Artificial Neurons

Danna Gurari
University of Colorado Boulder
Fall 2022

Review

- Last lecture:
  - Deep learning applications
  - History of neural networks and deep learning
  - How does a machine learn?
  - Course logistics

- Assignments (Canvas):
  - Problem set 1 due next week

- Questions?

Today's Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neuron: Perceptron

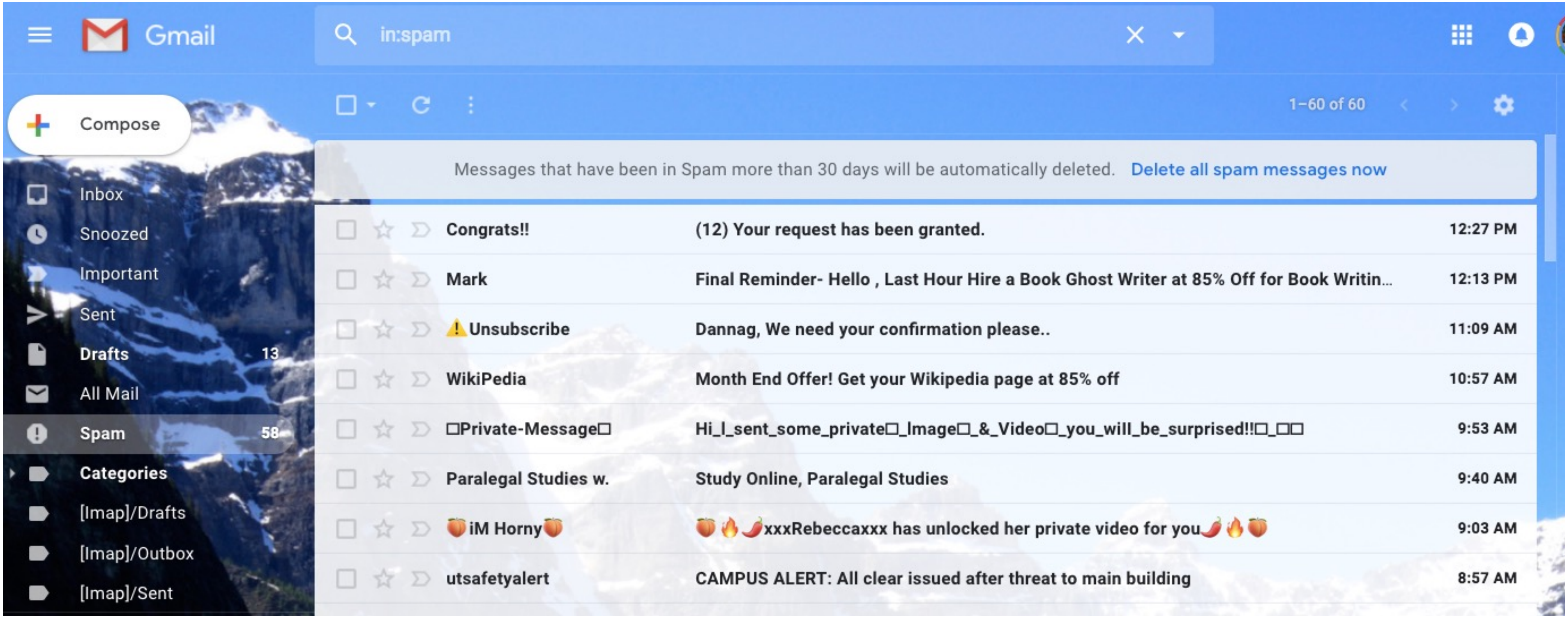
Today's Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neuron: Perceptron

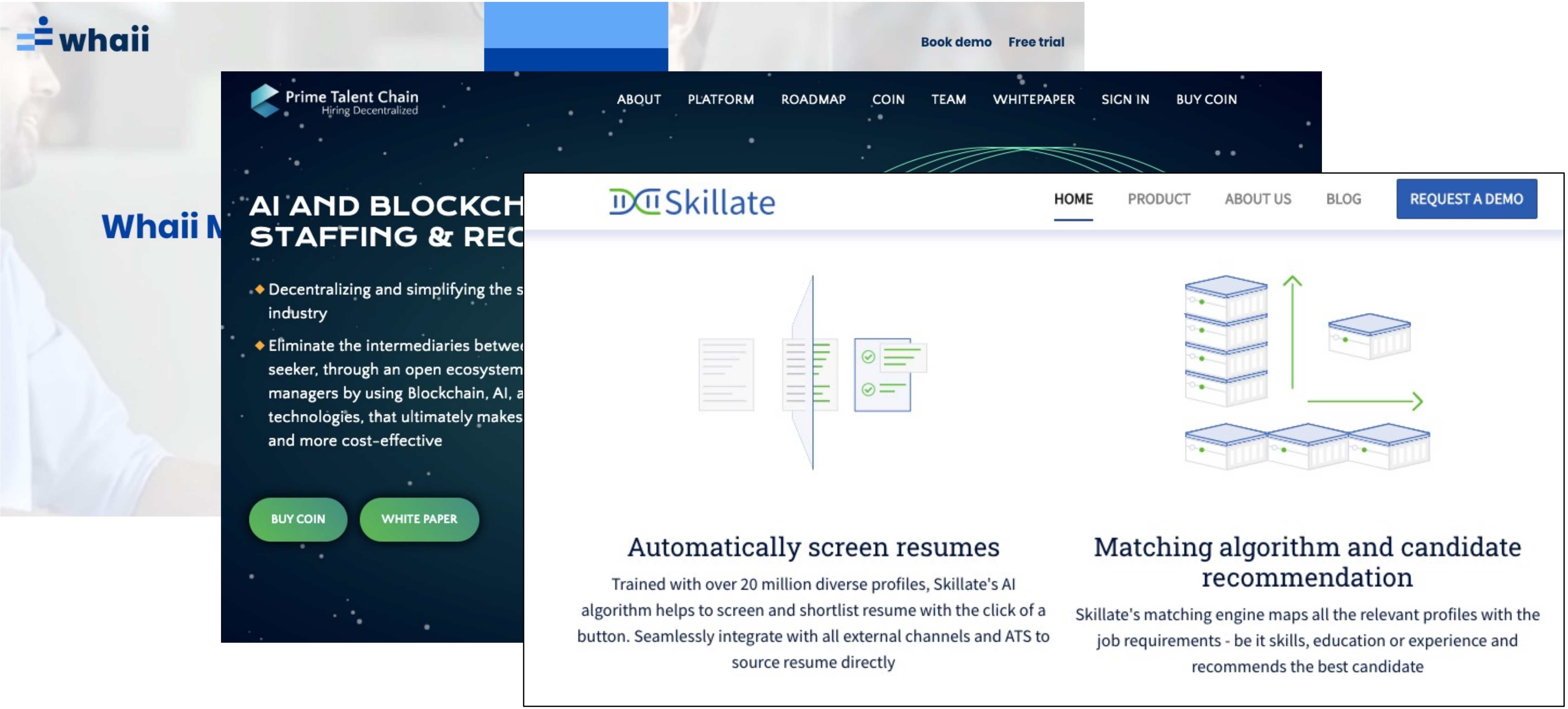
Today’s Scope: Binary Classification

Distinguish 2 classes

Binary Classification: Spam Detection



Binary Classification: Resume Pre-Screening



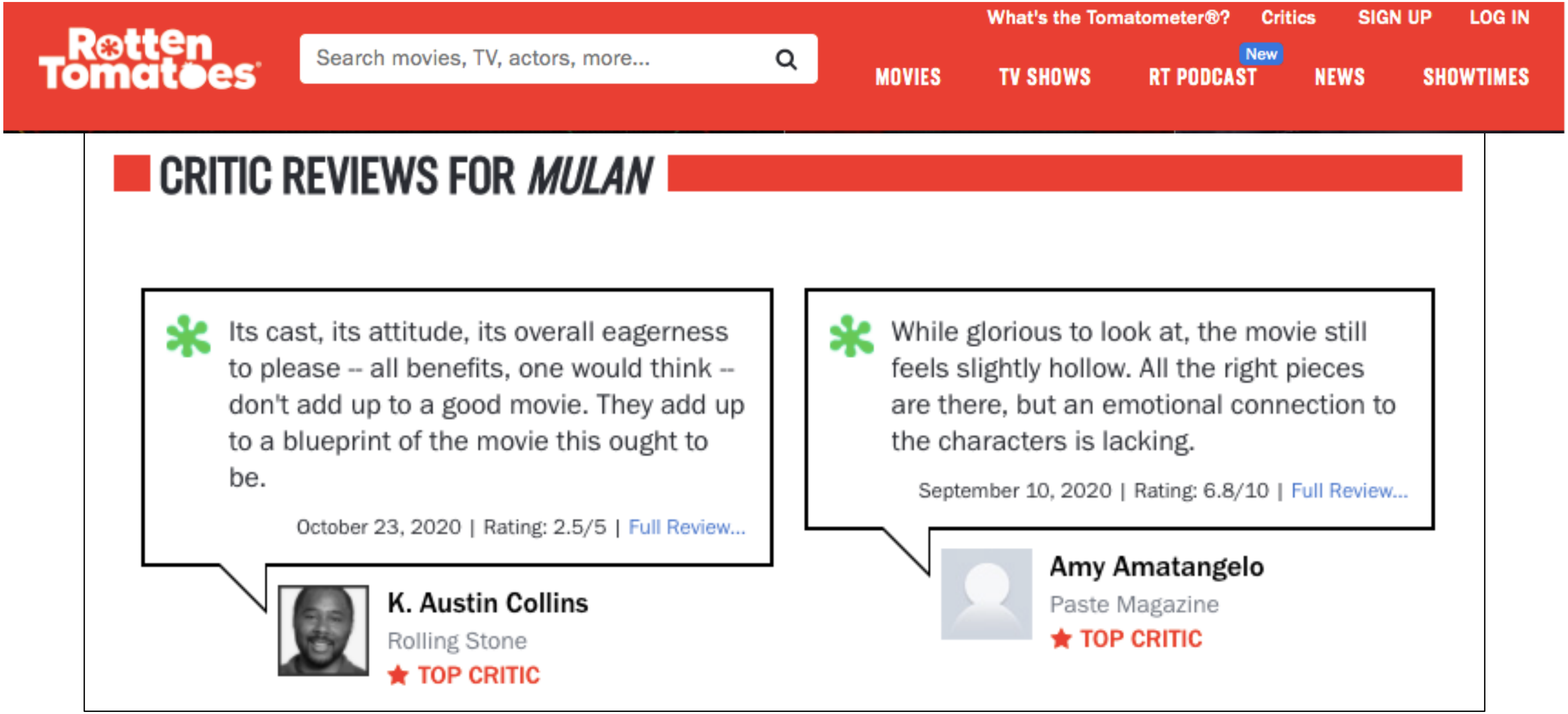
Binary Classification: Cancer Diagnosis

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-health/?utm_medium=tr_social&utm_campaign=site_visitor.unpaid.engagement&utm_source=Twitter#Echobox=1579899156

Binary Classification: Sentiment Analysis



Binary Classification: Food Quality Control

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

Can you think of other binary classification applications?

