalili

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

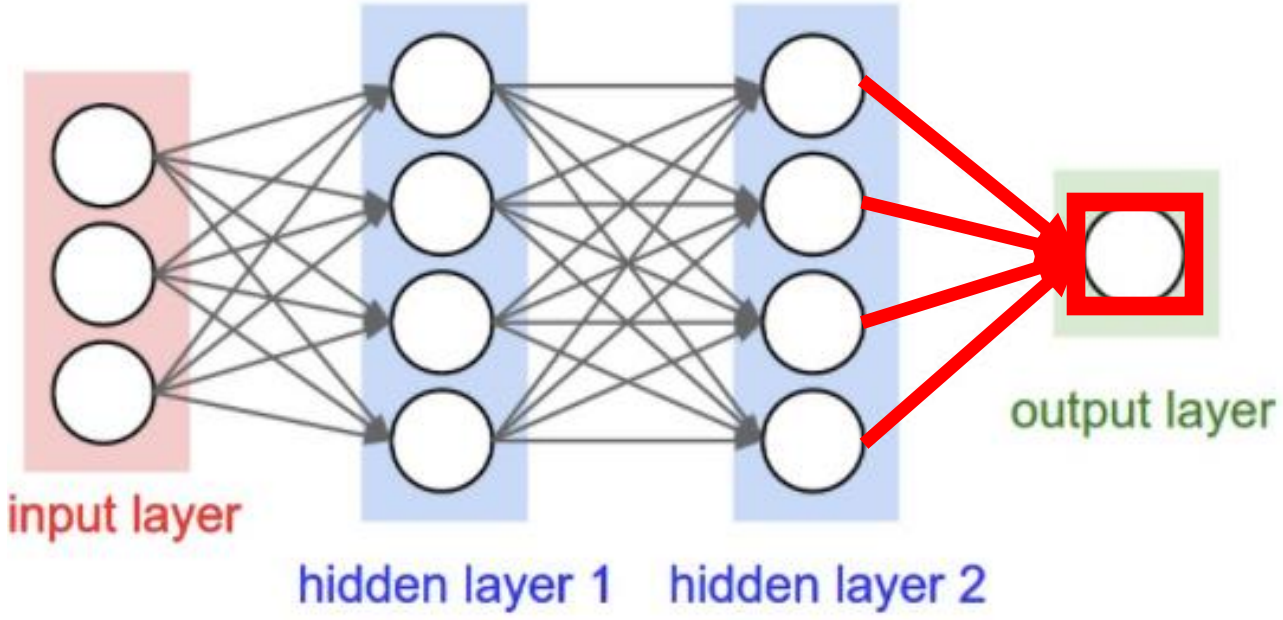


- Unit: computes a weighted sum and applies an activation function

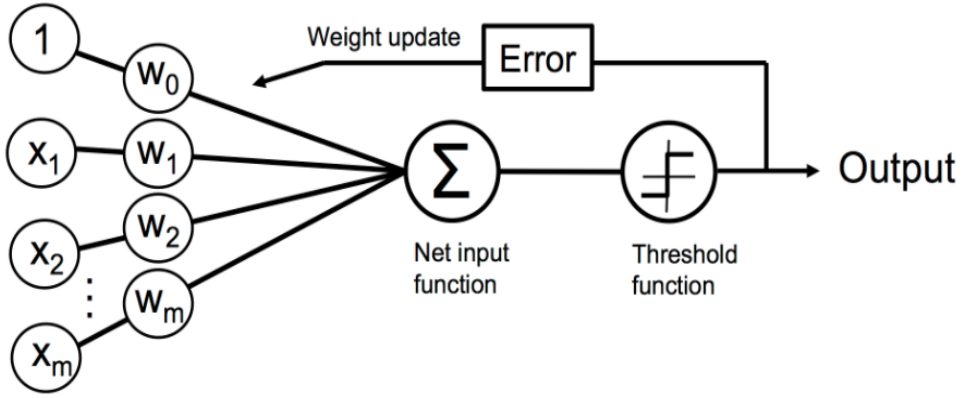
Python Machine Learning; Raschka & Mirjalili

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

Neural Network: Hidden Layers



- How does this relate to a perceptron?

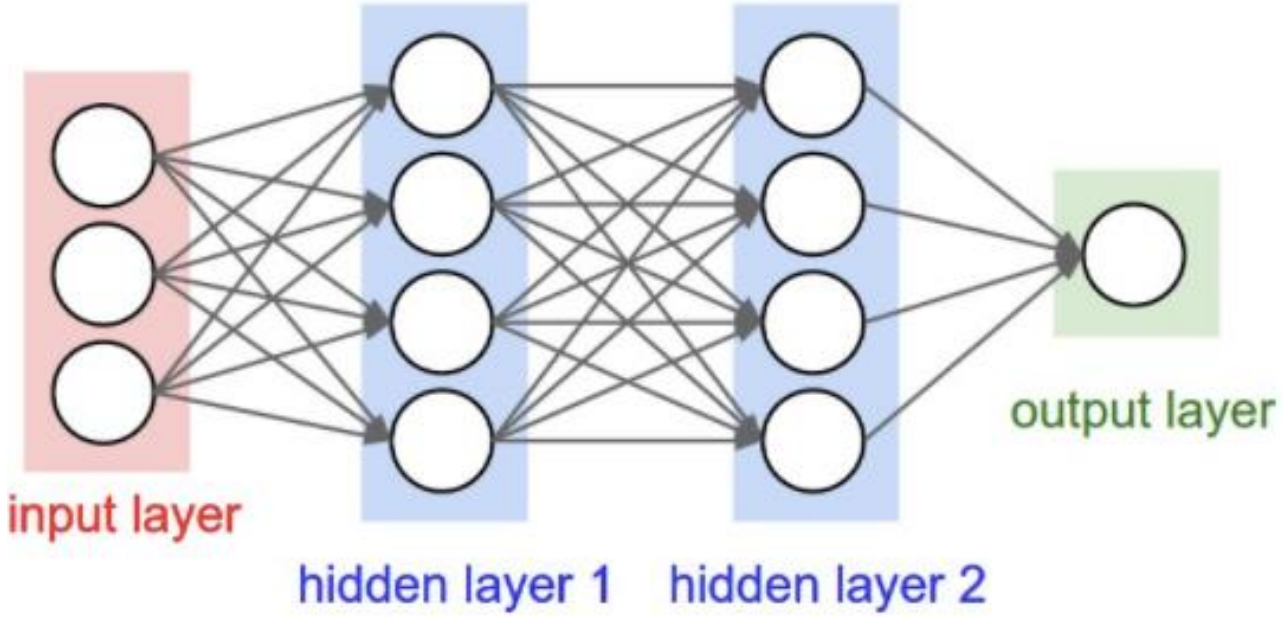


- Unit: computes a weighted sum and applies an activation function

Python Machine Learning; Raschka & Mirjalili

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

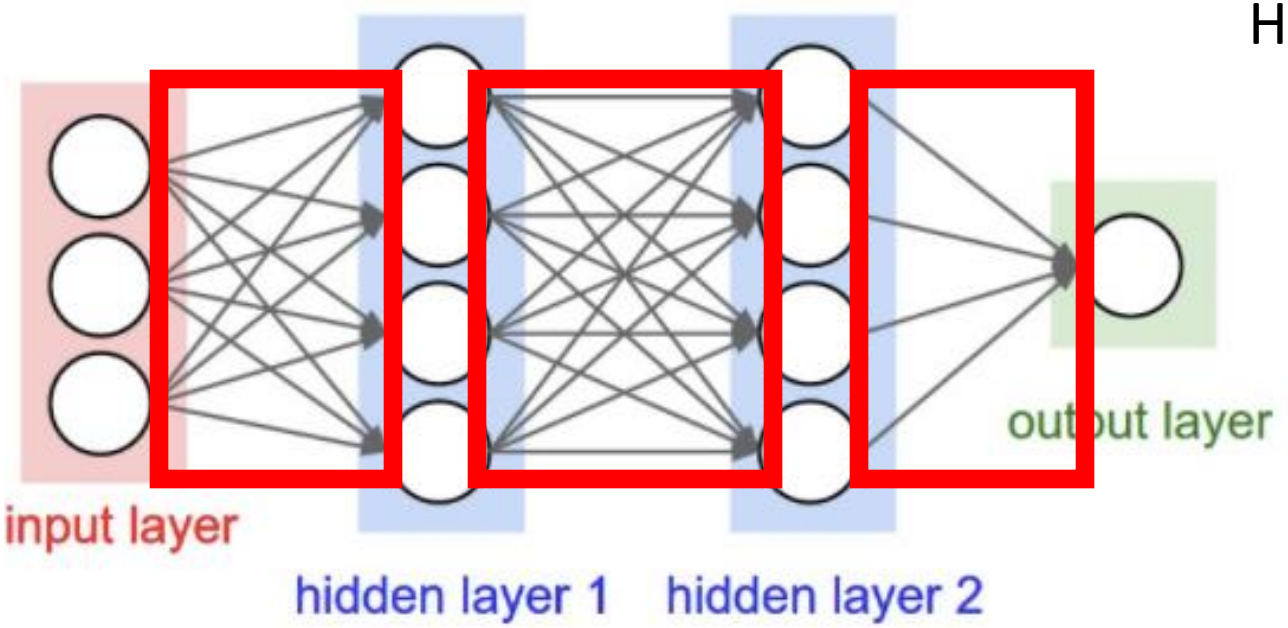
Neural Network: Hidden Layers



- **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: Hidden Layers

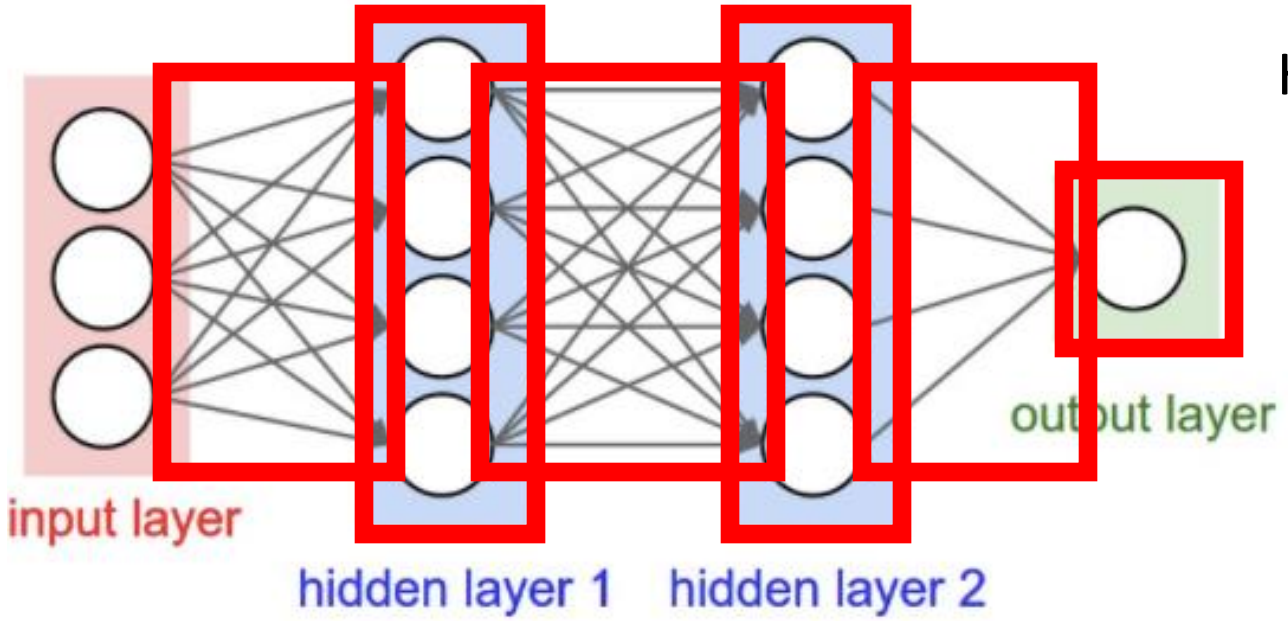


How many weights are in this model?

- 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: Hidden Layers

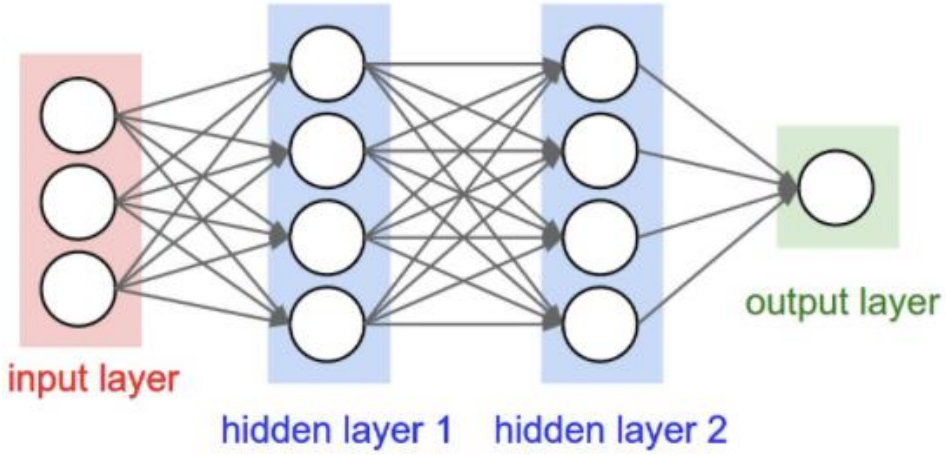


How many parameters are there to learn?