Today’s Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neuron: Perceptron

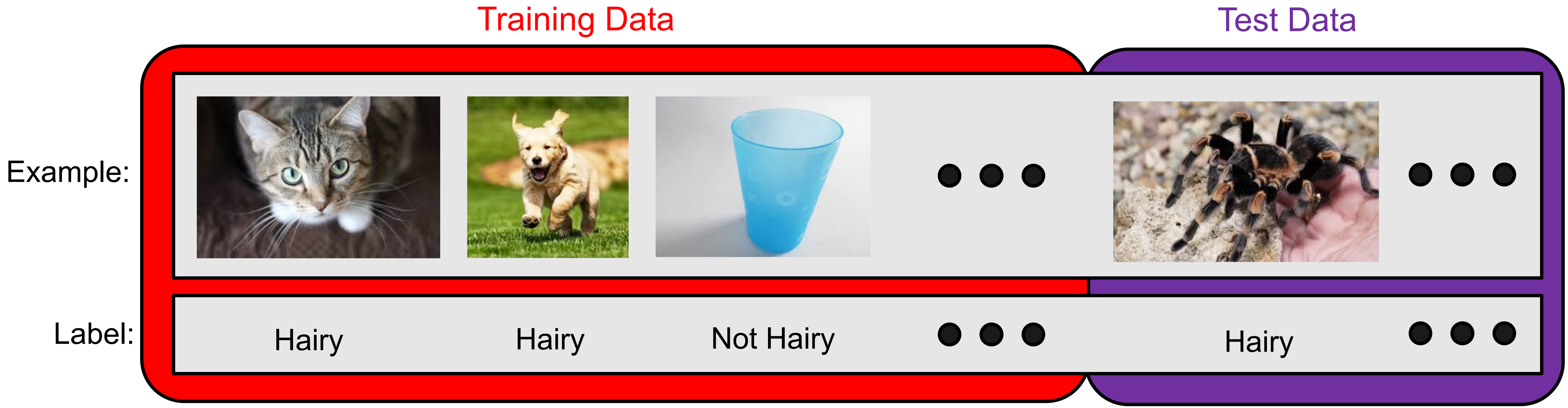
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

Example:    ● ● ●  ● ● ●

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

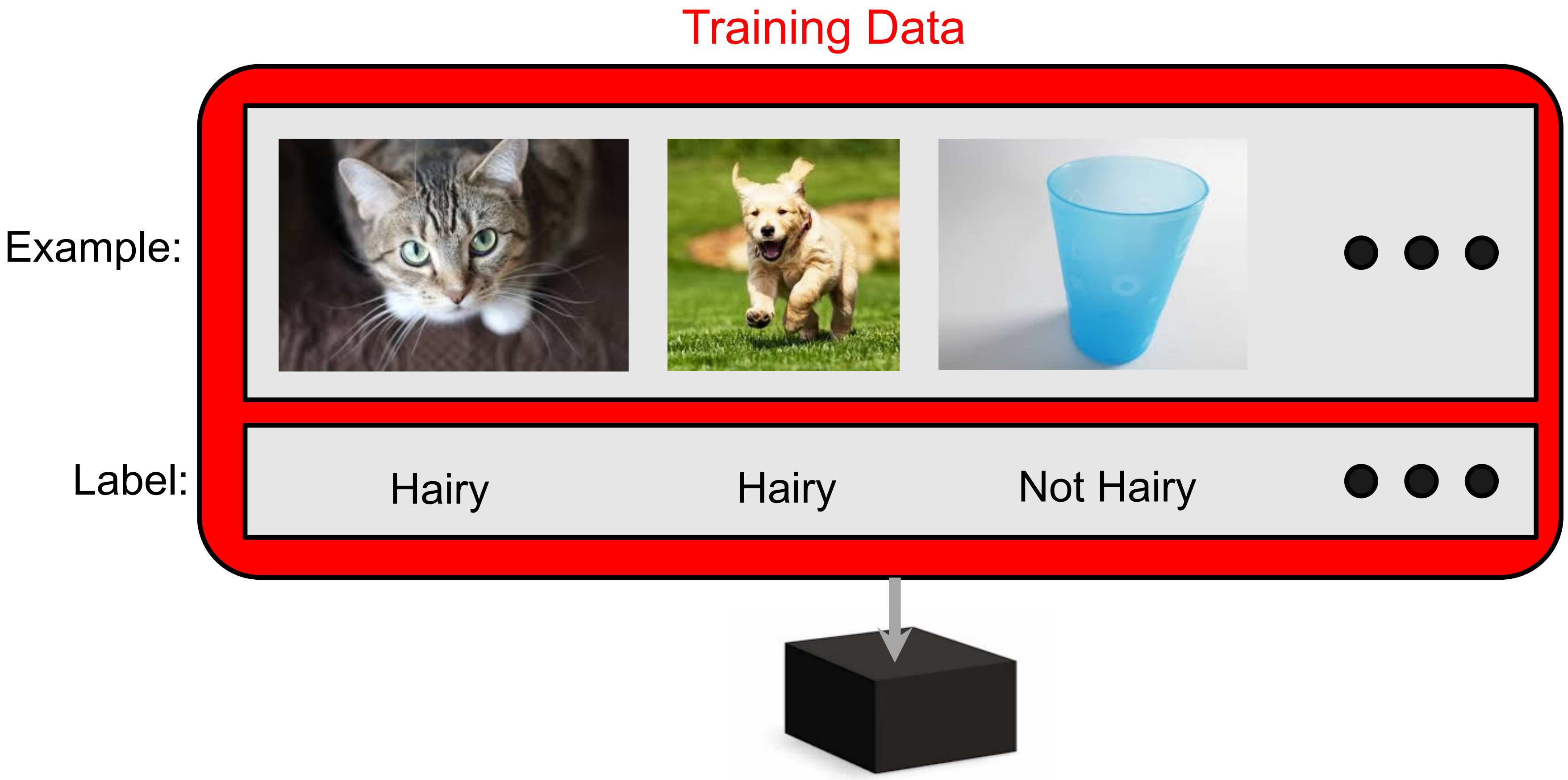
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

1. Split data into a “training set” and “test set ”



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

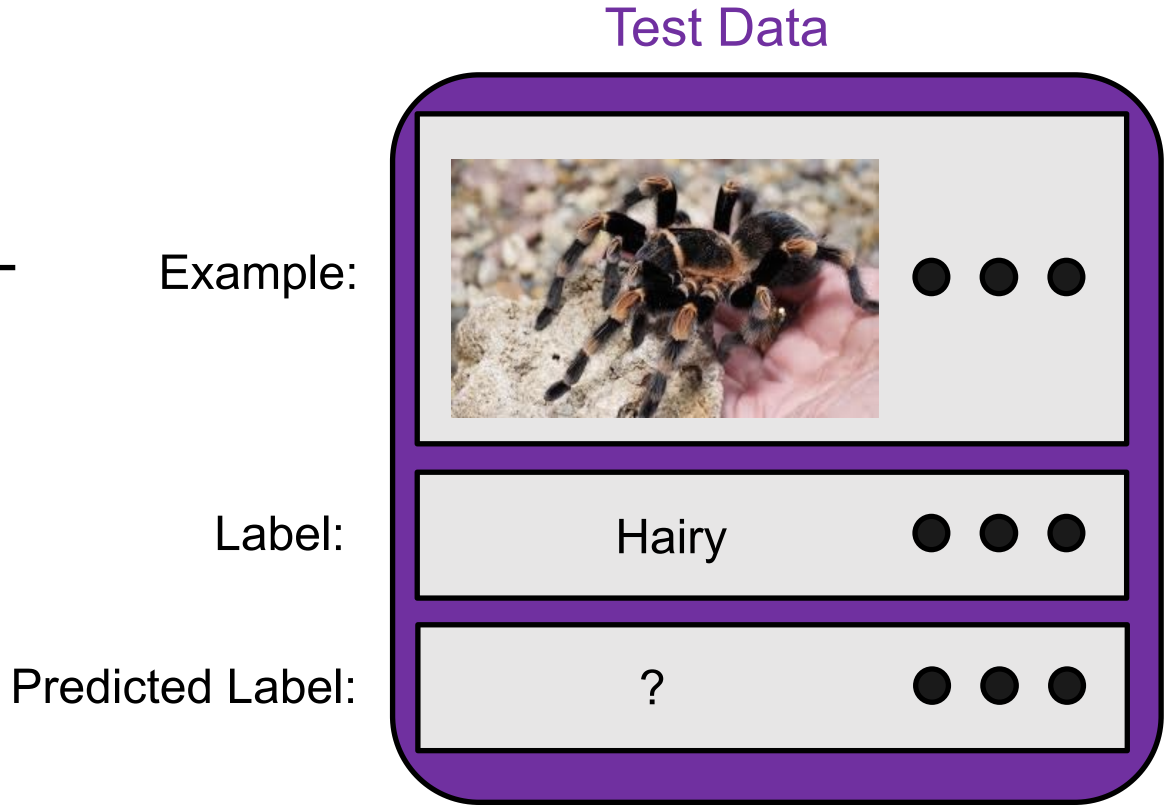
2. Train model on "training set" to try to minimize prediction error on it



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

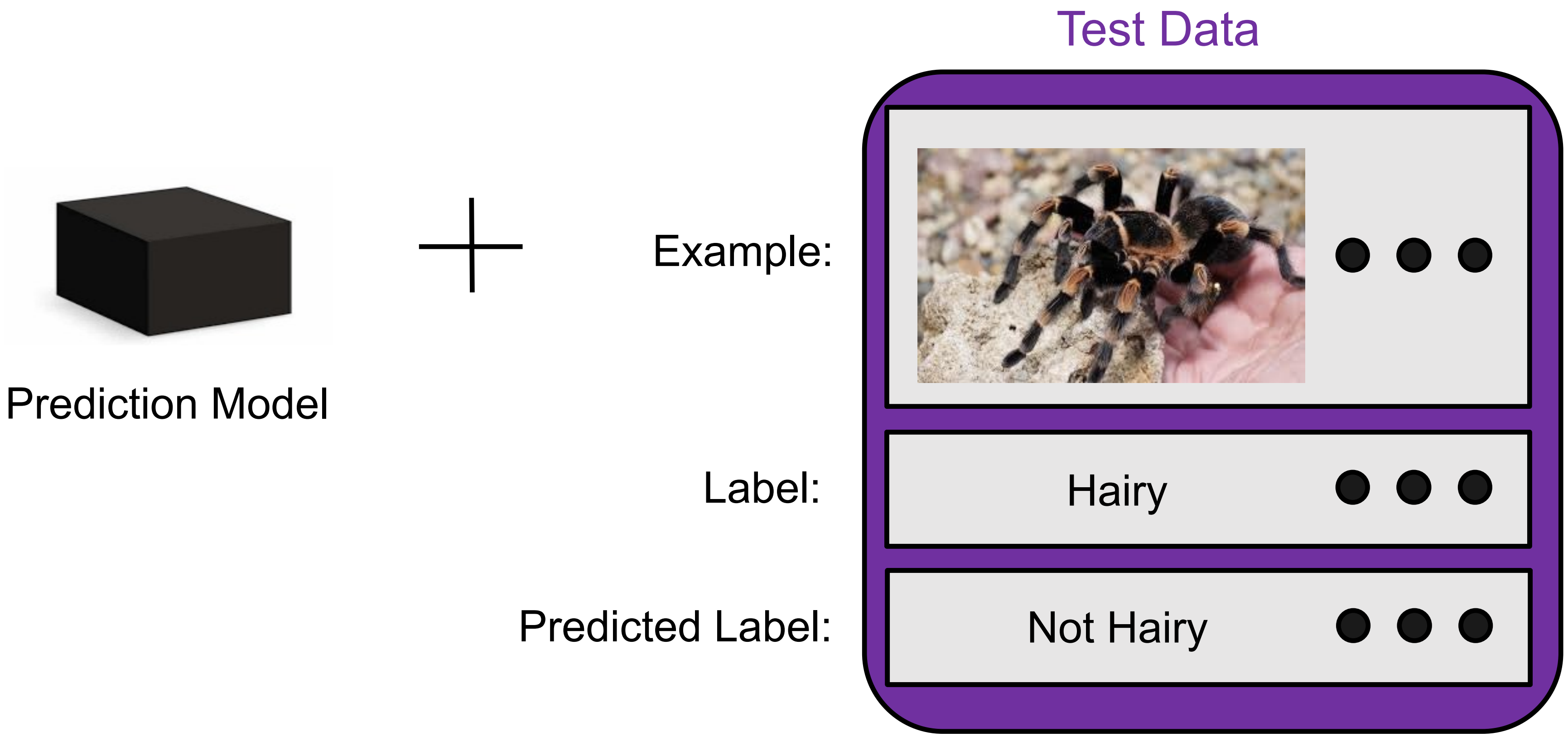
3. Apply trained model on "test set" to measure generalization error