- 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



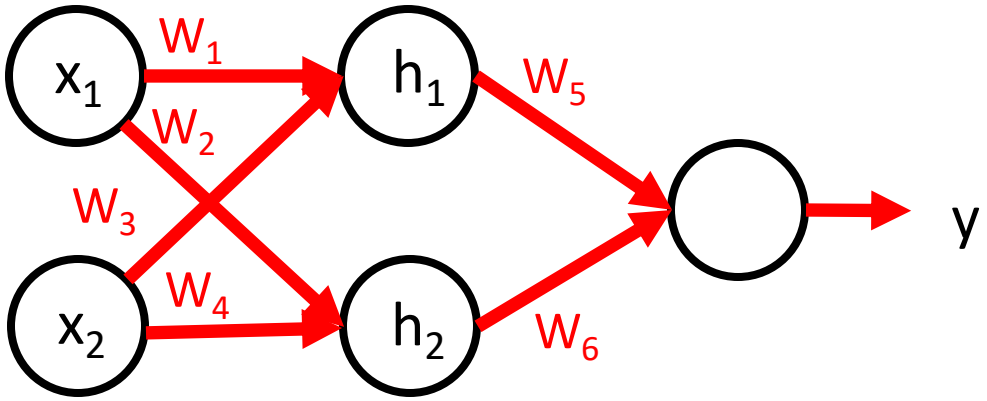
- What does it mean for a model to be fully connected?
  - Each unit provides input to each unit in the next layer
- What does it mean for a model to be feedforward?
  - Each layer serves as input to the next layer with no loops

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.,



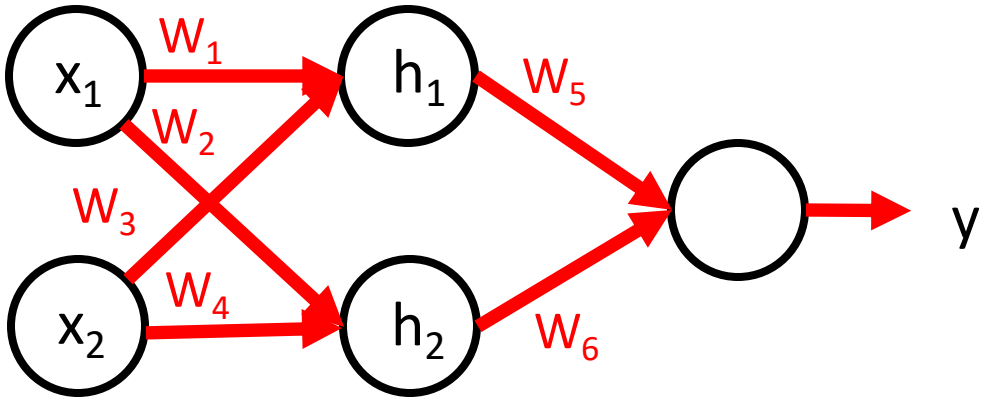
- What is function for h_1?
  - $h_1 = w_1x_1 + w_3x_2 + b_1$
- What is function for h_2?
  - $h_2 = w_2x_1 + w_4x_2 + b_2$
- What is function for y?
  - $y = h_1w_5 + h_2w_6 + b_3$
  - $y = (w_1x_1 + w_3x_2 + b_1)w_5 + (w_2x_1 + w_4x_2 + b_2)w_6 + b_3$
  - $y = w_1w_5x_1 + w_3w_5x_2 + w_5b_1 + w_2w_6x_1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

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.,



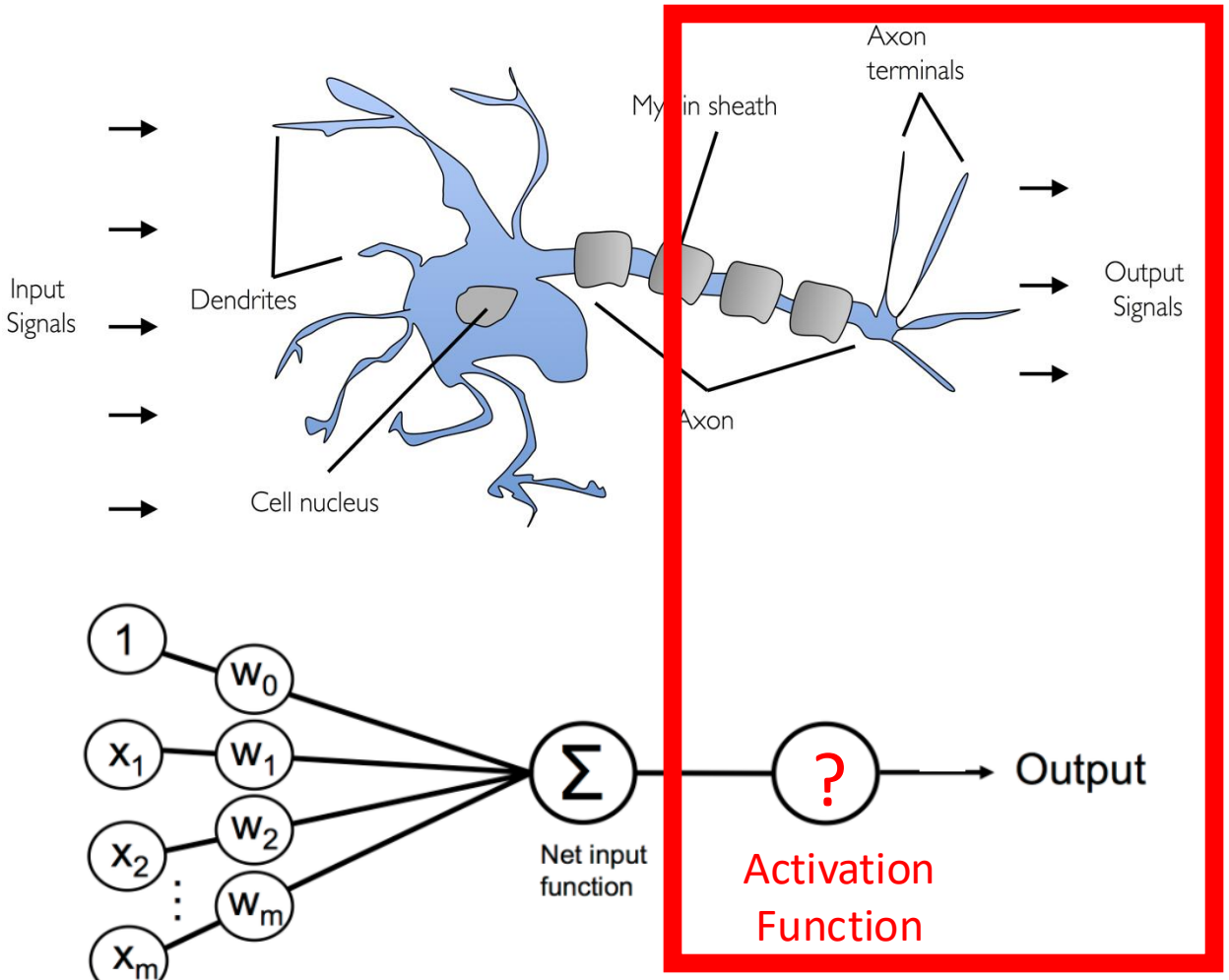
- What is function for h_1?
  - $h_1 = w_1x_1 + w_3x_2 + b_1$
- What is function for h_2?
  - $h_2 = w_2x_1 + w_4x_2 + b_2$
- What is function for y?
  - $y = h_1w_5 + h_2w_6 + b_3$

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 Neuron:



Artificial Neurons (e.g., Perceptron):

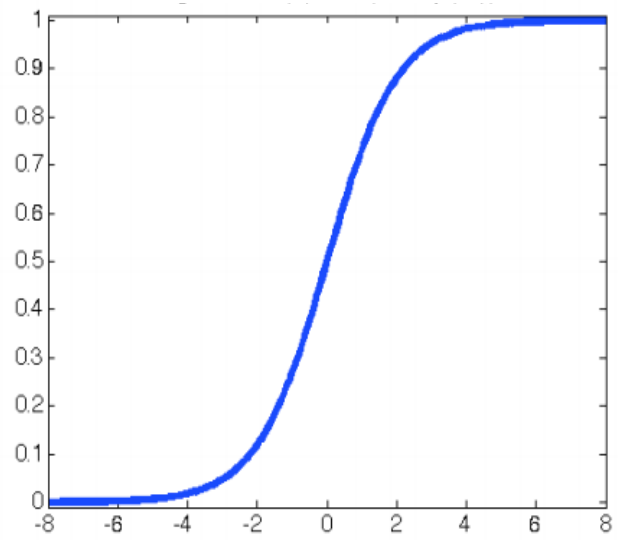
Mimic a neuron firing, by having each unit apply a non-linear "activation" function to the weighted input

Python Machine Learning; Raschka & Mirjalili

Non-Linear Activation Functions

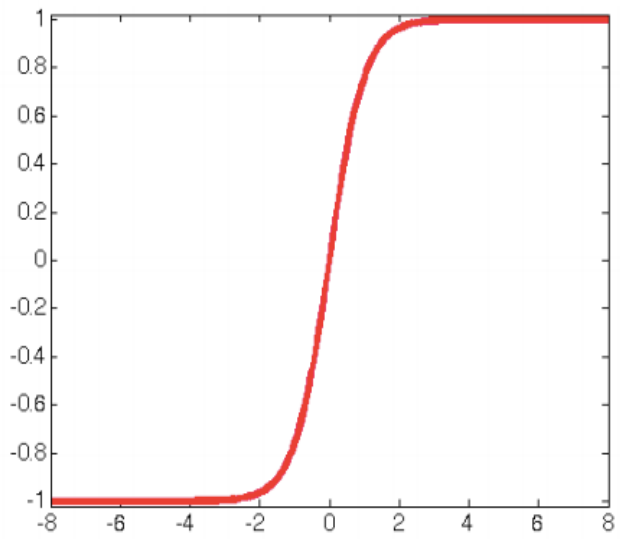
- Each unit applies a non-linear "activation" function to the weighted input to mimic a neuron firing

Sigmoid



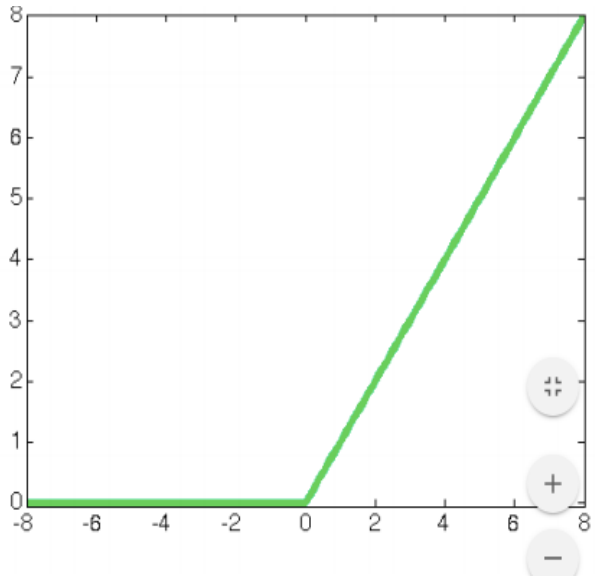
$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

Tanh



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

ReLU



$$\mathrm{ReLU}(z) = \max(0, z)$$

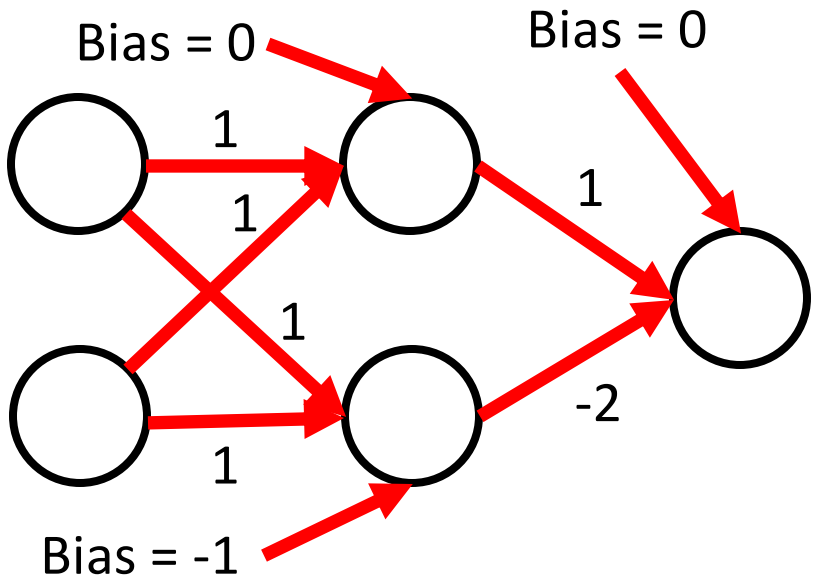
http://www.cs.utoronto.ca/~fidler/teaching/2015/slides/CSC411/10_nn1.pdf

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



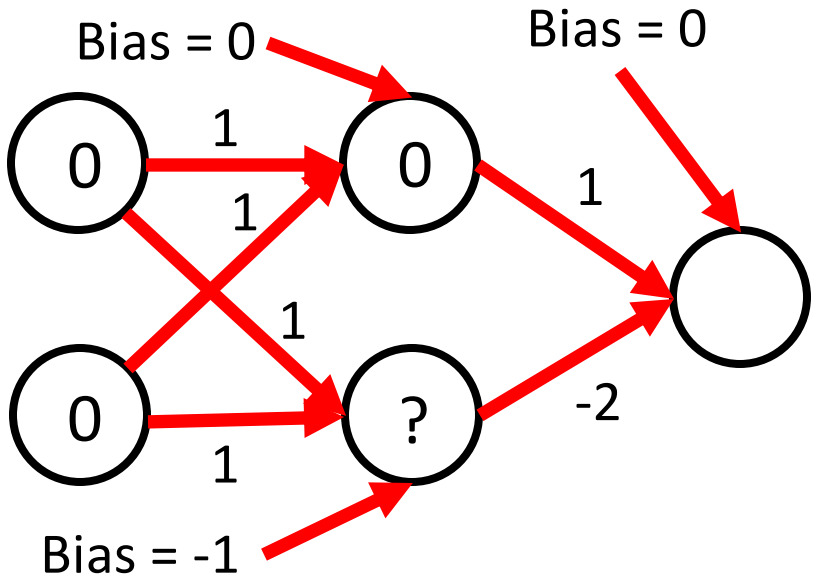
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



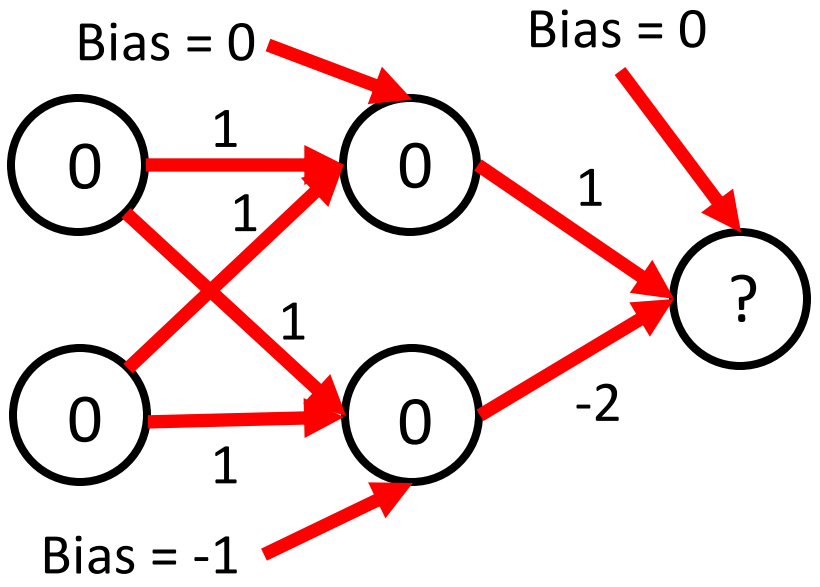
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



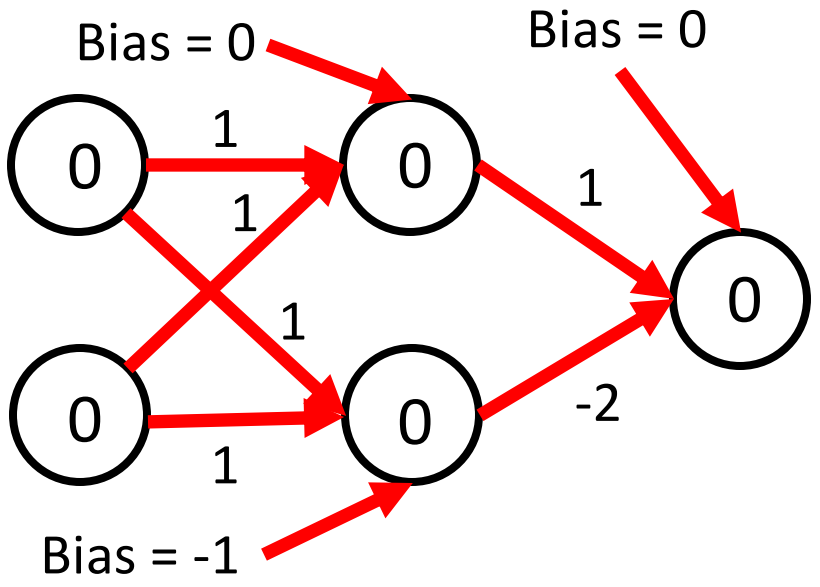
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



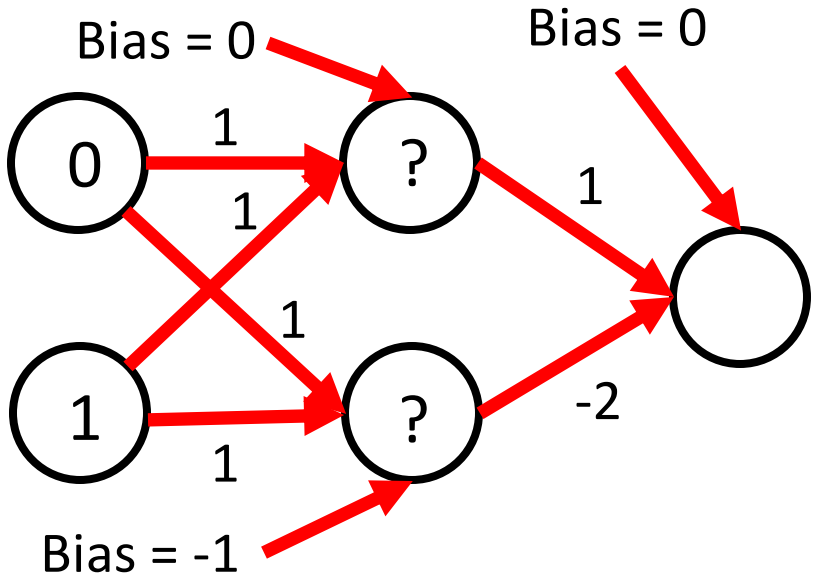
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



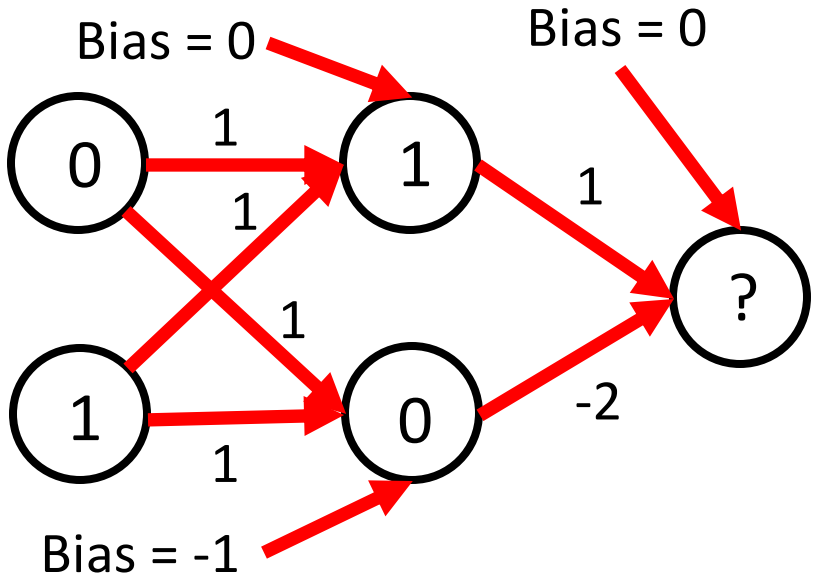
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



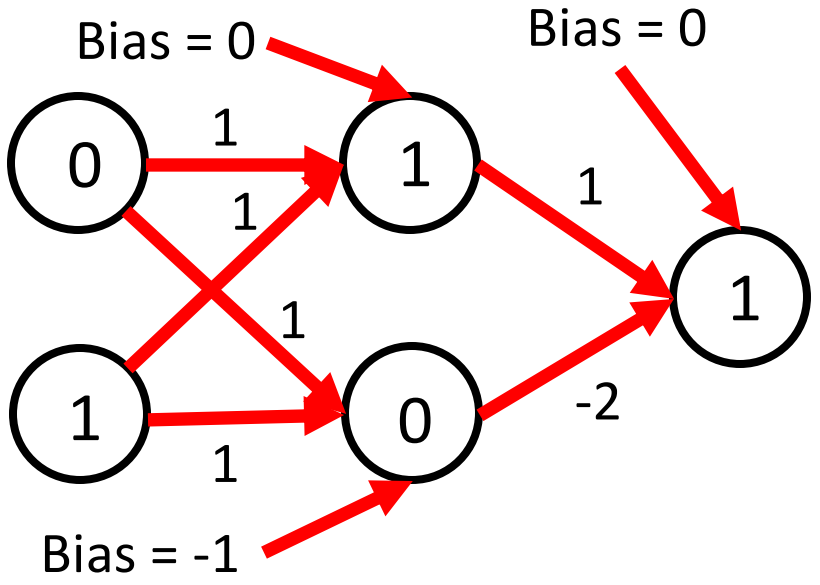
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



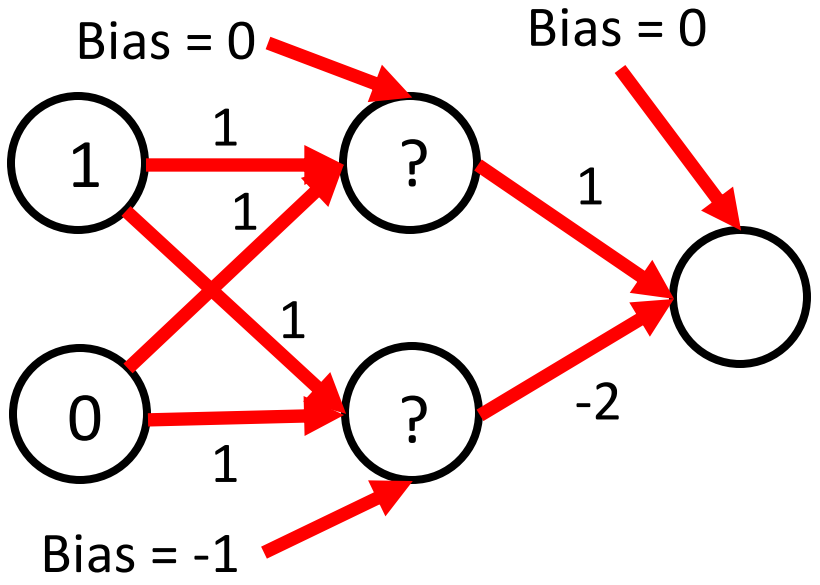
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



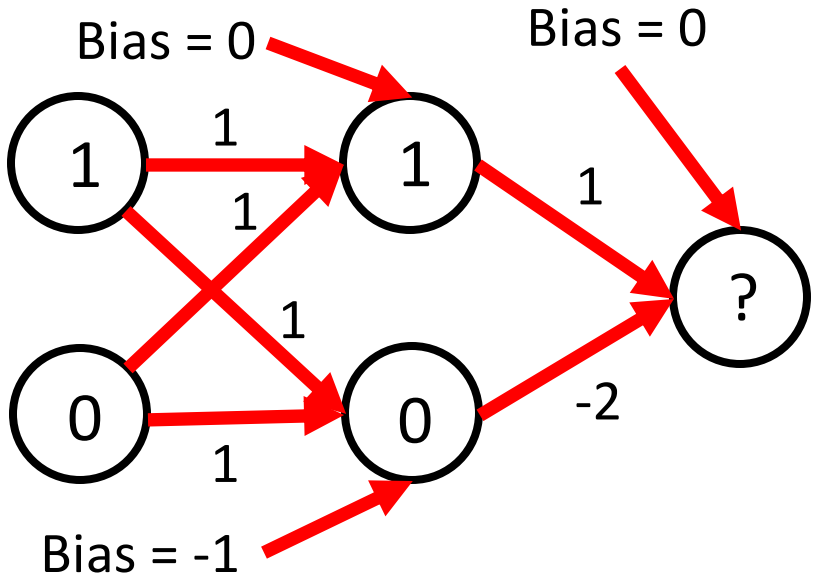
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



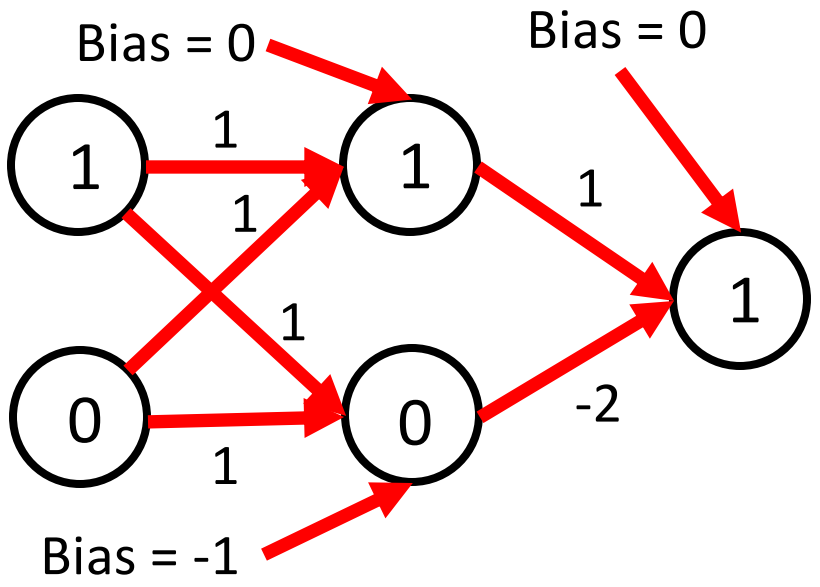
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



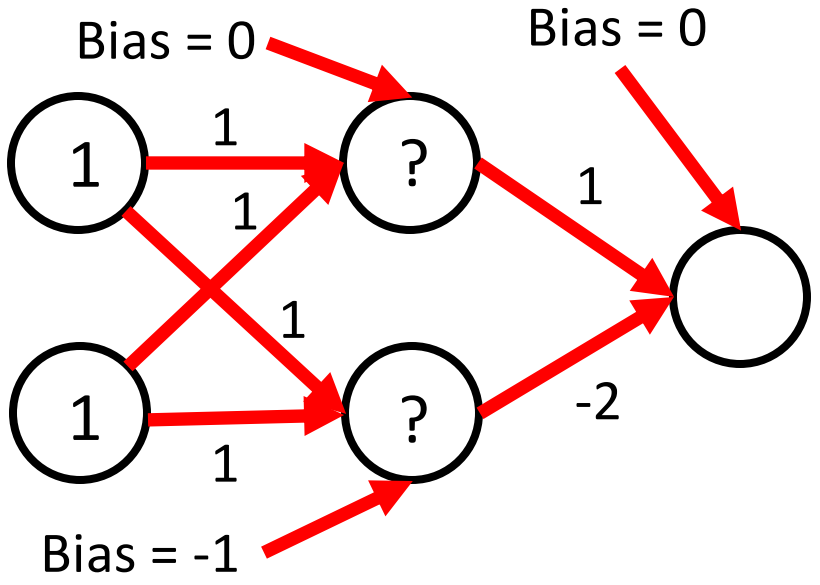
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



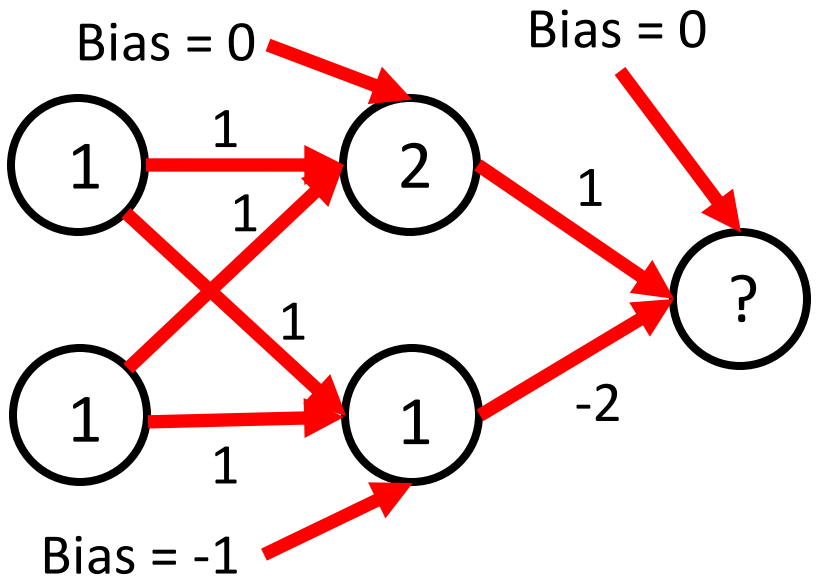
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network