Prediction Model



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

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

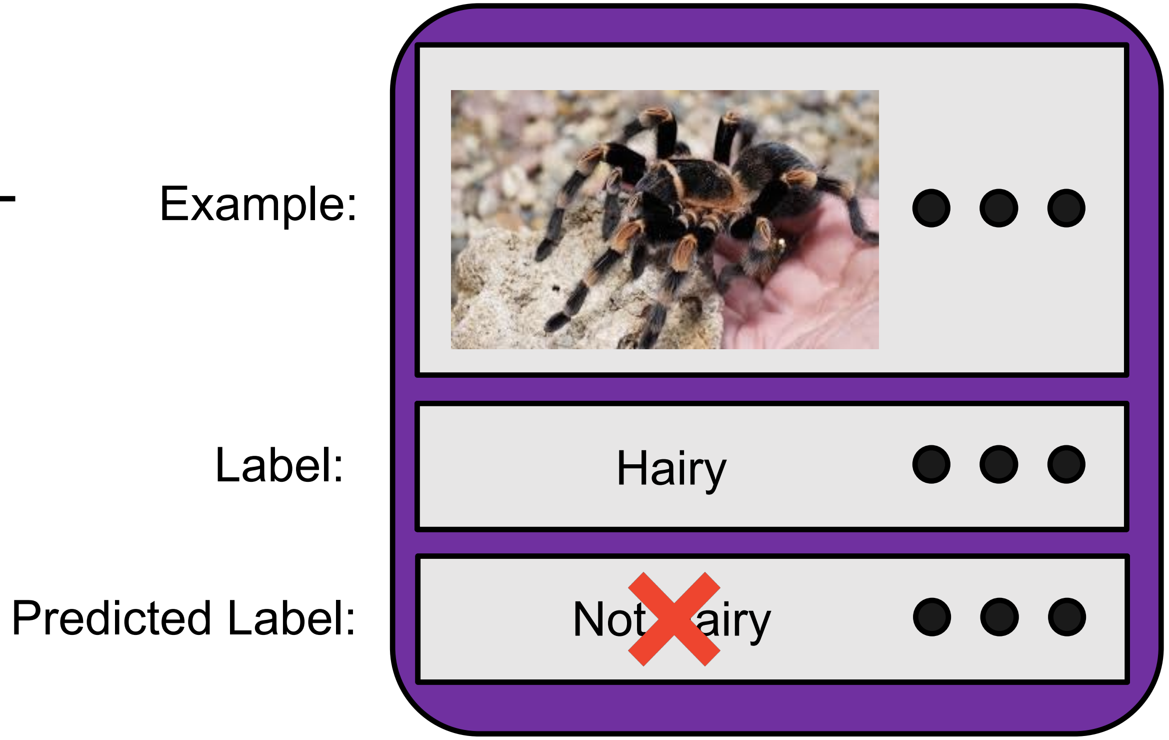


Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

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

Prediction Model

Test Data



Evaluation Methods: Confusion Matrix

		Actual	
		Spam	Not spam
Predicted	Spam	TP	FP
	Not spam	FN	TN

TP = true positive
TN = true negative
FP = false positive
FN = false negative

Evaluation Methods : Descriptive Statistics

e.g.,

		Actual	
		Spam	Not spam
Predicted	Spam	50	10
	Not spam	15	100

Commonly-used statistical descriptions:

- How many ***actual spam*** results are there? - 65
- How many ***actual trusted*** results are there? - 110
- How many ***correctly classified instances***? - 150/175 ~ 86%
- How many ***incorrectly classified instances***? - 25/175 ~ 14%

- What is the ***precision***? $\frac{TP}{TP + FP}$
  - 50/(50+10) ~ 83%

- What is the ***recall***? $\frac{TP}{TP + FN}$
  - 50/(50+15) ~ 77%

Group Discussion

- Which of these evaluation metrics would you use versus not use and why?
  - Accuracy (percentage of correctly classified examples)
  - Precision
  - Recall

- Scenario 1: Medical test for a rare disease affecting one in every million people.

- Scenario 2: Deciding which emails to flag as spam.

Today’s Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neuron: Perceptron

Inspiration: Animal's Computing Machinery

Neuron

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

"hot"

https://www.clipart.email/clipart/don
t-touch-hot-stove-clipart-73647.html

"loud"

https://kisselpaso.com/if-the-sun-city-
music-fest-gets-too-loud-there-is-a-
phone-number-you-can-call-to-complain/

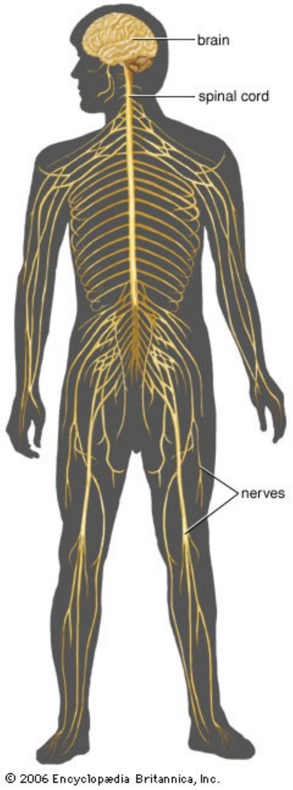
"spicy"

https://www.babycenter.com/404_when-
can-my-baby-eat-spicy-foods_1368539.bc

Inspiration: Animal’s Computing Machinery

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

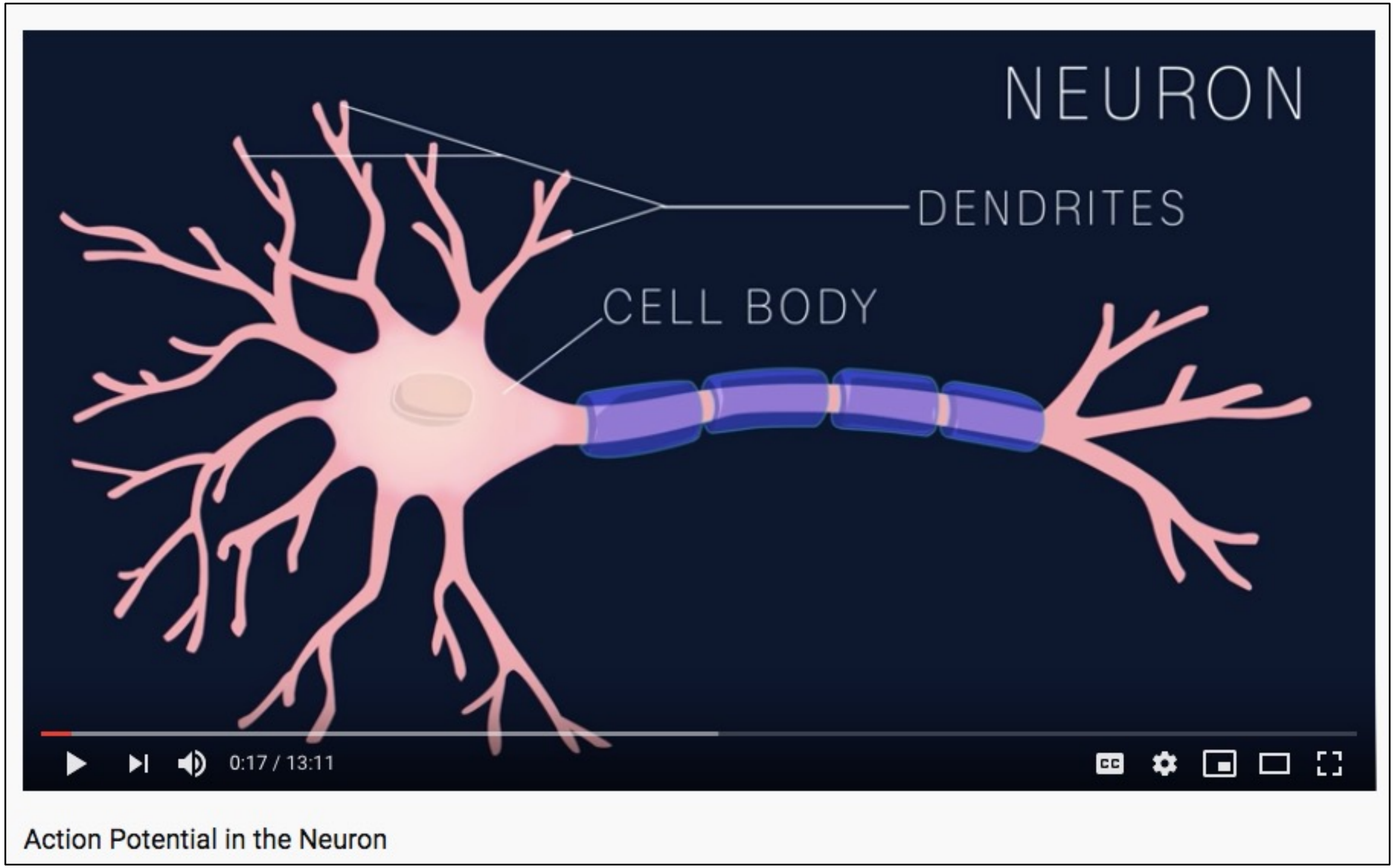
Nematode worm: 302 neurons



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

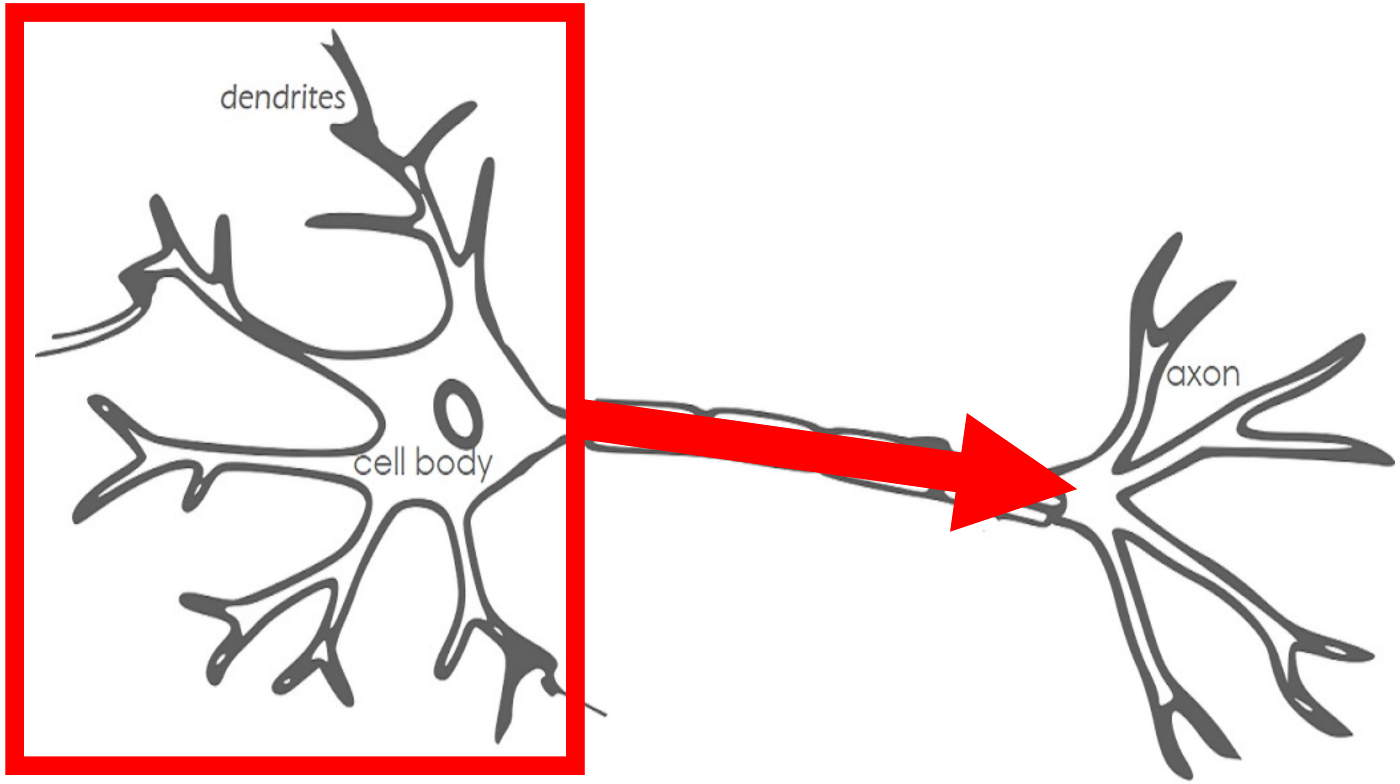
Human: ~100,000,000,000 neurons

Inspiration: Animal's Computing Machinery



Demo (0-1:14): https://www.youtube.com/watch?v=oa6rvUJlg7o

Inspiration: Basic Understanding of Neurons



- When the input signals exceed a certain threshold within a short period of time, a neuron "fires"
- Neuron "firing" is an "all-or-none" process, where either a signal is sent or nothing happens