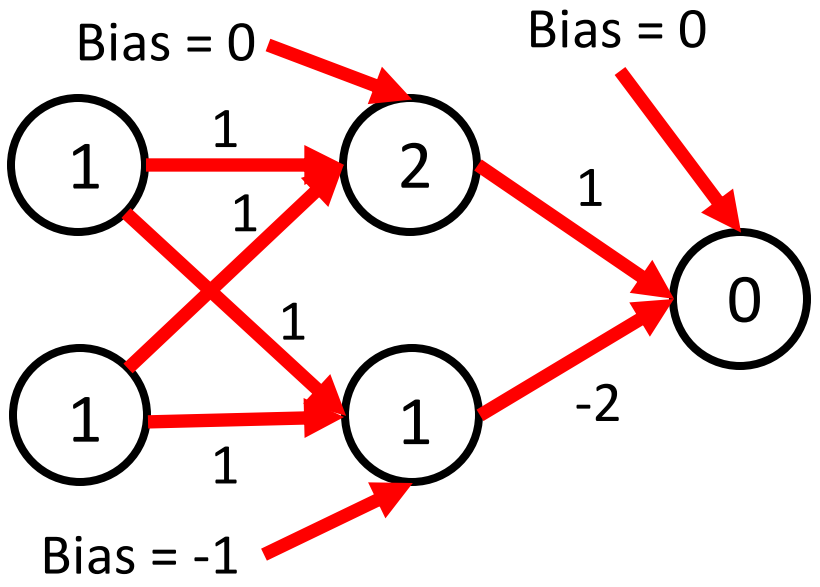
Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

- Separate 1s from 0s:

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Example neural network

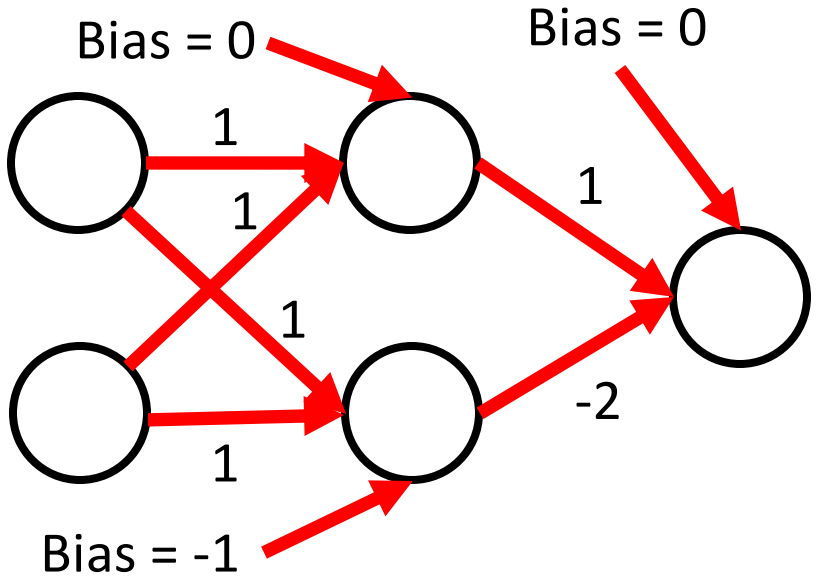


Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Non-Linear Example: Revisiting XOR problem

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

Example neural network



Activation function: $\mathrm{ReLU}(z) = \max(0, z)$

Key Questions When Creating Neural Networks

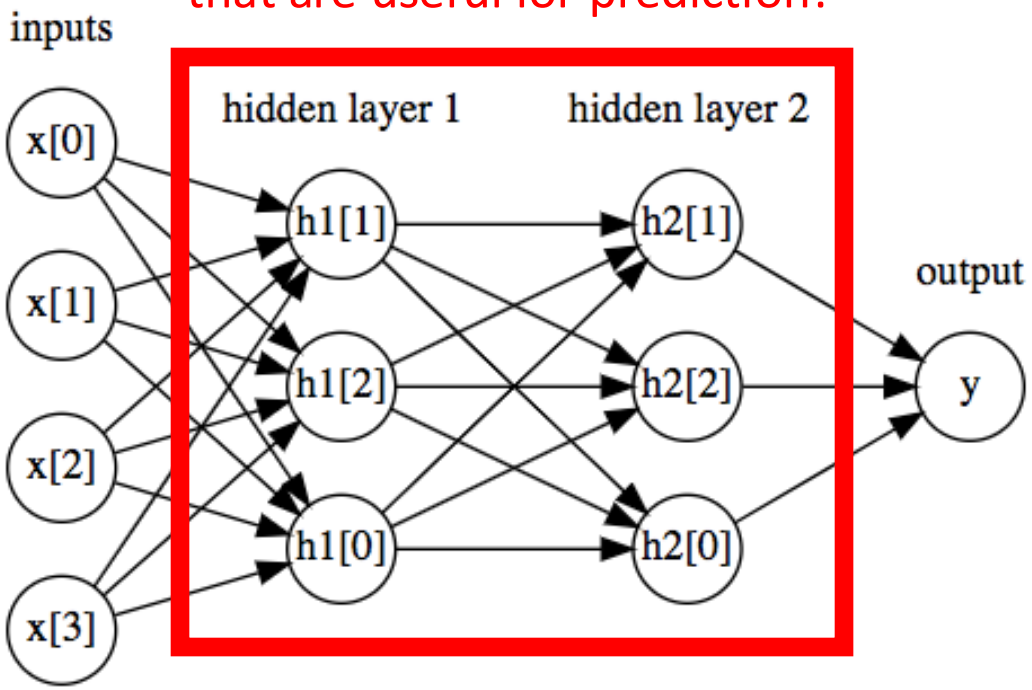
- How many layers should be used?
  - Note: field recoined as “Deep Learning” due to latest trend of adding layers
- How many nodes should be in each layer?
- Which activation function should be used?

Today's Topics

- Motivation for neural networks: need non-linear models
- Neural networks' basic ingredients: hidden layers and activation units
- Neural networks' support for diverse problems: output units
- Objective function: what a model should learn
- Programming tutorial

Recall: Neural Networks

Transforms input into features that are useful for prediction!

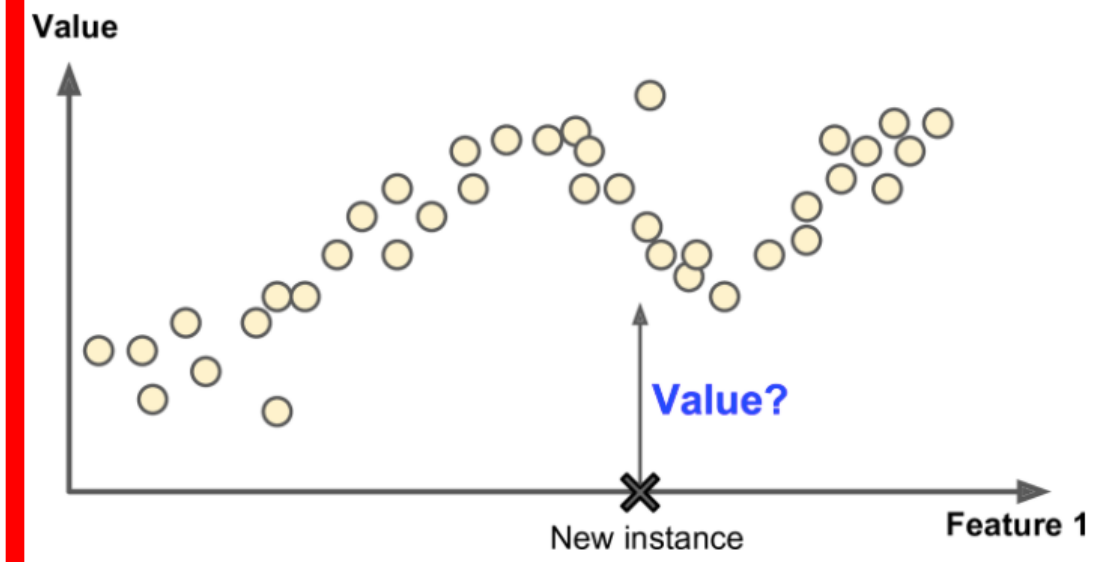


What should the model predict?

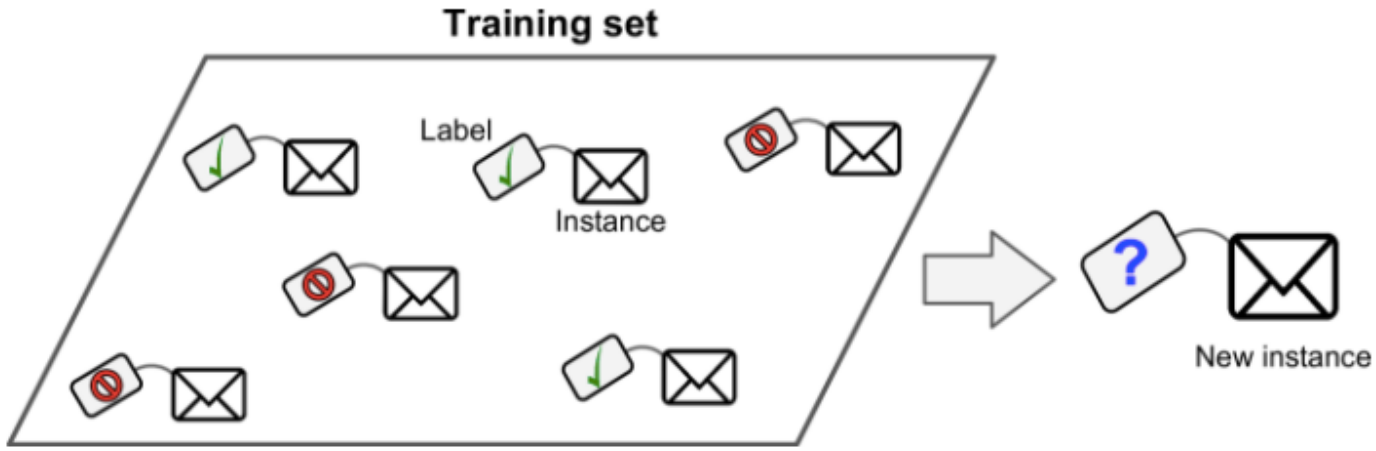
https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb

Desired Output Driven by Task

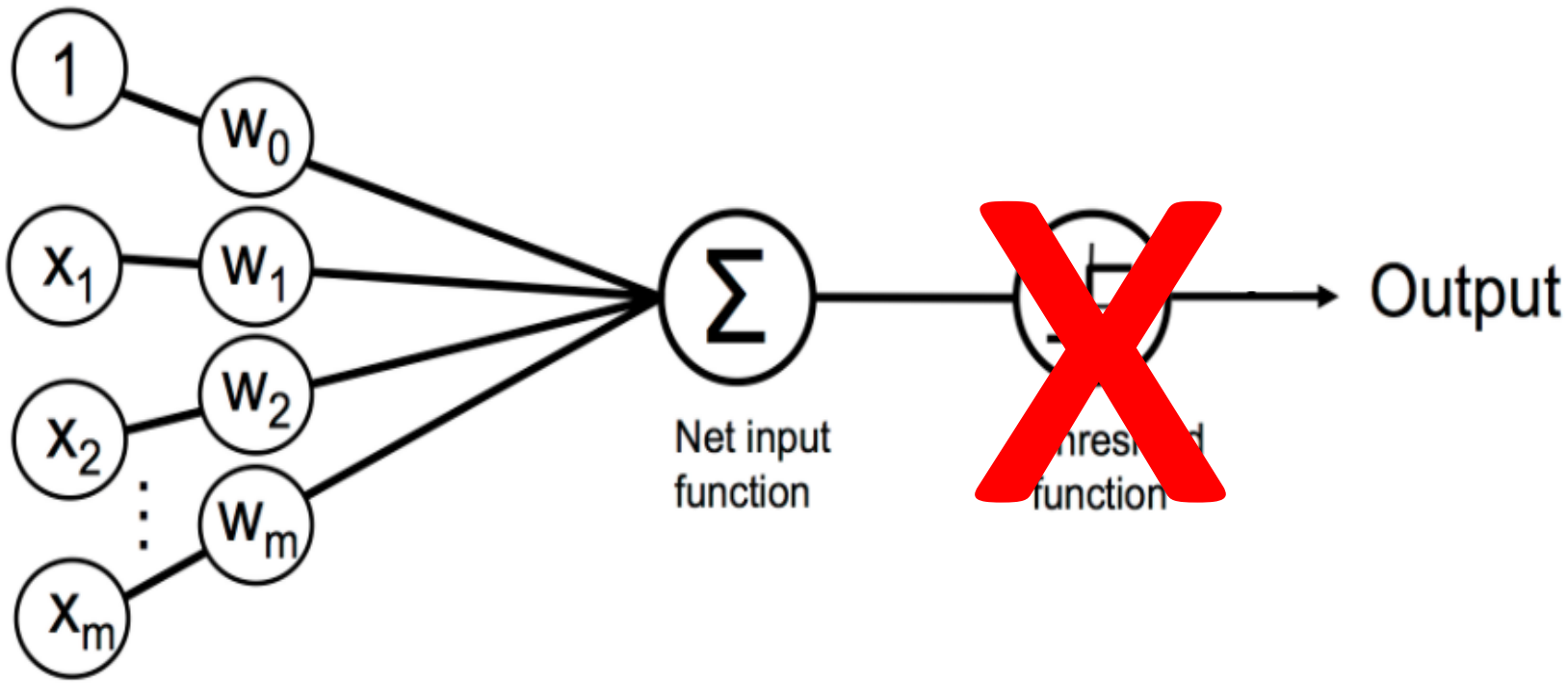
Regression
(predict **continuous** value)



Classification
(predict **discrete** value)



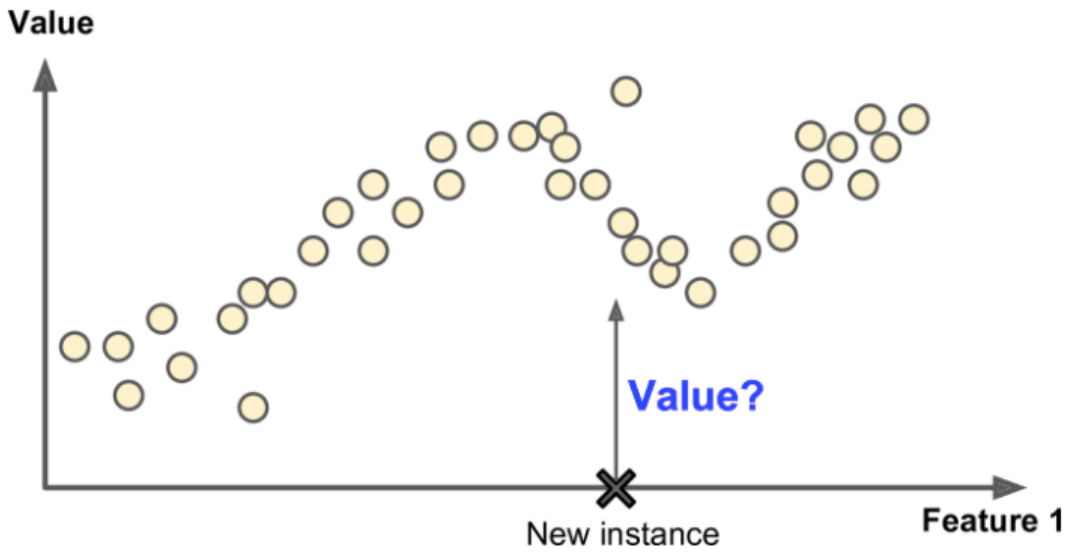
Linear (No Activation Function)



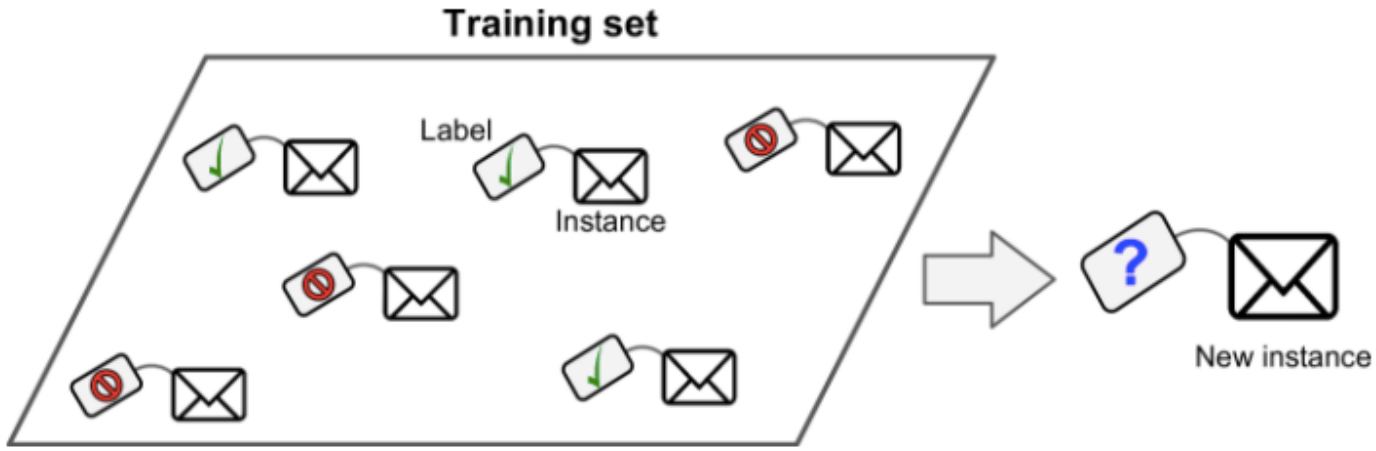
Python Machine Learning; Raschka & Mirjalili

Desired Output Driven by Task

Regression
(predict **continuous** value)

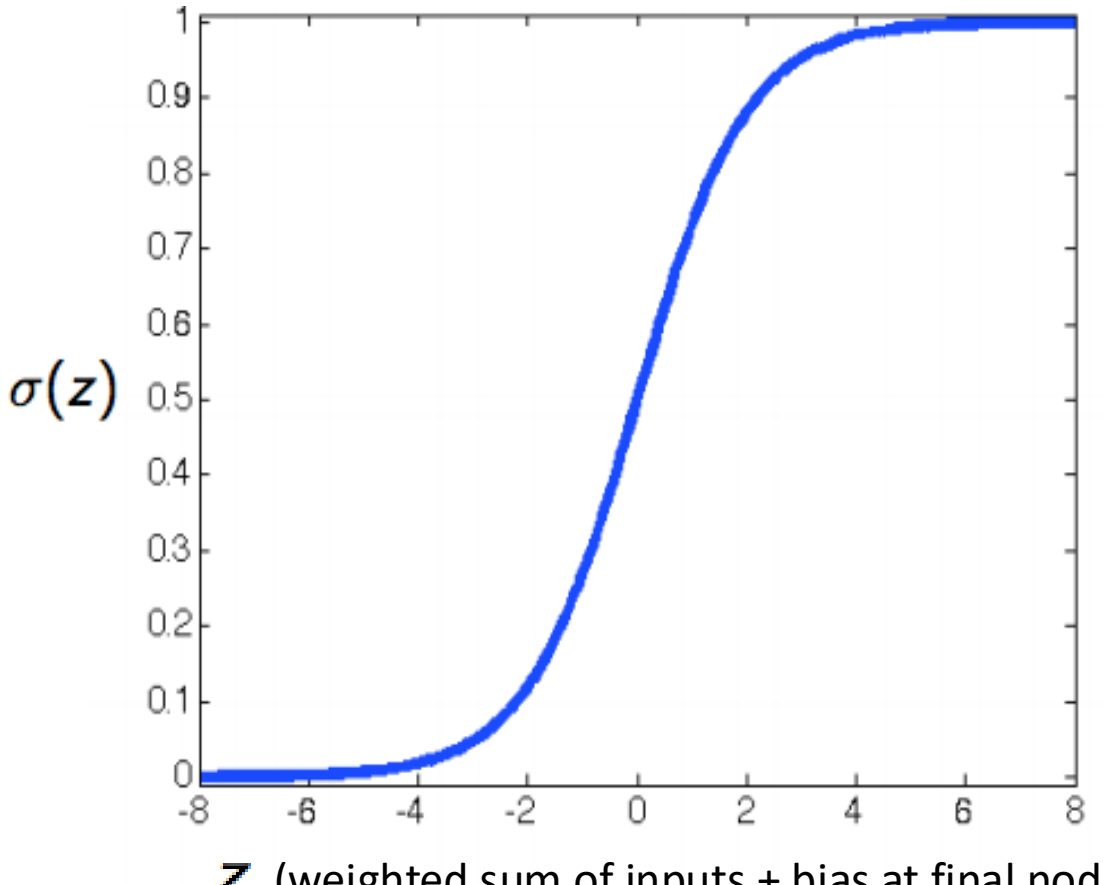


Classification
(predict **discrete** value)



Binary Classification: Sigmoid (aka, Logistic Regression)

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



If $\sigma(z)$ >= 0.5, output 1; Else, output 0

Why not use *z* instead of $\sigma(z)$?

- We want a probability in [0, 1]

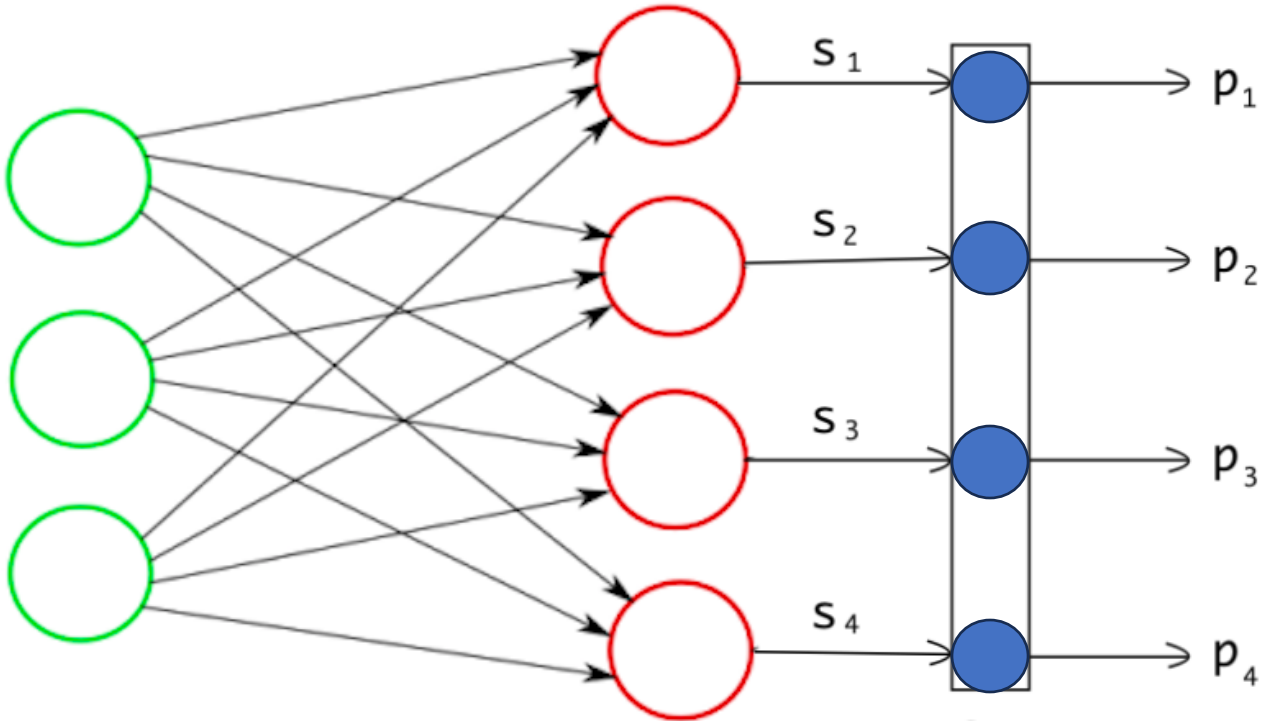
What happens to the output as z becomes more positive?

- e^-z approaches 0 so value approaches 1

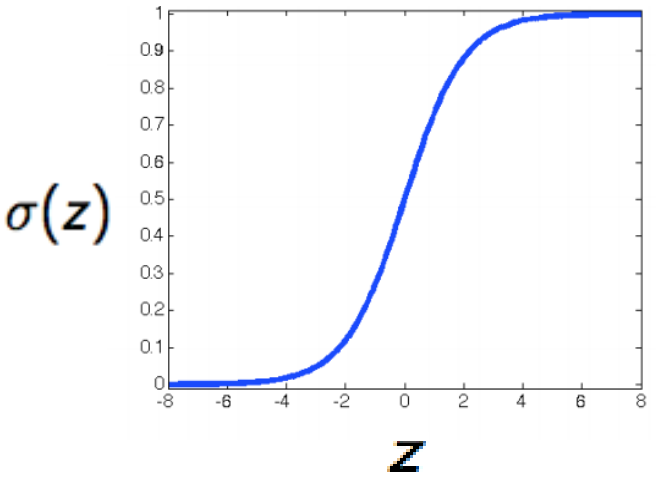
What happens to the output as z becomes more negative?