Image Source: https://becominghuman.ai/introduction-to-neural-networks-bd042ebf2653

Sidenote: It Remains An Open Research Problem to Understand How Individual Neurons Work

Today's Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neuron: Perceptron

Historical Context: Artificial Neurons

First mathematical model of neuron

1943 1945 1950 1956 1957 1959

First programmable machine

Turing test

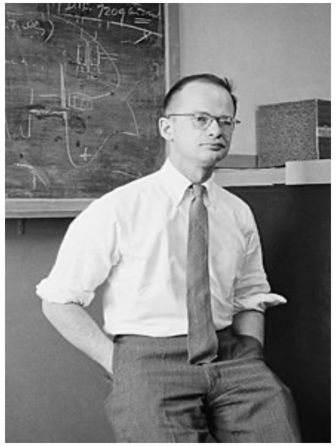
AI

Perceptron

Machine learning

Warren McCulloch (Neurophysiologist)
http://web.csulb.edu/~cwallis/artificialn/warren_mcculloch.html

Emerges from interdisciplinary collaboration

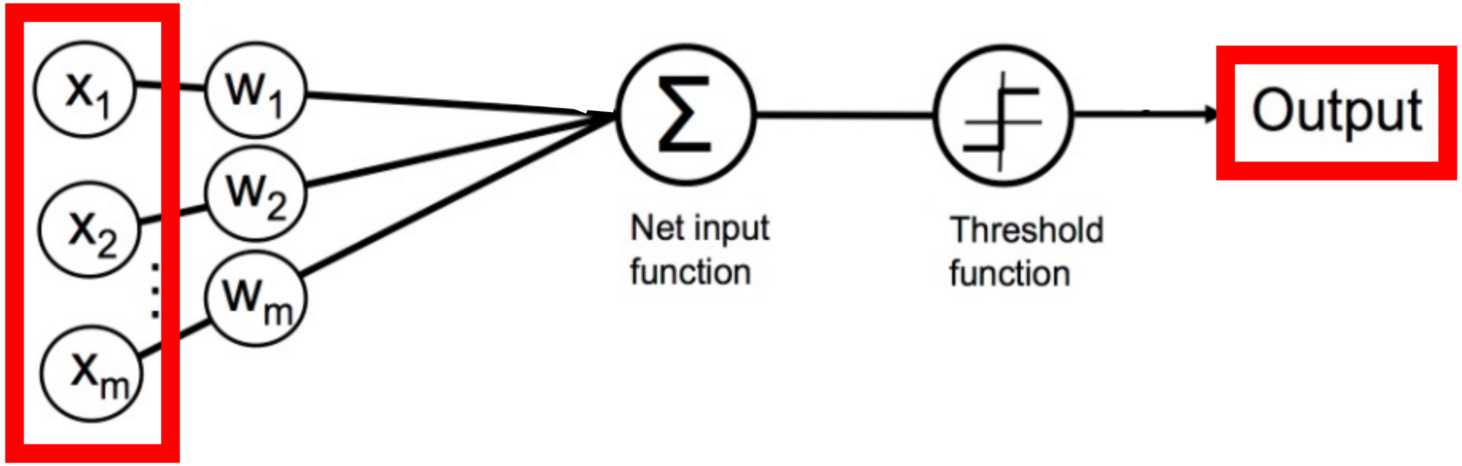


Walter Pitts (Mathematician)
https://en.wikipedia.org/wiki/Walter_Pitts

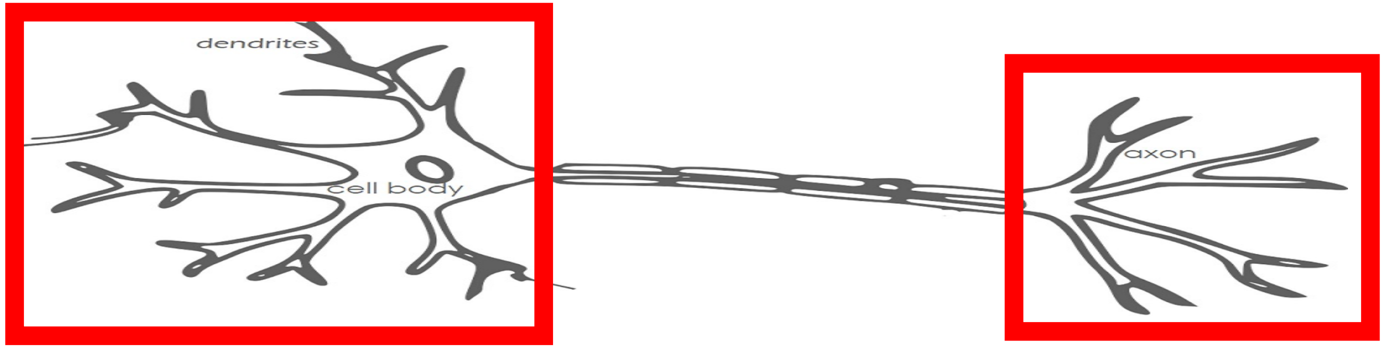
Artificial Neuron: McCulloch-Pitts Neuron

"Input signals" "Output signal"

Artificial Neuron:



Biological Neuron:



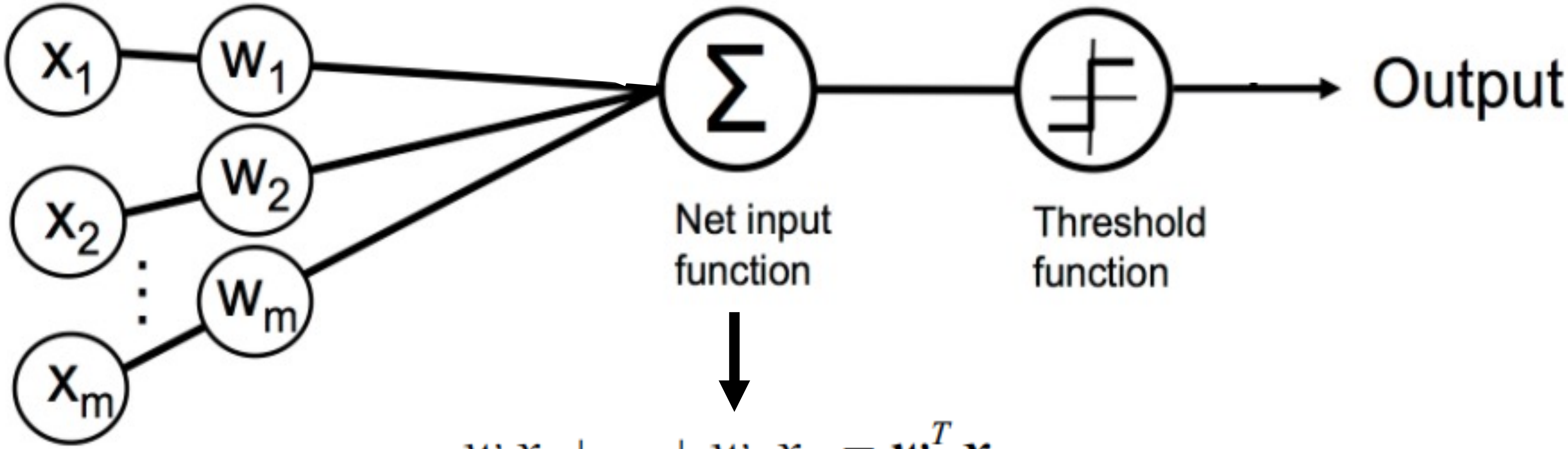
Python Machine Learning; Raschka & Mirjalili

Image Source: https://becominghuman.ai/introduction-to-neural-networks-bd042ebf2653

Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed

- outputs 1 or 0 (mimics neurons by "firing" only when aggregate value exceeds threshold)



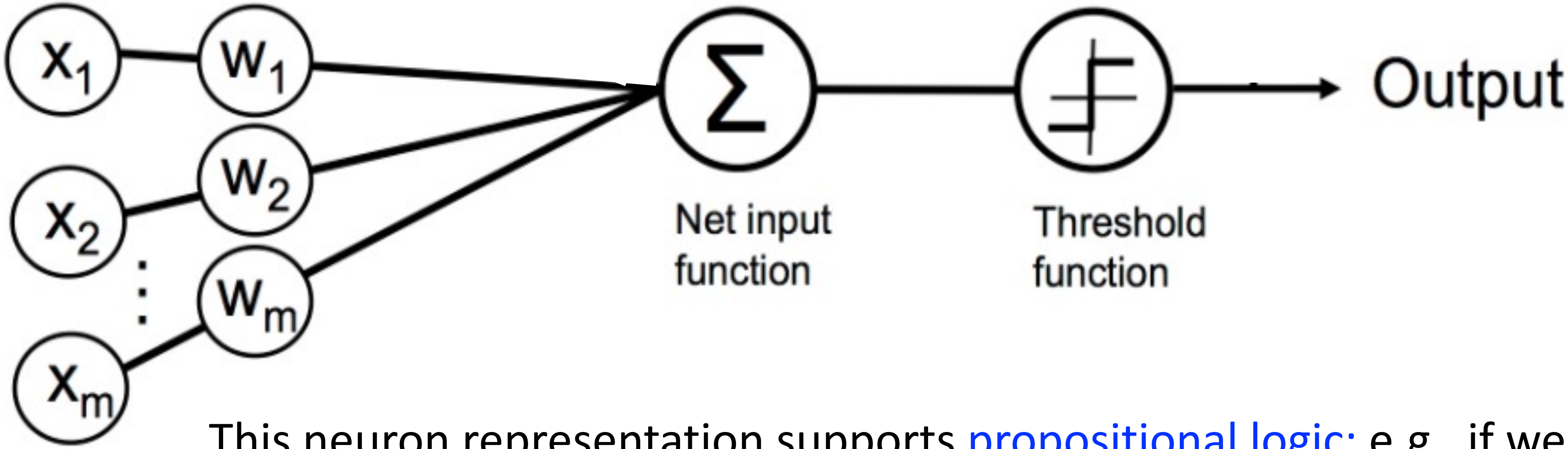
$$w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

Figure source: Python Machine Learning; Raschka & Mirjalili
Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943

Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed

- outputs 1 or 0 (mimics neurons by "firing" only when aggregate value exceeds threshold)



This neuron representation supports propositional logic; e.g., if weights equal 1 and there are 3 inputs, how is the AND function achieved?

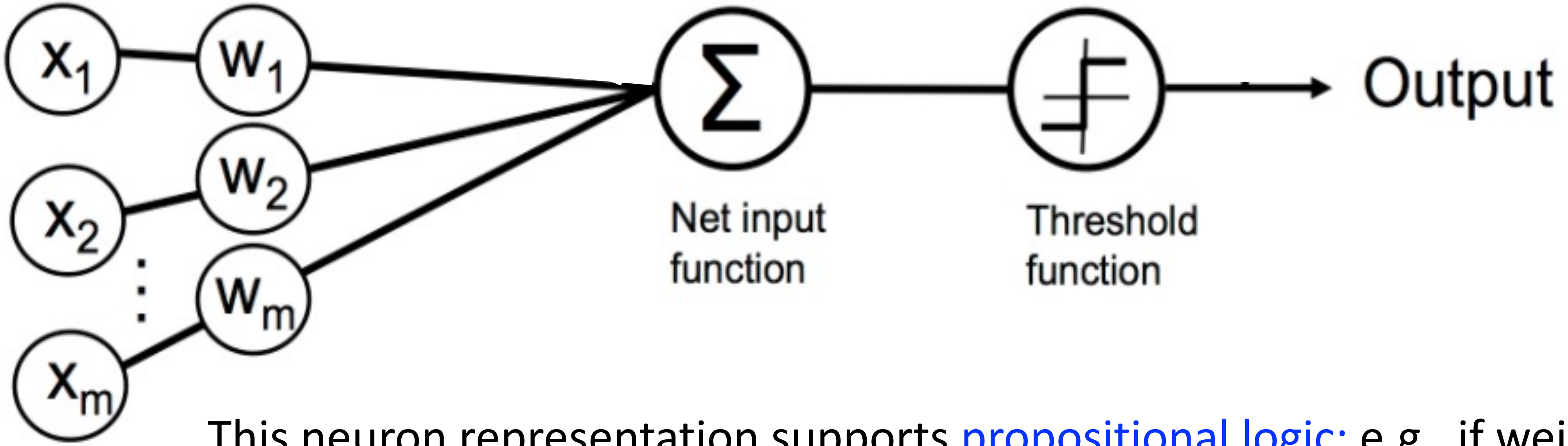
Figure source: Python Machine Learning; Raschka & Mirjalili

Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943

Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed

- outputs 1 or 0 (mimics neurons by “firing” only when aggregate value exceeds threshold)



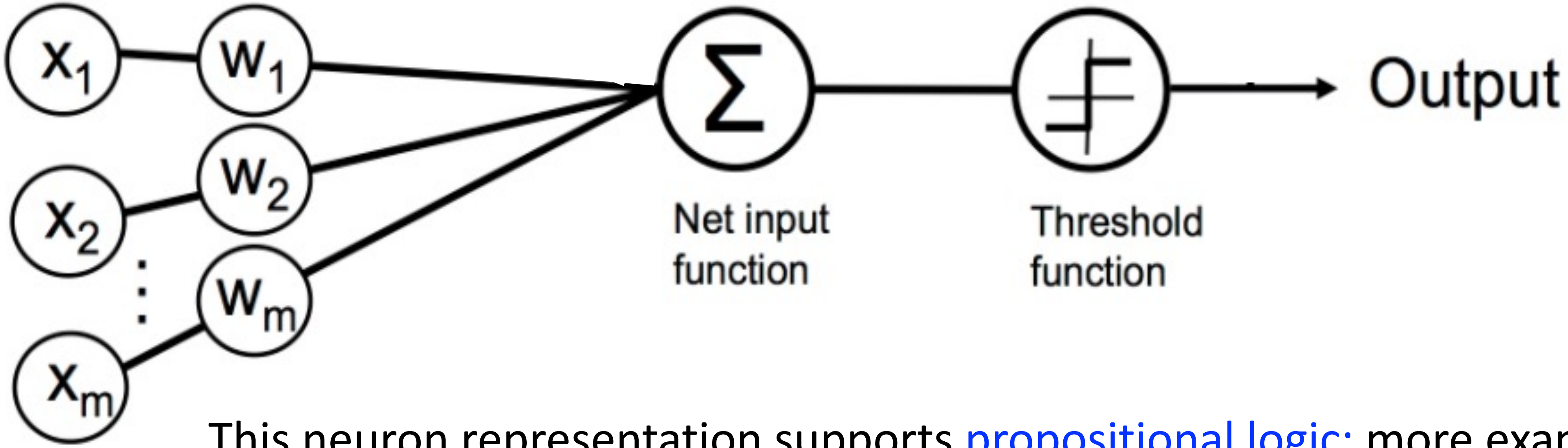
This neuron representation supports propositional logic; e.g., if weights equal 1 and there are 3 inputs, how is the OR function achieved?

Figure source: Python Machine Learning; Raschka & Mirjalili
Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943

Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed

- outputs 1 or 0 (mimics neurons by “firing” only when aggregate value exceeds threshold)



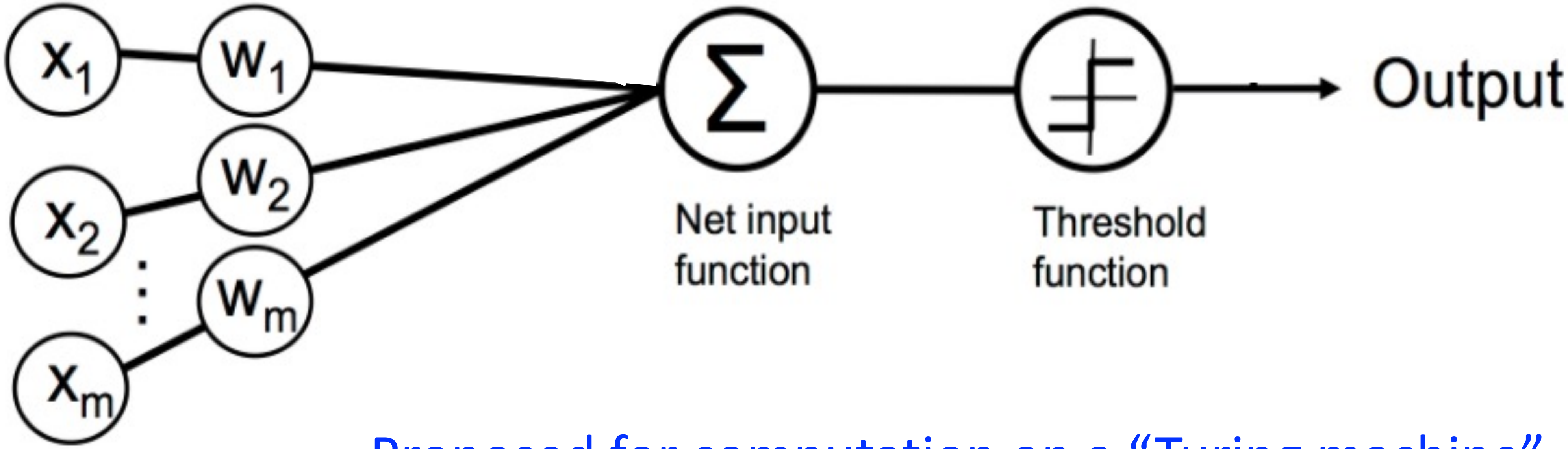
This neuron representation supports propositional logic; more examples found at https://home.csulb.edu/~cwallis/artificialn/History.htm

Figure source: Python Machine Learning; Raschka & Mirjalili
Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943

Artificial Neuron: McCulloch-Pitts Neuron

- inputs (x) and weights (w) can be 0 or 1
- weights (w) and threshold values are fixed

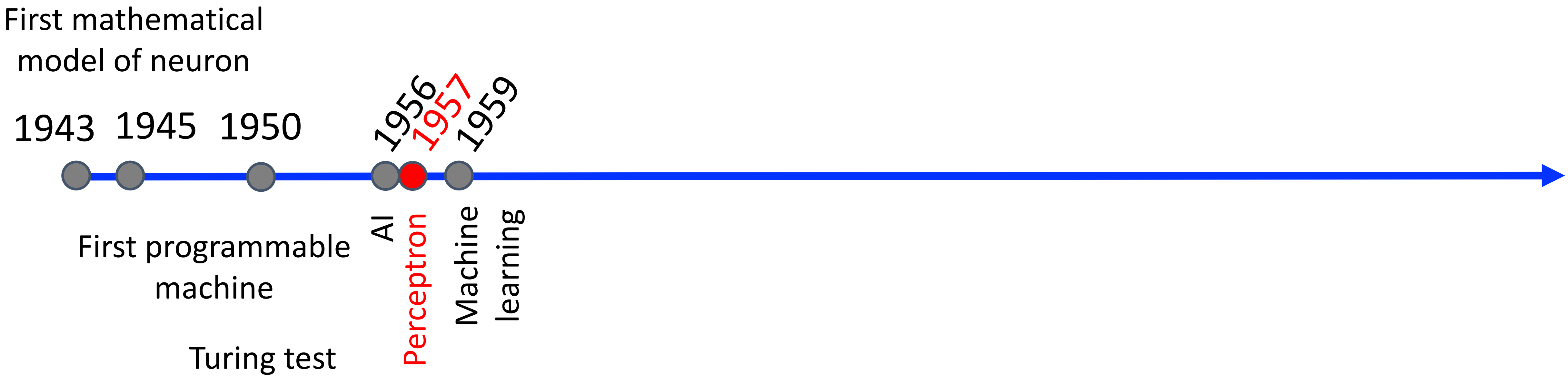
- outputs 1 or 0 (mimics neurons by "firing" only when aggregate value exceeds threshold)



Proposed for computation on a "Turing machine"

Figure source: Python Machine Learning; Raschka & Mirjalili
Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943

Historical Context: Artificial Neurons



Perceptron: Innovator and Vision

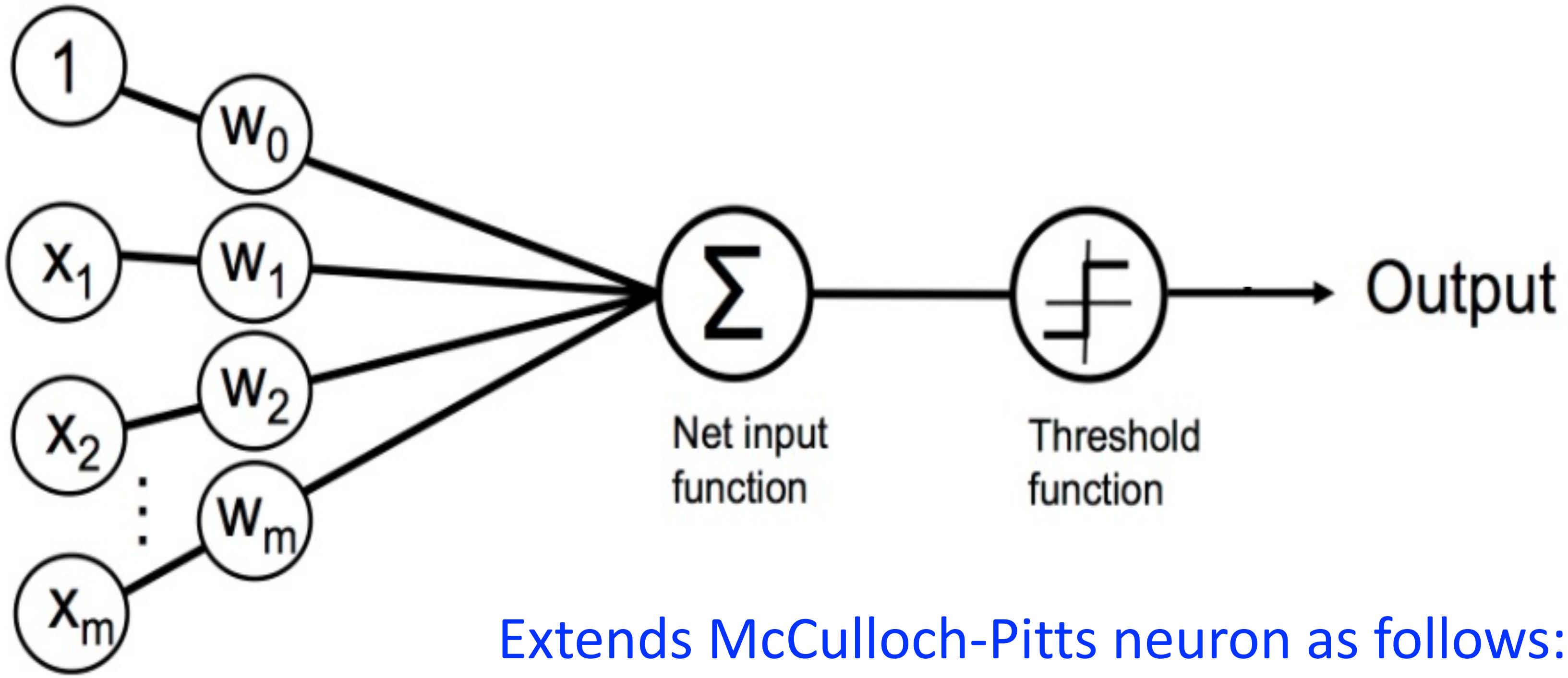
Frank Rosenblatt
(Psychologist)

“[The perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence…. [It] is expected to be finished in about a year at a cost of $100,000.”