- e^-z becomes larger so value approaches 0

Multilabel Classification: Sigmoid Per Class



denotes a sigmoid function

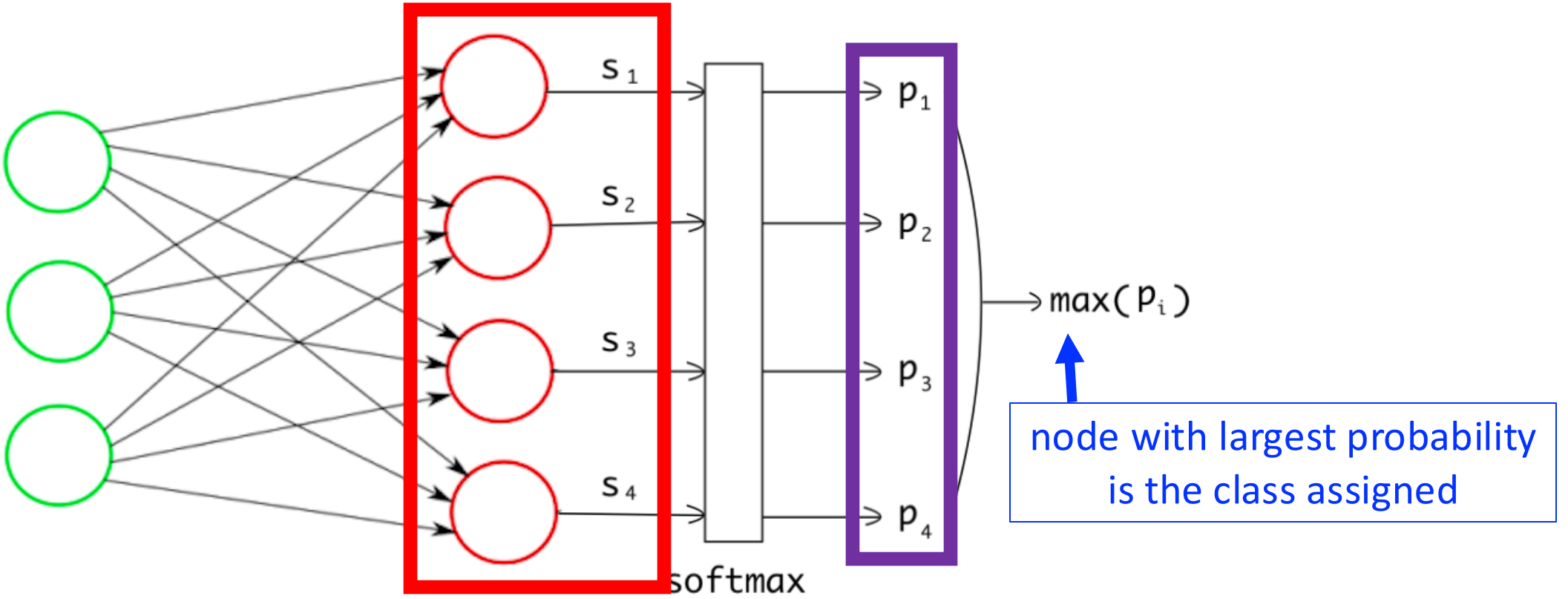


Outputs 1 when greater than or equal to threshold (e.g., 0.5) and 0 otherwise

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

Multiclass Classification: Softmax

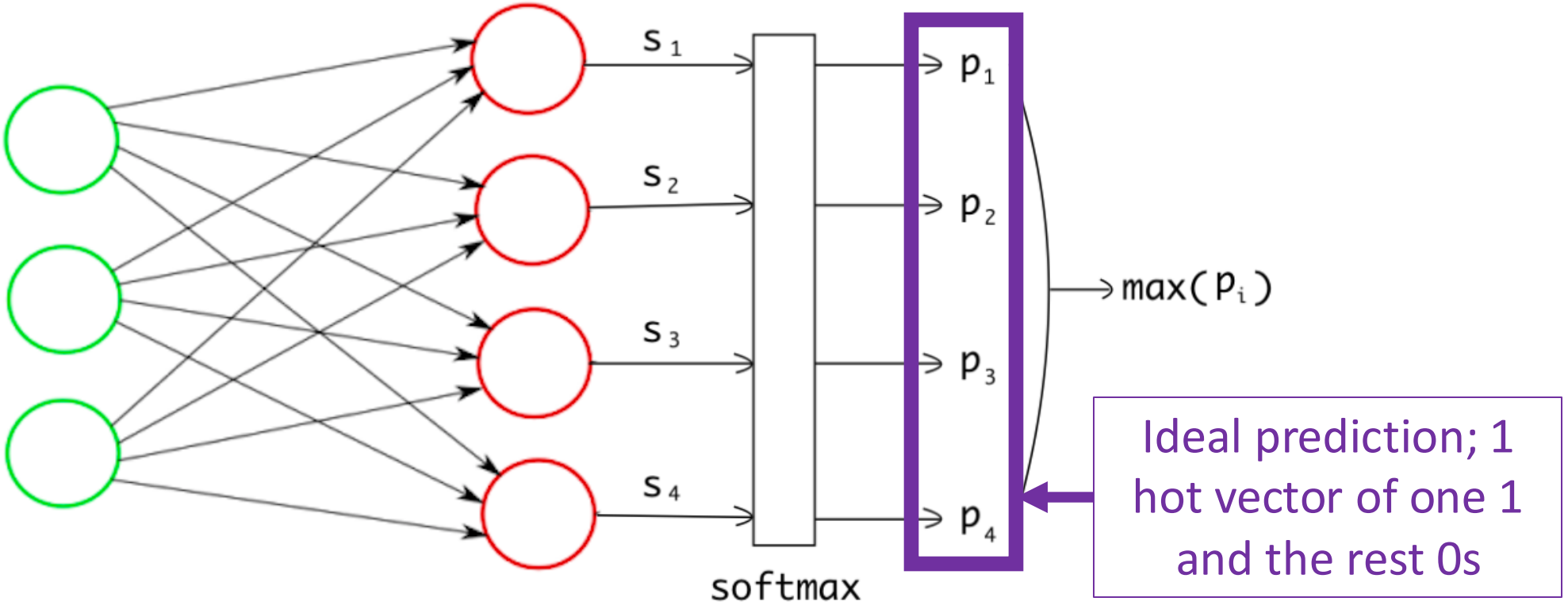
Converts vector of scores into a probability distribution that sums to 1; e.g.,



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

Multiclass Classification: Softmax

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



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

Multiclass Classification: Softmax

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

Exponentiating ensures positive values by making negative scores slightly larger than 0 and positive values grow exponentially; *e* is a convenient exponent base for computation during training

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

i = 1, …, K

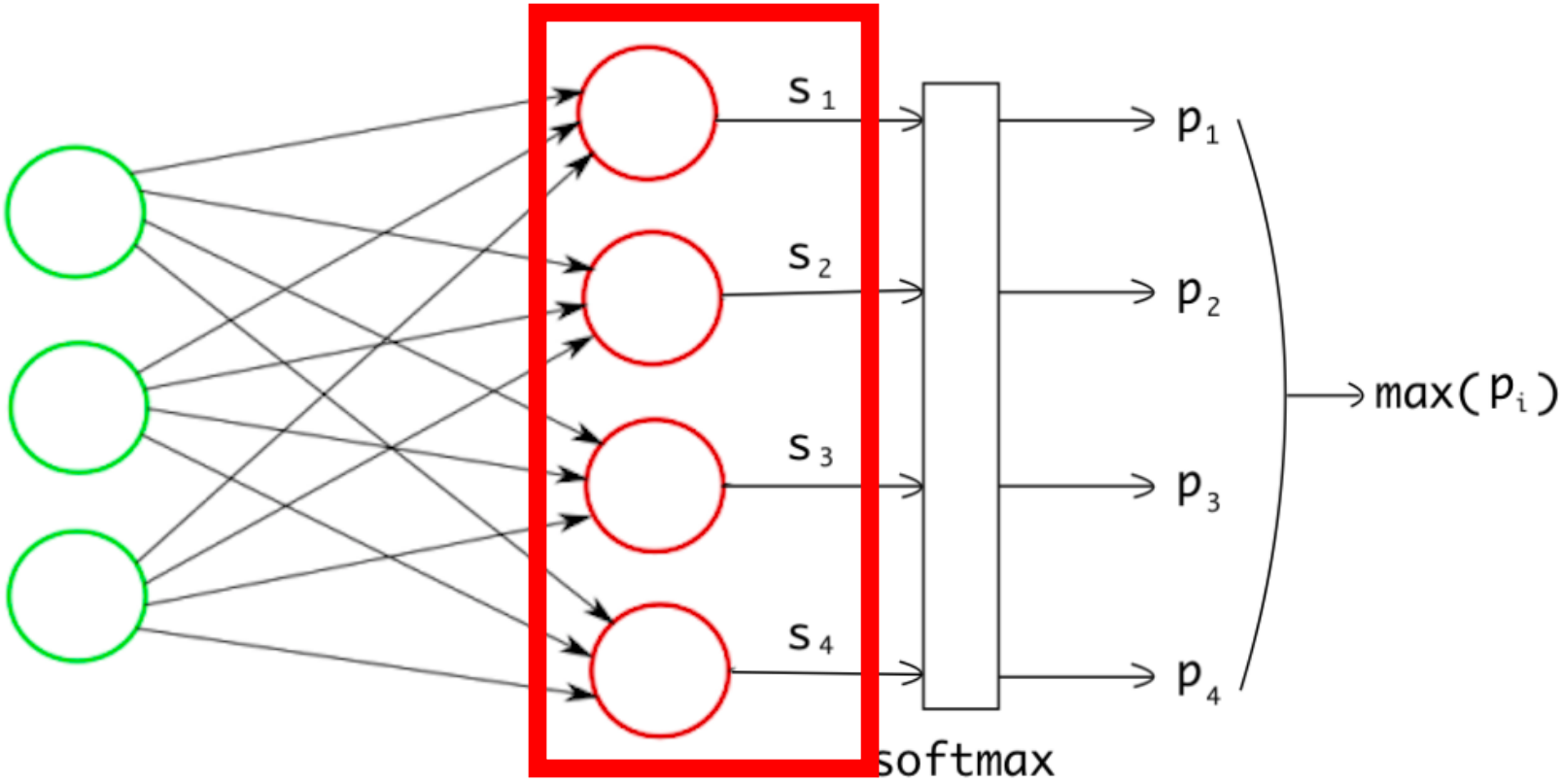
Number of classes

Produces probability by dividing each class score with sum across all classes (normalization)

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

Multiclass Classification: Softmax

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



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

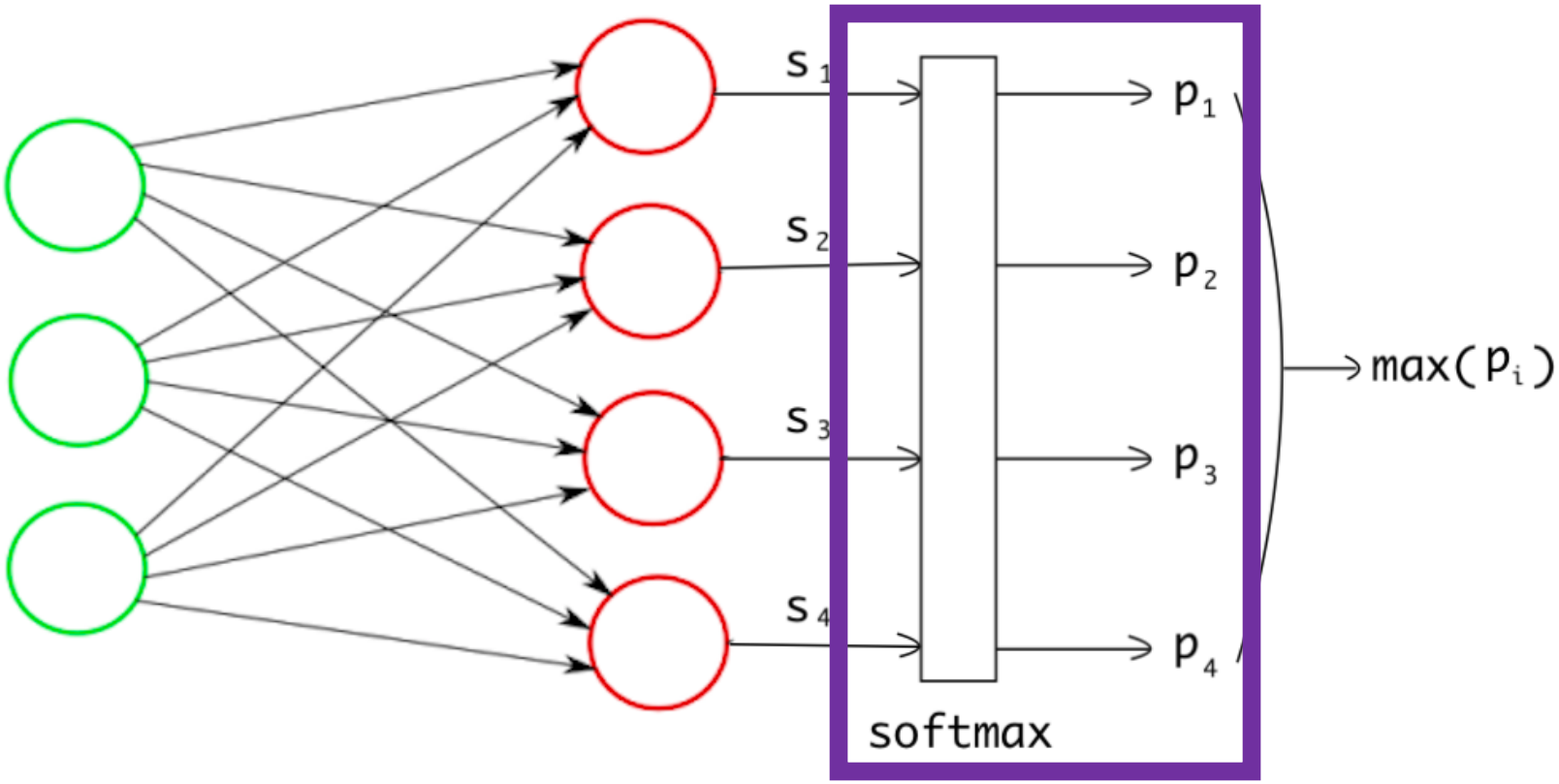
Multiclass Classification: Softmax

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

Multiclass Classification: Softmax

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



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

Multiclass Classification: Softmax

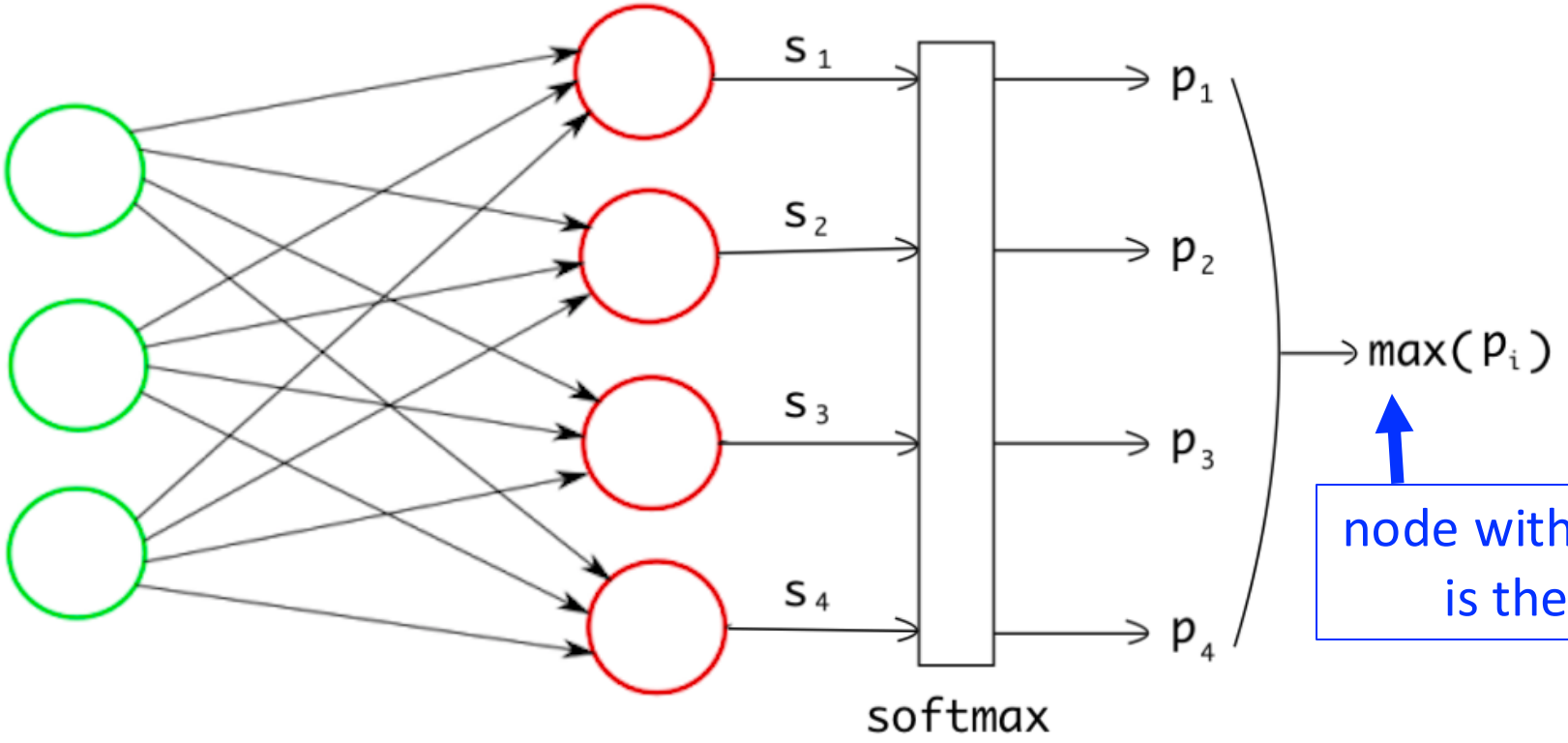
Converts vector of scores into a probability distribution that sums to 1; e.g.,