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

https://en.wikipedia.org/wiki/Frank_Rosenblatt

Perceptron: Architecture (Linear Threshold Unit)



Extends McCulloch-Pitts neuron as follows:

- inputs and weights can be any value
- weights (W) are learned

Python Machine Learning; Raschka & Mirjalili

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

Perceptron: Architecture (Linear Threshold Unit)

- Function deciding output value ("fire" or not):

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

- Rewriting function:

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

  - Where:

$$z = \underbrace{\boxed{w_0}}_{\boxed{-\theta}}\underbrace{\boxed{x_0}}_{\boxed{1}} + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

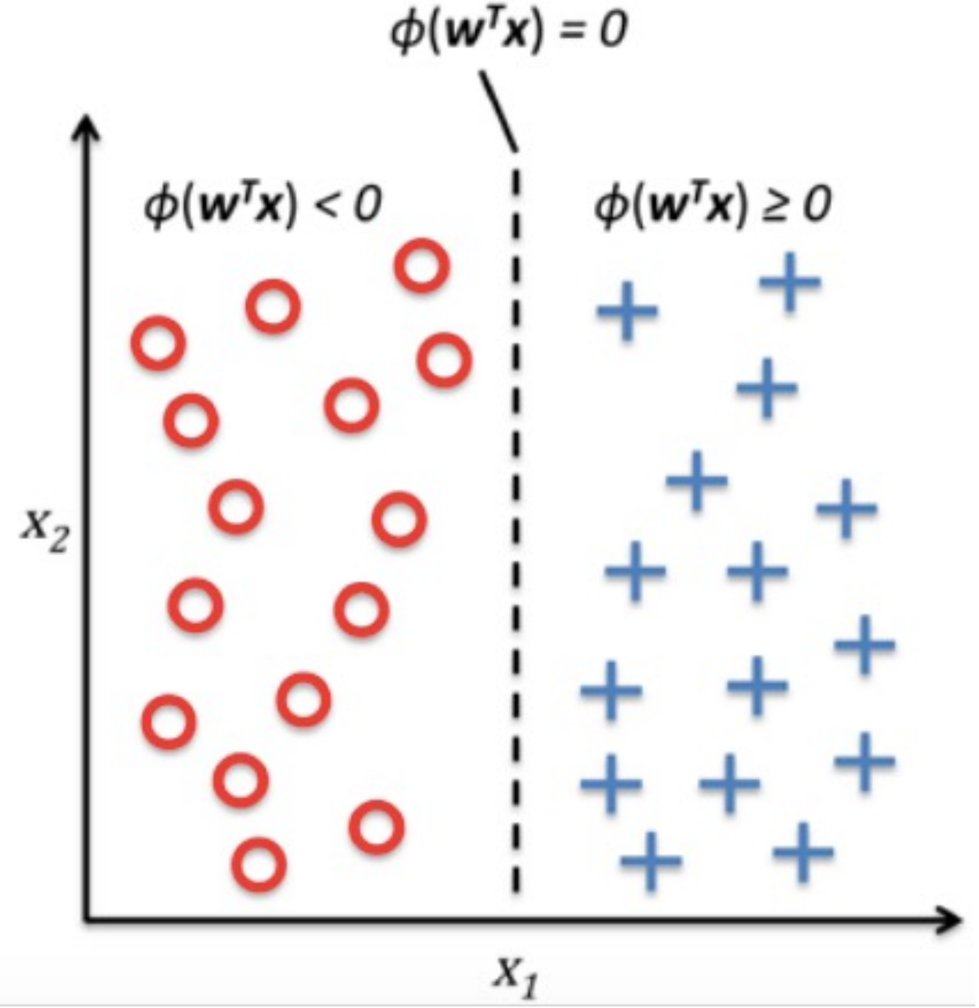
Bias

*** Note: Kamath textbook offers two common conventions for Perceptrons of using two possible output values of {-1, 1} and {0, 1}, in Chapters 2.5 and 4. The output choice dictates whether the threshold should be set to 0.5 or 0.**

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

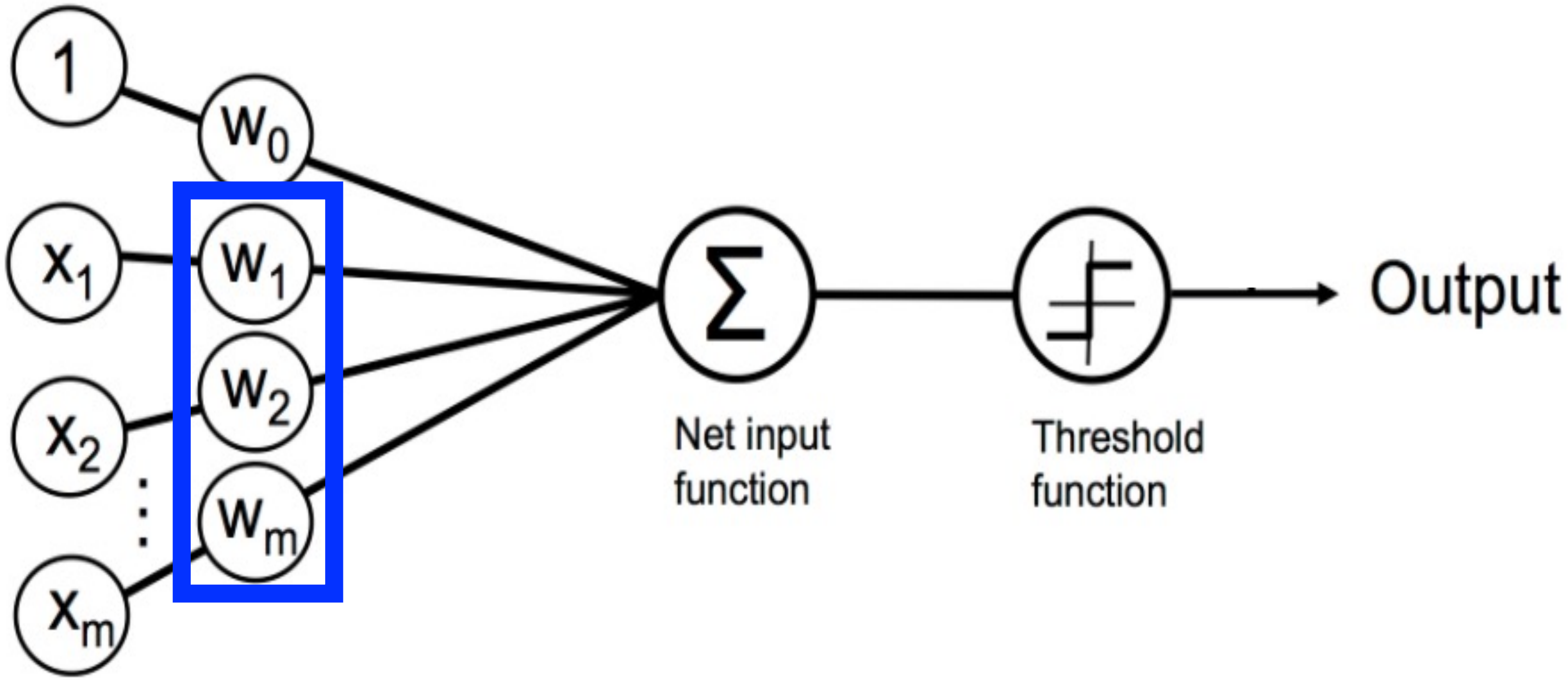
Perceptron: Architecture (Linear Threshold Unit)

Graphical representation:



Python Machine Learning; Raschkka & Mirjalili

Perceptron: Architecture (Linear Threshold Unit)



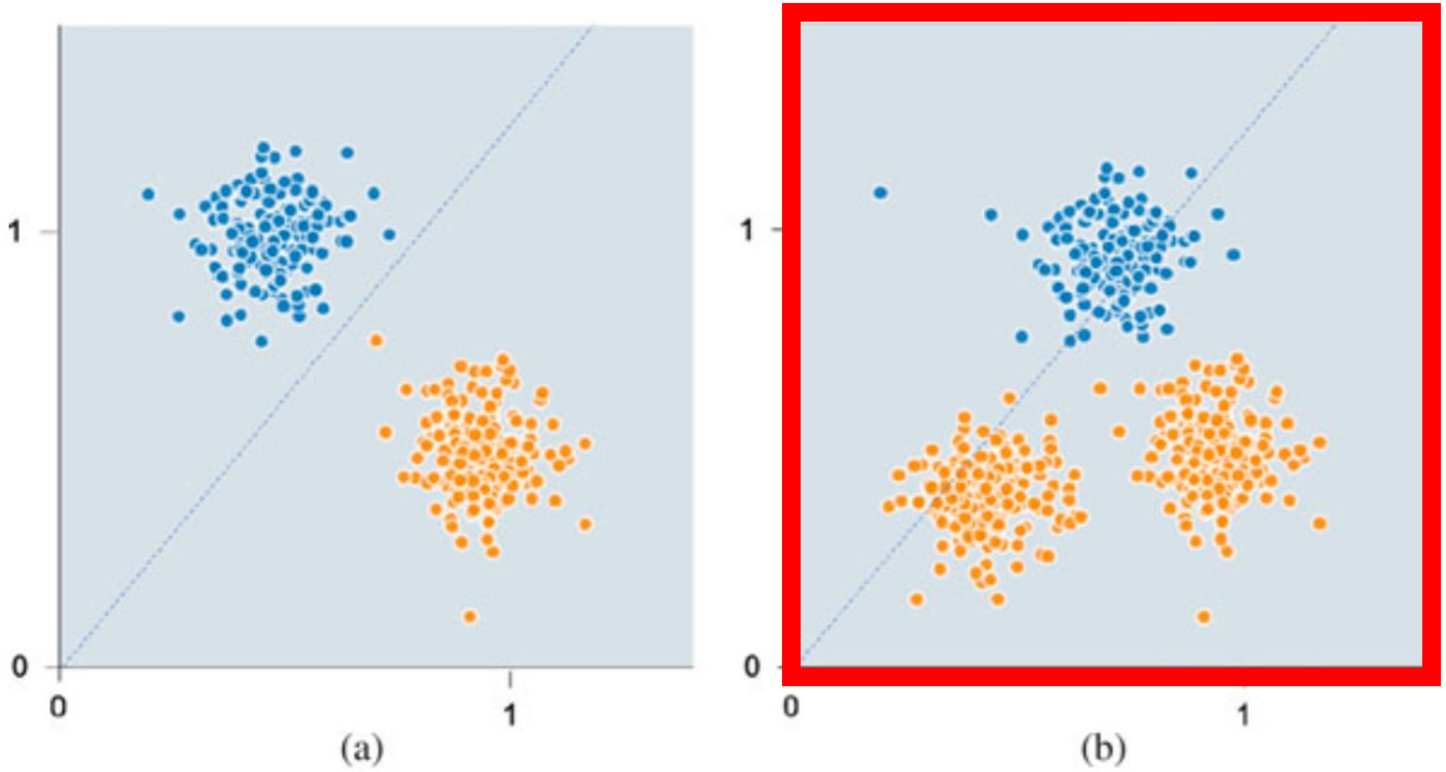
What is the motivation for weights? e.g., for predicting if you will like a movie?

Python Machine Learning; Raschka & Mirjalili

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

Perceptron: Architecture (Linear Threshold Unit)

Motivation for **bias**: without it, model must go through origin



Error

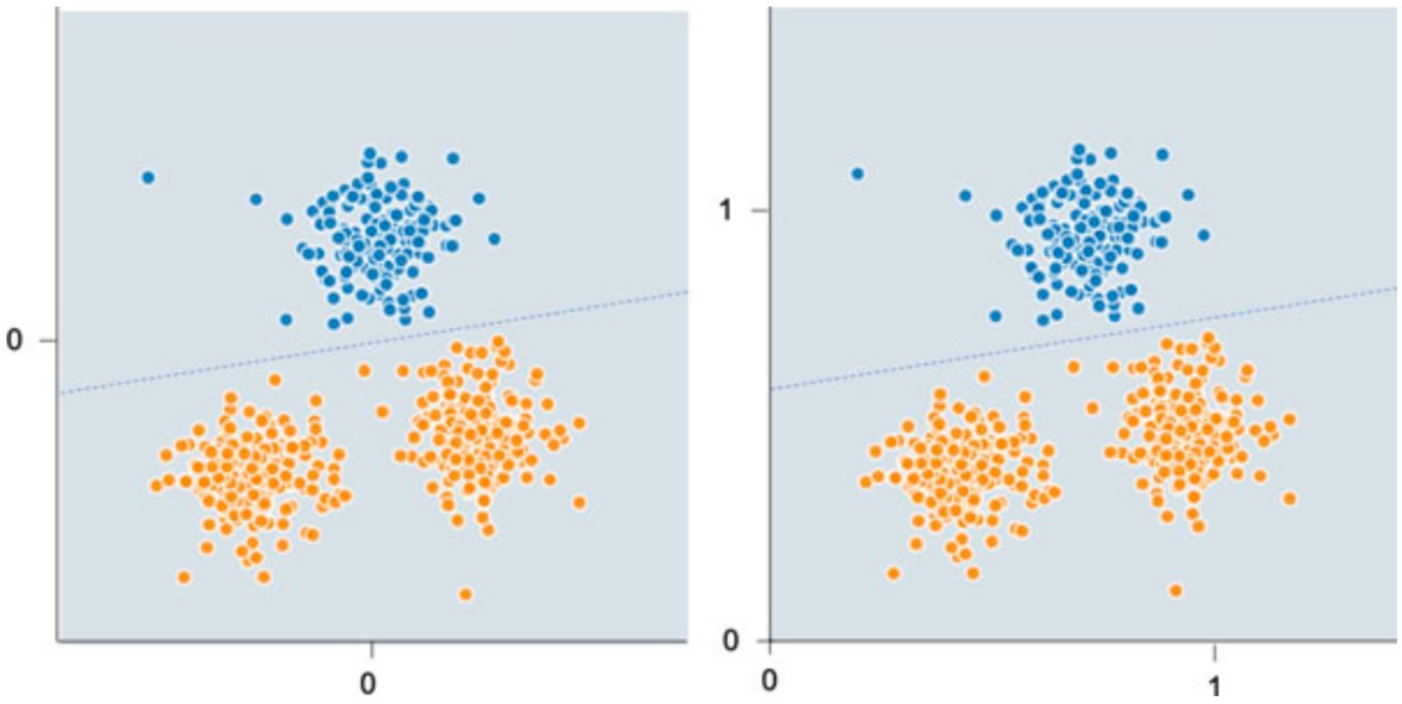
$$z = \boxed{w_0}\boxed{x_0} + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