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

	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

Multiclass Classification: Softmax

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



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

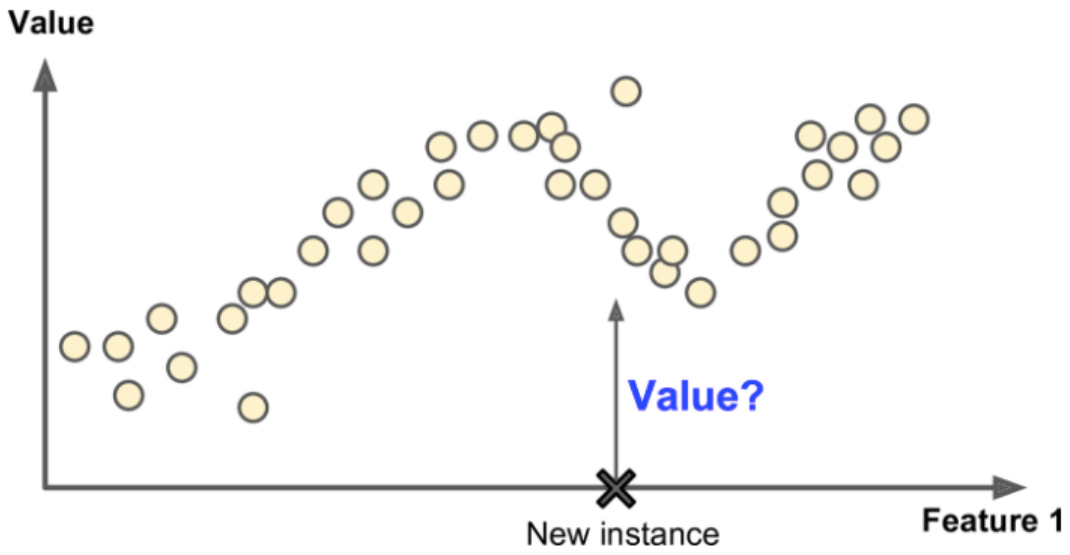
Multiclass Classification: Softmax

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

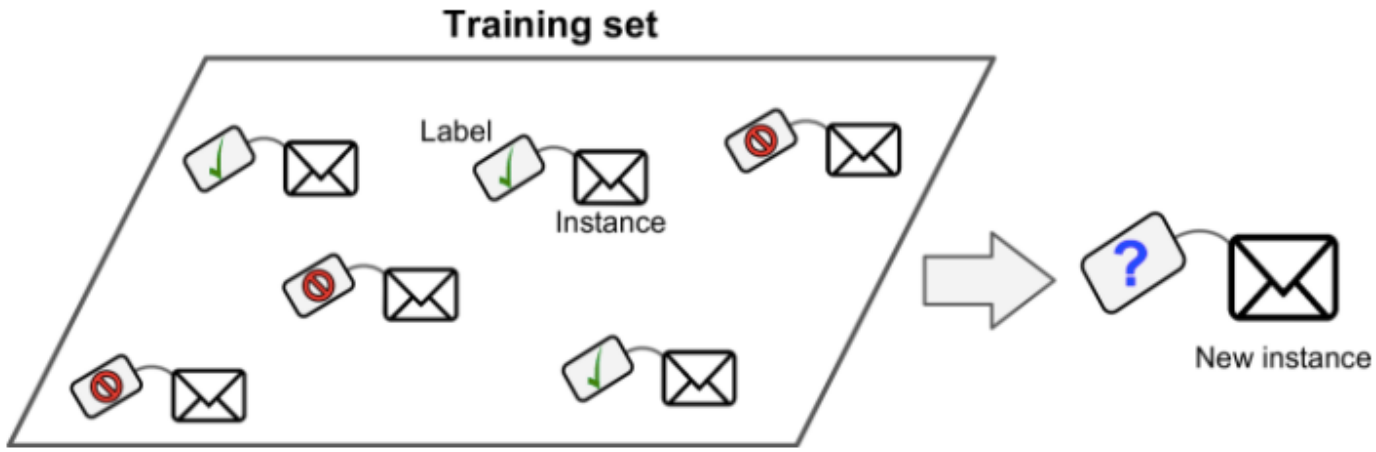
	Scoring Function	Unnormalized Probabilities	Normalized Probabilities
Dog	-3.44	0.0321	0.0006
Cat	1.16	3.1899	0.0596
Boat	-0.81	0.4449	0.0083
Airplane	3.91	49.8990	0.9315

Desired Output Driven by Task

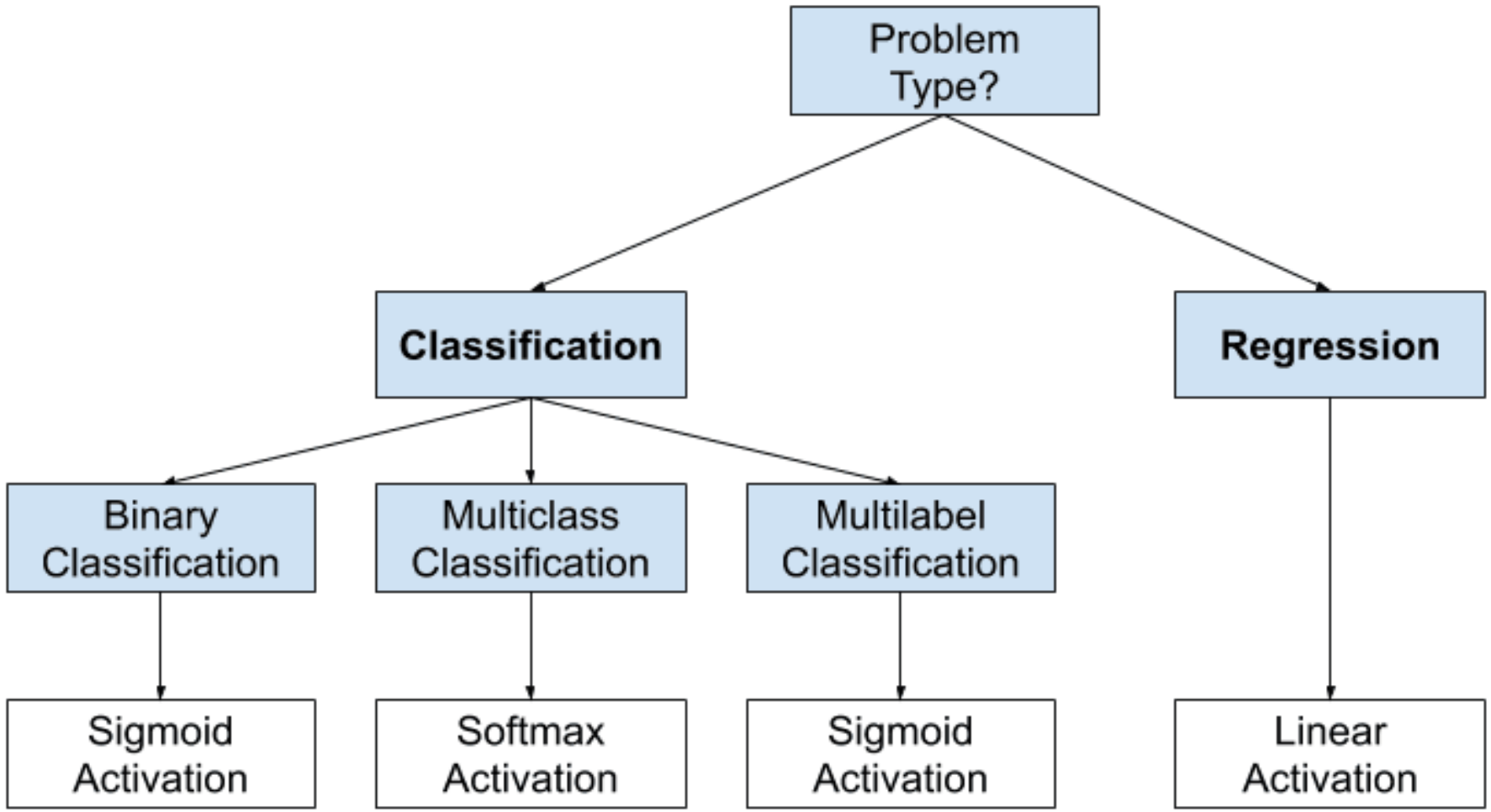
Regression
(predict **continuous** value)



Classification
(predict **discrete** value)



Desired Output Driven by Task



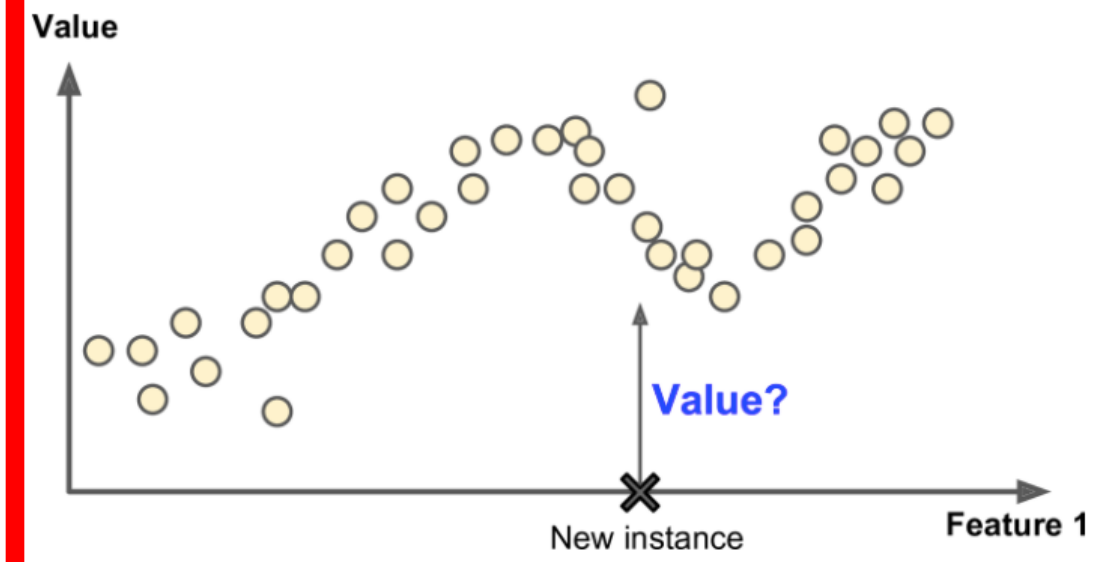
https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

Today's Topics

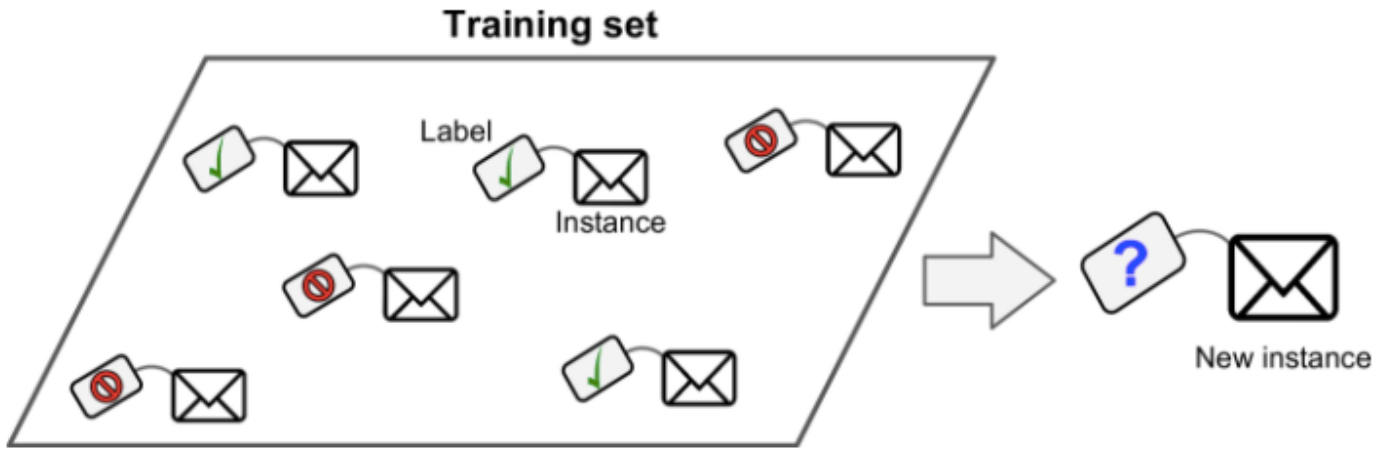
- Motivation for neural networks: need non-linear models
- Neural networks' basic ingredients: hidden layers and activation units
- Neural networks' support for diverse problems: output units
- Objective function: what a model should learn
- Programming tutorial

Have Model Achieve a Specified (Measurable) Goal: Objective Function

Regression
(predict **continuous** value)



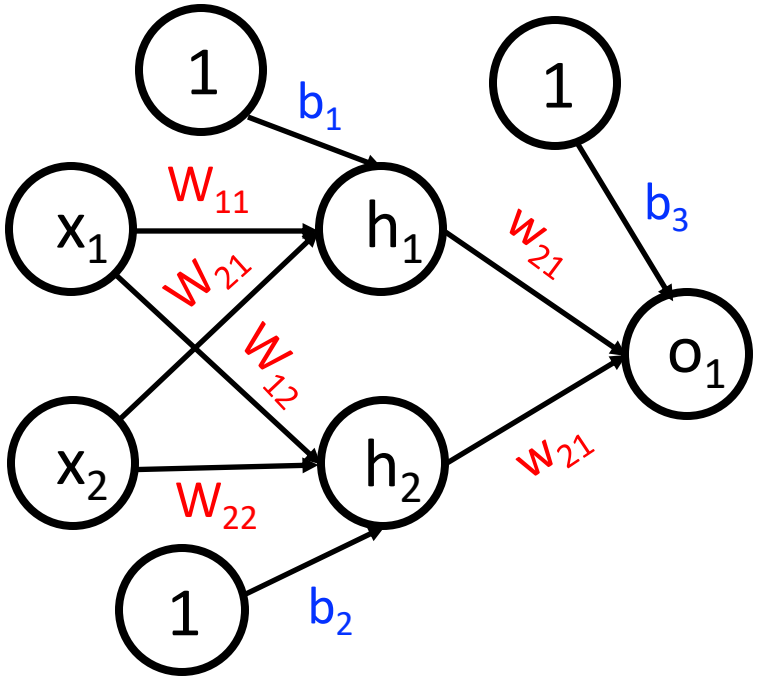
Classification
(predict **discrete** value)



Key question: how do you measure/quantify task success?

Have Model Achieve a Specified (Measurable) Goal: Objective Function

e.g., learn weights and biases that yield the smallest possible squared error (aka, L2 loss, quadratic loss)



Mean taken over *n* instances

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

True value

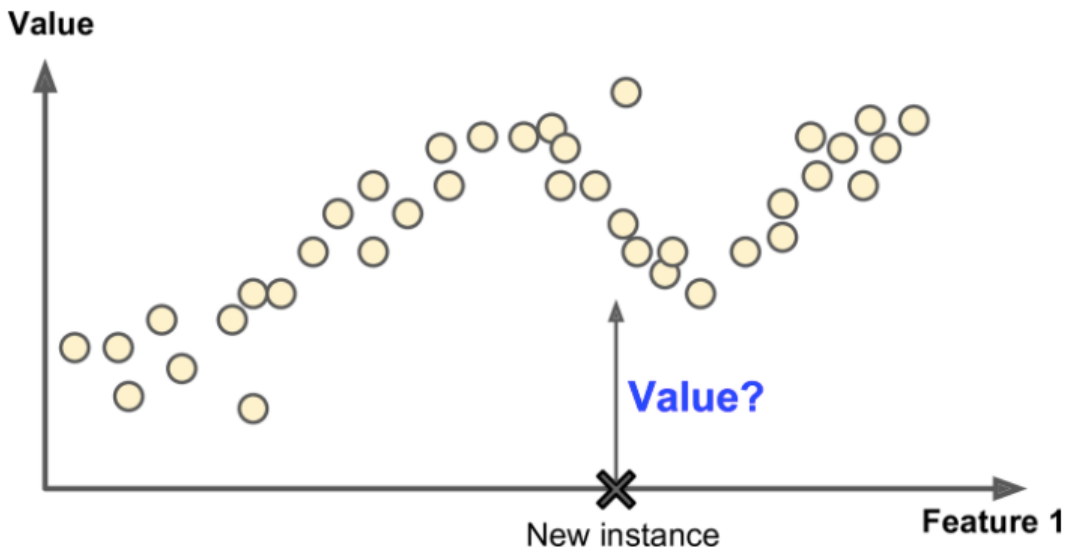
Predicted value

What is the minimum possible value?

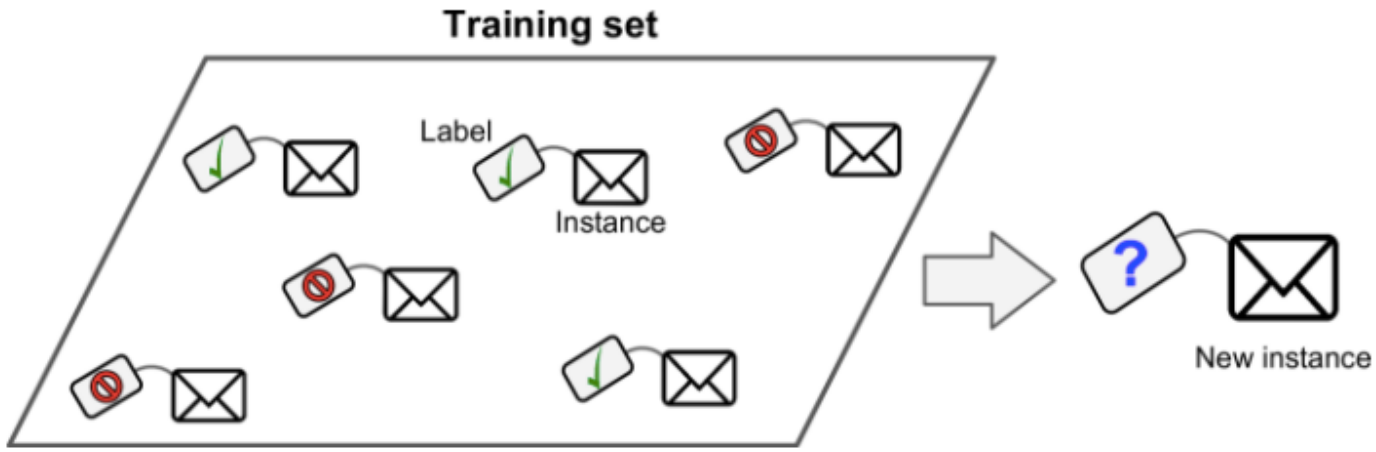
- 0 (i.e., all correct predictions)

Have Model Achieve a Specified (Measurable) Goal: Objective Function

Regression
(predict **continuous** value)

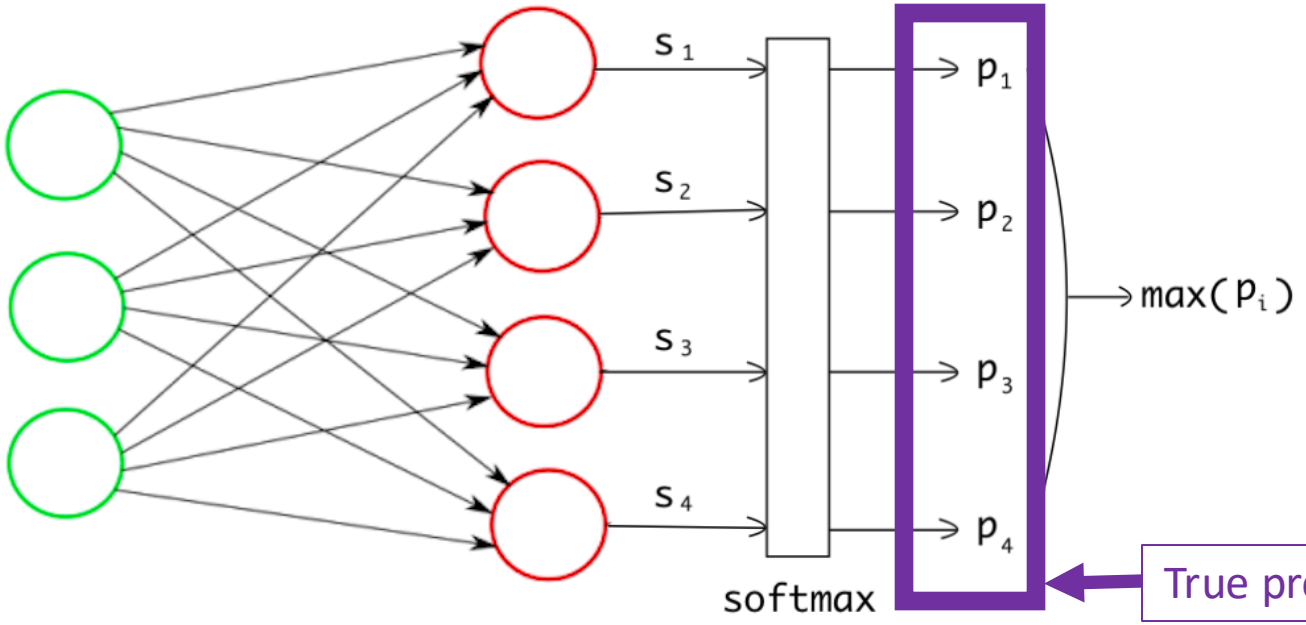


Classification
(predict **discrete** value)



Key question: how do you measure/quantify task success?

Have Model Achieve a Specified (Measurable) Goal: Objective Function



e.g., learn model parameters yielding the smallest possible distance between predicted and true class distributions for the training examples with cross entropy loss

True prediction is 1 hot vector (i.e., one 1 and the rest 0s)

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

Have Model Achieve a Specified (Measurable) Goal: Objective Function

Probability distribution of true class

Probability distribution of predicted class

Number of classes

$$L_{\text{CE}}(\hat{y}, y) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

Recall, truth is set to 1 for one class and 0 otherwise

$$= -\sum_{k=1}^{K} y_k \log \hat{p}(y = k|x)$$

Observed features

$$= -\log \hat{y}_k, \quad \text{(where } k \text{ is the correct class)}$$

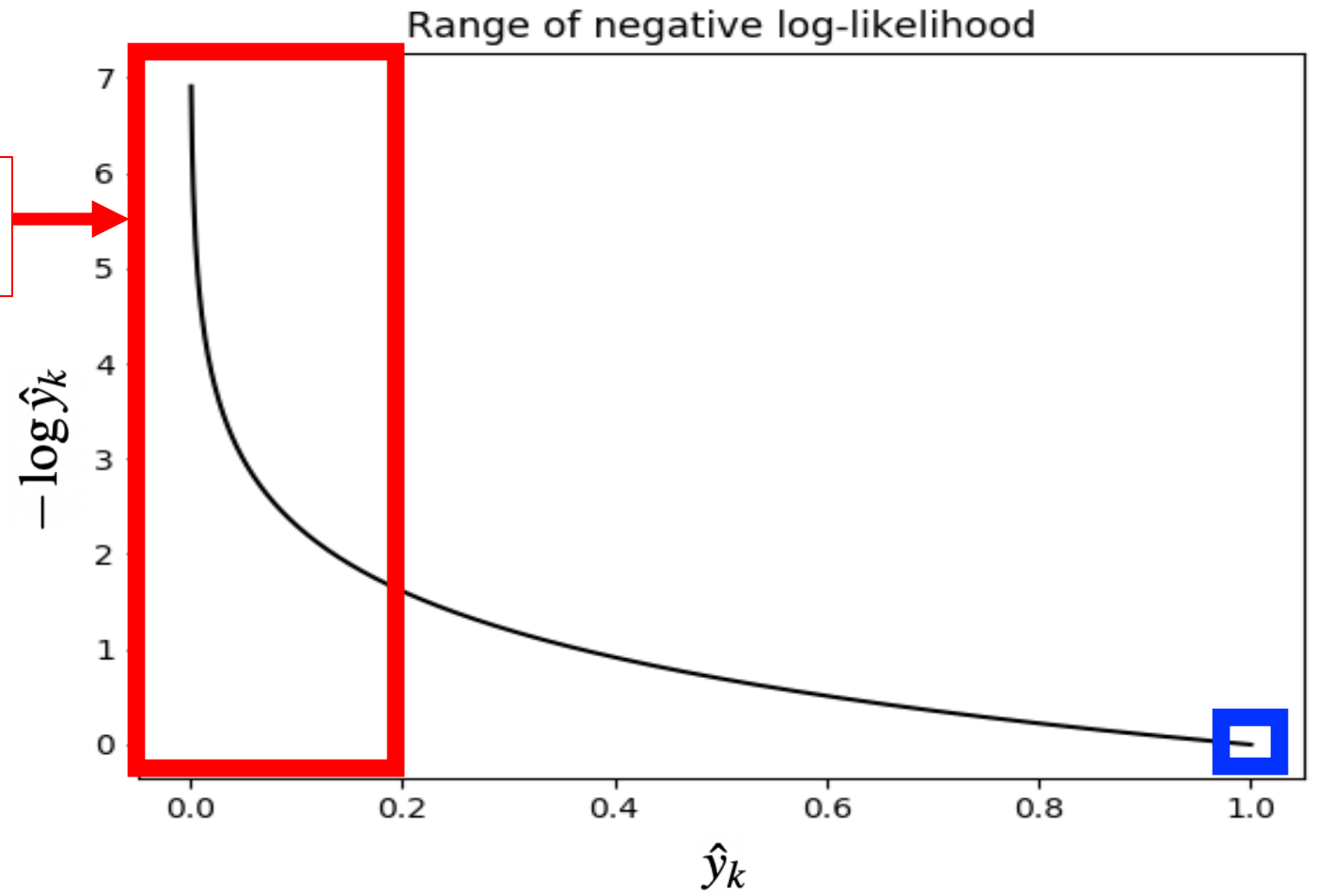
Simplifies to the log of the predicted probability for the correct class (i.e., negative log likelihood loss)

$$= -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

Excellent background: https://web.stanford.edu/~jurafsky/slp3/5.pdf

Have Model Achieve a Specified (Measurable) Goal: Objective Function

More confidently wrong predictions lead to greater error



Range of possible values?

- Minimum: 0
  - i.e., correct prediction: negative log of 1
- Maximum: Infinity
  - i.e., incorrect prediction: negative log of 0

https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/

MANY objective functions exist, and we will examine popular ones in this course

Note: “objective function” is often used interchangeably with “loss function” and “cost function” (more here: https://www.baeldung.com/cs/cost-vs-loss-vs-objective-function)

Key Question: How to Train a Model to Achieve the Objective (Function)?

Stay tuned for the
next three lectures…

Today's Topics

- Motivation for neural networks: need non-linear models
- Neural networks' basic ingredients: hidden layers and activation units
- Neural networks' support for diverse problems: output units
- Objective function: what a model should learn
- Programming tutorial

Today's Topics

- Motivation for neural networks: need non-linear models
- Neural networks' basic ingredients: hidden layers and activation units
- Neural networks' support for diverse problems: output units
- Objective function: what a model should learn
- Programming tutorial


The End