Bias $\boxed{-\theta}$ $\boxed{1}$

Uday Kamath, John Liu, and James Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Perceptron: Architecture (Linear Threshold Unit)

Motivation for **bias**: with it, model does not have to go through origin

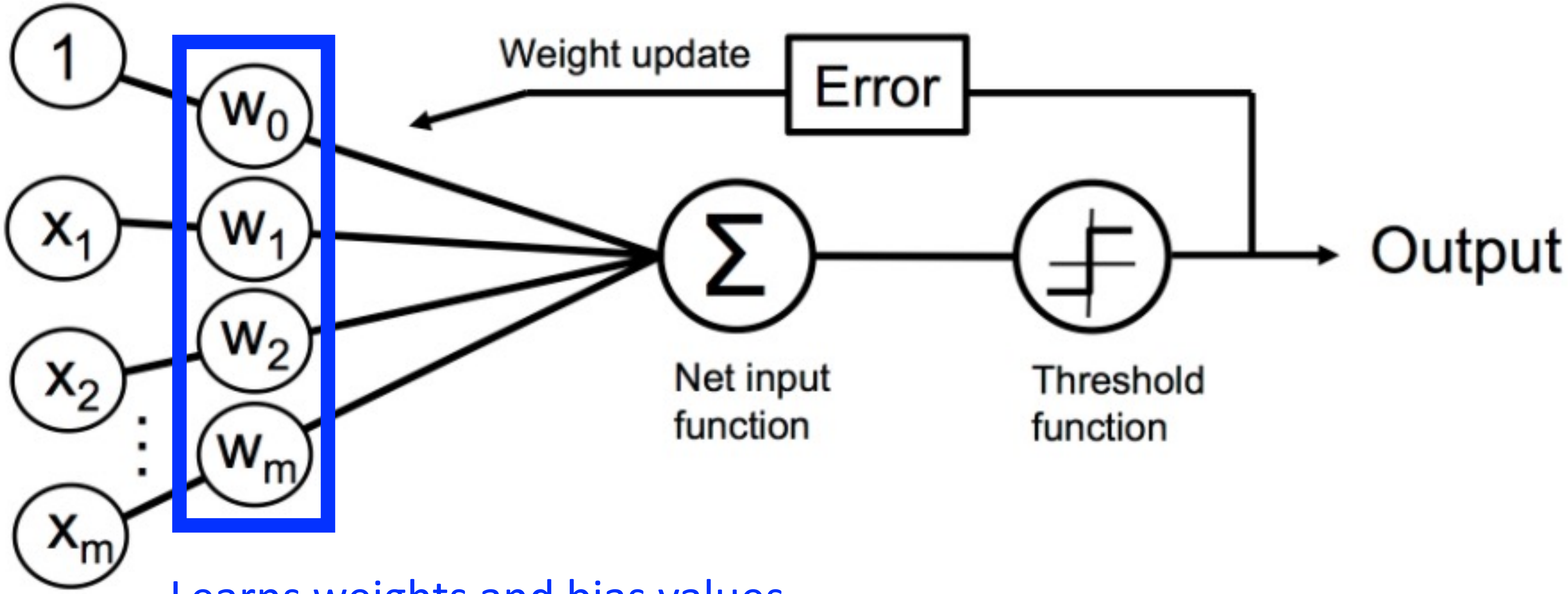


$$z = \boxed{w_0}\boxed{x_0} + w_1x_1 + \ldots + w_mx_m = \boldsymbol{w}^T\boldsymbol{x}$$

Bias $\boxed{-\theta}$ $\boxed{1}$

Uday Kamath, John Liu, and James Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

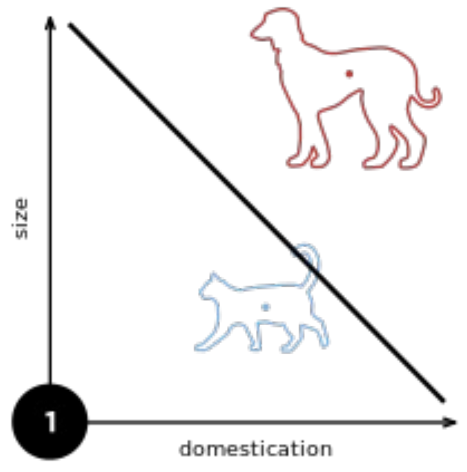
Perceptron: Learning Algorithm



Learns weights and bias values

Perceptron: Learning Algorithm

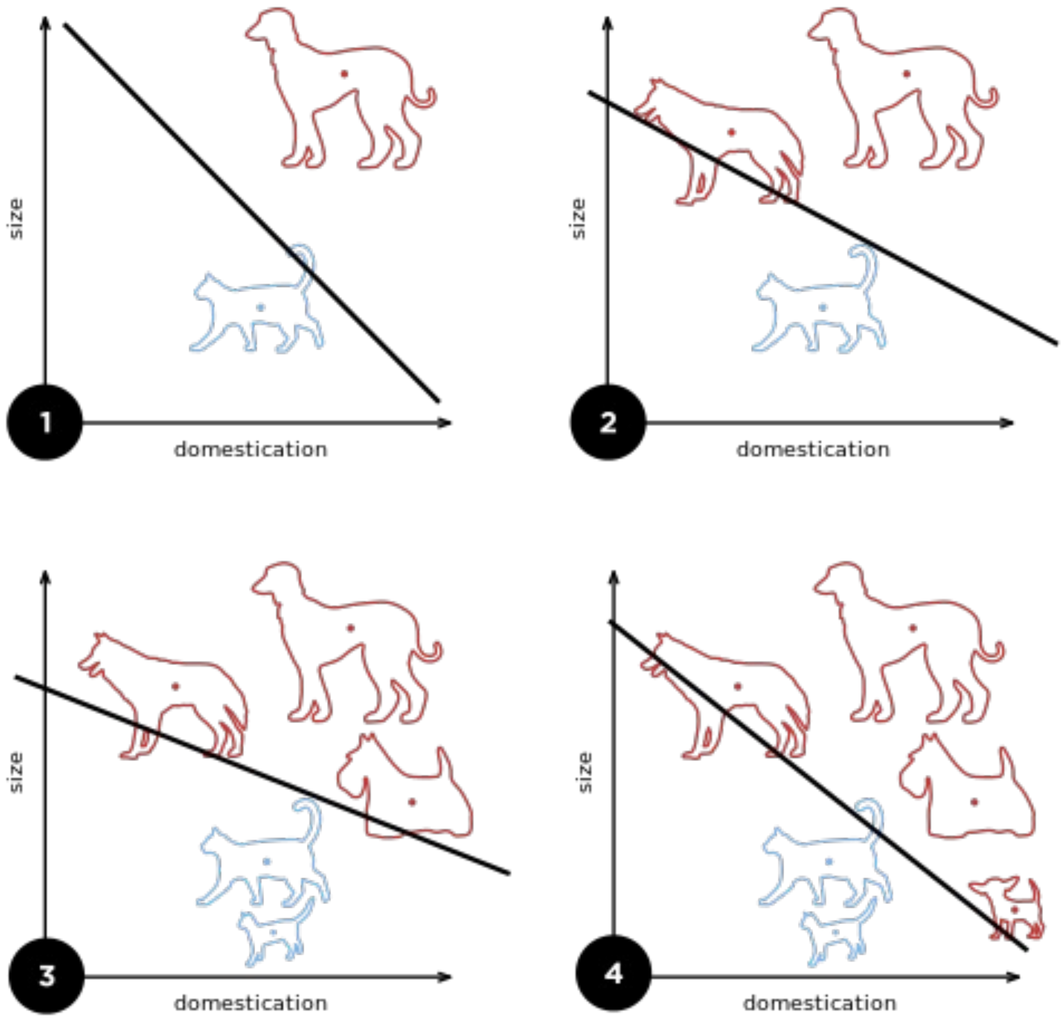
Process: iteratively update boundary with observation of each additional example:



https://en.wikipedia.org/wiki/Perceptron

Perceptron: Learning Algorithm

Process: iteratively update boundary with observation of each additional example:



https://en.wikipedia.org/wiki/Perceptron

Perceptron: Learning Algorithm

1. Initialize weights/bias to 0 or small random numbers
2. For each training sample (i.e., *i*) :
    1. Compute predicted value (i.e., {-1, 1}): $\sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}$
    2. Update parameters based on prediction success: $w_j := w_j + \boxed{\Delta w_j}$

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

Learning Rate

(set a priori and held constant)

True Class Label

Predicted Class Label

https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html

Perceptron: Learning Algorithm

1. Initialize weights/bias to 0 or small random numbers
2. For each training sample (i.e., *i*) :
    1. Compute predicted value (i.e., {-1, 1}): $\sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}$
    2. Update parameters based on prediction success: $w_j := w_j + \boxed{\Delta w_j}$

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

What happens to the weights when the model predicts the **correct** class label?

- no weight update since result is 0

https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html

Perceptron: Learning Algorithm

1. Initialize weights/bias to 0 or small random numbers
2. For each training sample (i.e., *i*) :
    1. Compute predicted value (i.e., {-1, 1}): $\sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}$
    2. Update parameters based on prediction success: $w_j := w_j + \boxed{\Delta w_j}$

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

What happens to the weights when the model predicts the **wrong** class label?

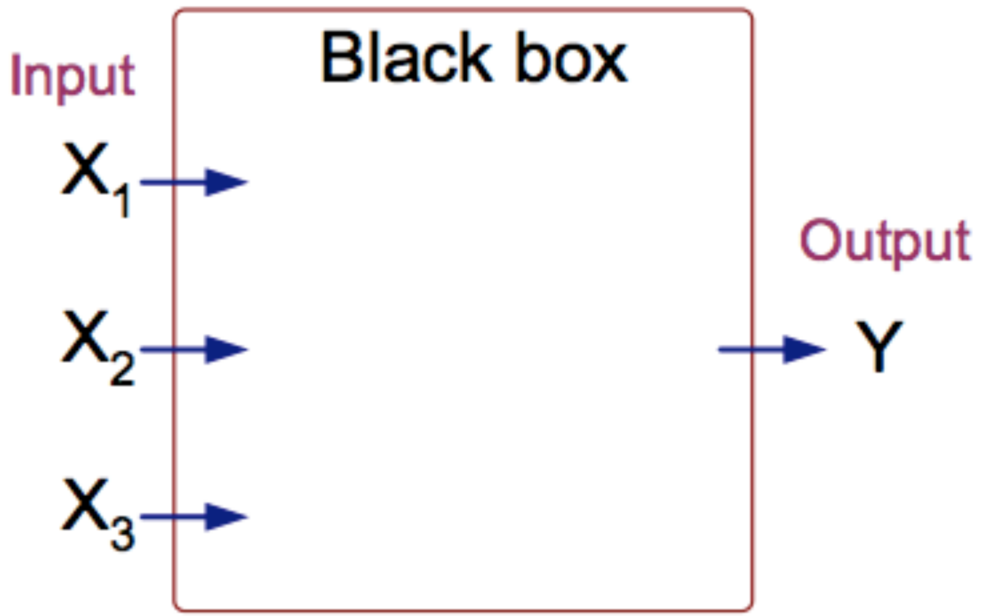
- weights change since result is “2” or “-2”

https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html

Perceptron: Example

- True Model: Y is 1 if at least two of the three inputs are equal to 1.

X_1	X_2	X_3	Y
1	0	0	
1	0	1	
1	1	0	
1	1	1	?
0	0	1	
0	1	0	
0	1	1	
0	0	0	

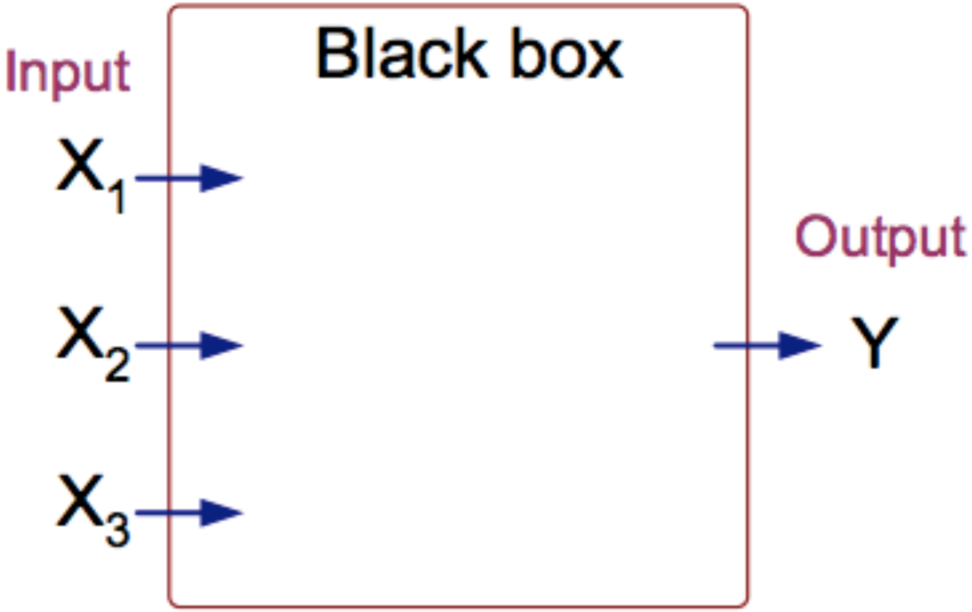


https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example

- True Model: Y is 1 if at least two of the three inputs are equal to 1.

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	
1	1	0	
1	1	1	
0	0	1	?
0	1	0	
0	1	1	
0	0	0	

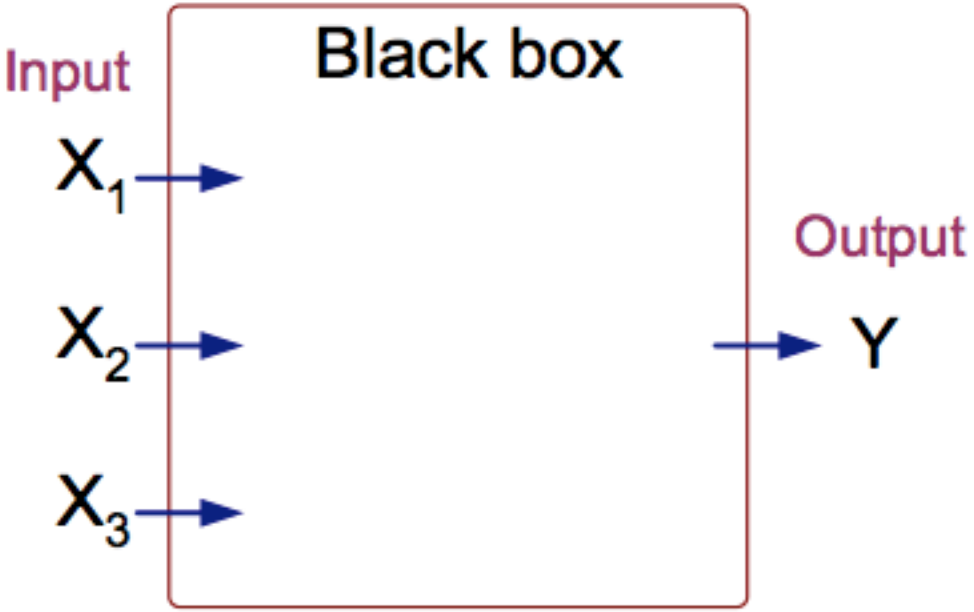


https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example

- True Model: Y is 1 if at least two of the three inputs are equal to 1.

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	
1	1	1	
0	0	1	?
0	1	0	
0	1	1	
0	0	0	

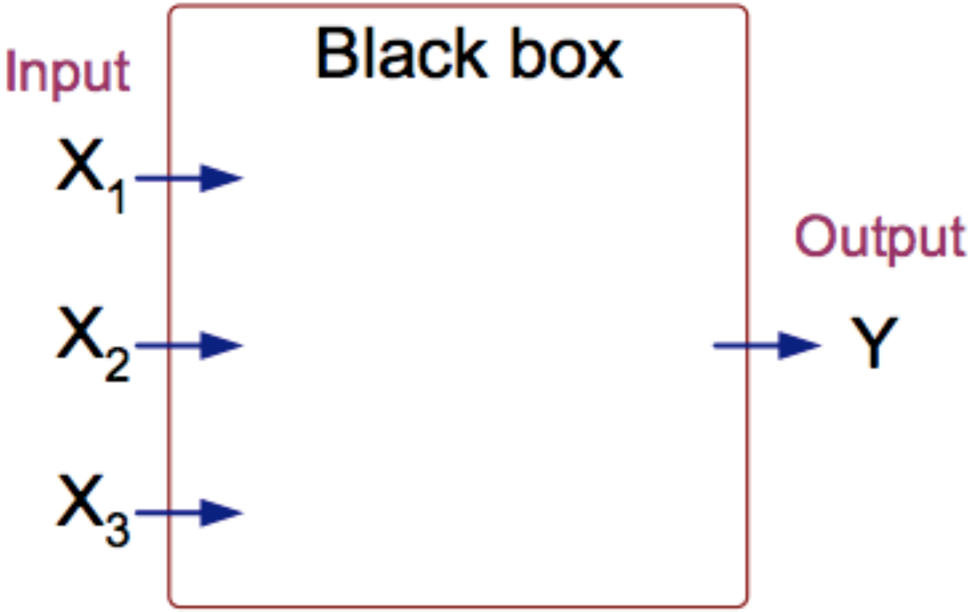


https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example

- True Model: Y is 1 if at least two of the three inputs are equal to 1.

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example (Training with 1rst Sample)

- Compute predicted value: $\sum_{j=0}^{m} x_j w_j = w^T x$; $\phi(w^T x)$ = $\begin{cases} 1 \text{ if } \phi(w^T x) \geq 0 \\ -1 \text{ otherwise} \end{cases}$

X_1	X_2	X_3	Y
1	0	0	-1

Predicted
?

w_0	w_1	w_2	w_3
0	0	0	0

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example (Training with 1rst Sample)

- Update params: $w_j = w_j + \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_j^{(i)}$; learning rate = 0.1

X_1	X_2	X_3	Y
1	0	0	-1

Predicted
1

w_0	w_1	w_2	w_3
0	0	0	0
?	?	?	?

$$\Delta w_0 = \eta\ (\text{target}^{(i)} - \text{output}^{(i)})$$

$$\Delta w_1 = \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_1^{(i)}$$

$$\Delta w_2 = \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_2^{(i)}$$

$$\Delta w_3 = \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_3^{(i)}$$

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example (Training with 1rst Sample)

- Update params: $w_j = w_j + \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_j^{(i)}$; learning rate = 0.1

X_1	X_2	X_3	Y
1	0	0	-1

Predicted
1

w_0	w_1	w_2	w_3
0	0	0	0
?	?	?	?

Δw_0 = 0.1(-1-1)*1 = -0.2

Δw_1 = 0.1(-1-1)*1 = -0.2

Δw_2 = 0.1(-1-1)*0 = 0

Δw_3 = 0.1(-1-1)*0 = 0

Updates make weights more negative so that the model is more likely to classify the sample as -1 next time

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example (Training with 2nd Sample)

- Compute output value: $\sum_{j=0}^{m} x_j w_j = w^T x$; $\phi(w^T x) = \begin{cases} 1 \text{ if } \phi(w^T x) \geq 0 \\ -1 \text{ otherwise} \end{cases}$

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1

Predicted
1
?

w_0	w_1	w_2	w_3
0	0	0	0
-0.2	-0.2	0	0

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example (Training with 2nd Sample)

- Update params: $w_j = w_j + \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_j^{(i)}$; learning rate = 0.1

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1

Predicted
1
-1

w_0	w_1	w_2	w_3
0	0	0	0
-0.2	-0.2	0	0
?	?	?	?

$$\Delta w_0 = \eta\,(\text{target}^{(i)} - \text{output}^{(i)})$$

$$\Delta w_1 = \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_1^{(i)}$$

$$\Delta w_2 = \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_2^{(i)}$$

$$\Delta w_3 = \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_3^{(i)}$$

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example (Training with 2nd Sample)

- Update params: $w_j = w_j + \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_j^{(i)}$; learning rate = 0.1

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1

Predicted
1
-1

w_0	w_1	w_2	w_3
0	0	0	0
-0.2	-0.2	0	0
?	?	?	?

Δw_0 = 0.1(1--1)*1 = 0.2

Δw_1 = 0.1(1--1)*1 = 0.2

Δw_2 = 0.1(1--1)*0 = 0

Δw_3 = 0.1(1--1)*1 = 0.2

Updates make weights more positive so that the model is more likely to classify the sample as 1 next time

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example (Training with 2nd Sample)

- Update params: $w_j = w_j + \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_j^{(i)}$; learning rate = 0.1

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1

Predicted
1
-1

w_0	w_1	w_2	w_3
0	0	0	0
-0.2	-0.2	0	0
0	0	0	0.2

Δw_0 = 0.1(1--1)*1 = 0.2

Δw_1 = 0.1(1--1)*1 = 0.2

Δw_2 = 0.1(1--1)*0 = 0

Δw_3 = 0.1(1--1)*1 = 0.2

What is the influence of the learning rate? i.e., what would happen if the value was larger/smaller?

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example – One Epoch (Training with All Samples)

- $w_j = w_j + \eta \, (\text{target}^{(i)} - \text{output}^{(i)}) \, x_j^{(i)}$; learning rate = 0.1

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	-0.2	0	0
2	0	0	0	0.2
3	0	0	0	0.2
4	0	0	0	0.2
5	-0.2	0	0	0
6	-0.2	0	0	0
7	0	0	0.2	0.2
8	-0.2	0	0.2	0.2

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example – Six Epochs

- $w_j = w_j + \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_j^{(i)}$; learning rate = 0.1

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

Epoch	w_0	w_1	w_2	w_3
0	0	0	0	0

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Example – Six Epochs

- $w_j = w_j + \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\, x_j^{(i)}$; learning rate = 0.1

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	-0.2	0	0
2	0	0	0	0.2
3	0	0	0	0.2
4	0	0	0	0.2
5	-0.2	0	0	0
6	-0.2	0	0	0
7	0	0	0.2	0.2
8	-0.2	0	0.2	0.2

Epoch	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	0	0.2	0.2
2	-0.2	0	0.4	0.2
3	-0.4	0	0.4	0.2
4	-0.4	0.2	0.4	0.4
5	-0.6	0.2	0.4	0.2
6	-0.6	0.4	0.4	0.2

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Learning Algorithm Choices

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

Today’s Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neuron: Perceptron


The End

Credits

- Image of Boulder: http://boulderrunning.com/where2run/five-trails-for-hill-running-and-mountain-training/
- Stick person figure: https://drawception.com/game/AsPNcppPND/draw-yourself-blindfolded-pio/
- Figure: https://www.quora.com/What-is-meant-by-gradient-descent-in-laymen-terms
- Figure and great reference: https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html