Artificial Neurons and Gradient Descent

Danna Gurari

University of Texas at Austin

Spring 2020

Review

- Last week:
  - Regression applications
  - Evaluating regression models
  - Background: notation
  - Linear regression
  - Polynomial regression
  - Regularization (Ridge regression and Lasso regression)
- Assignments (Canvas):
  - Problem set 2 due yesterday
  - Lab assignment 1 due next week
- Questions?

Today’s Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neurons: Perceptron & Adaline
- Gradient descent
- Lab

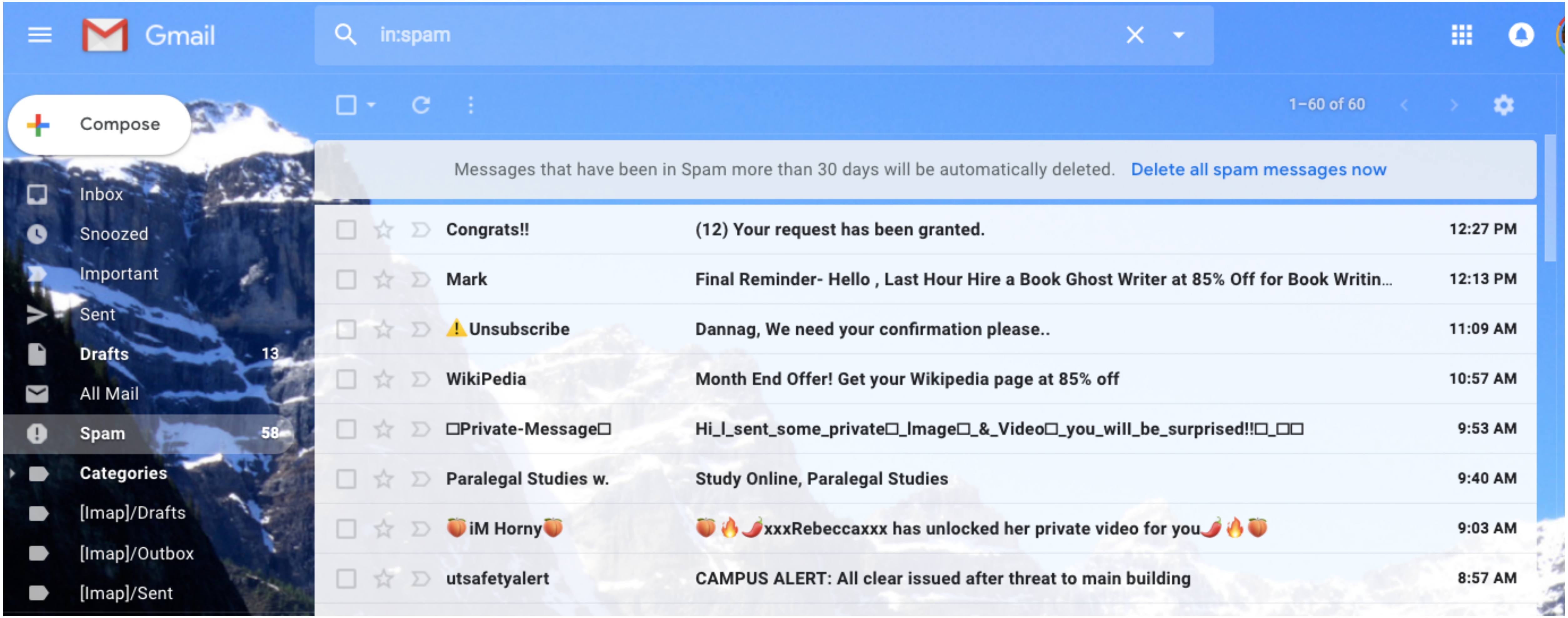
Today’s Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neurons: Perceptron & Adaline
- Gradient descent
- Lab

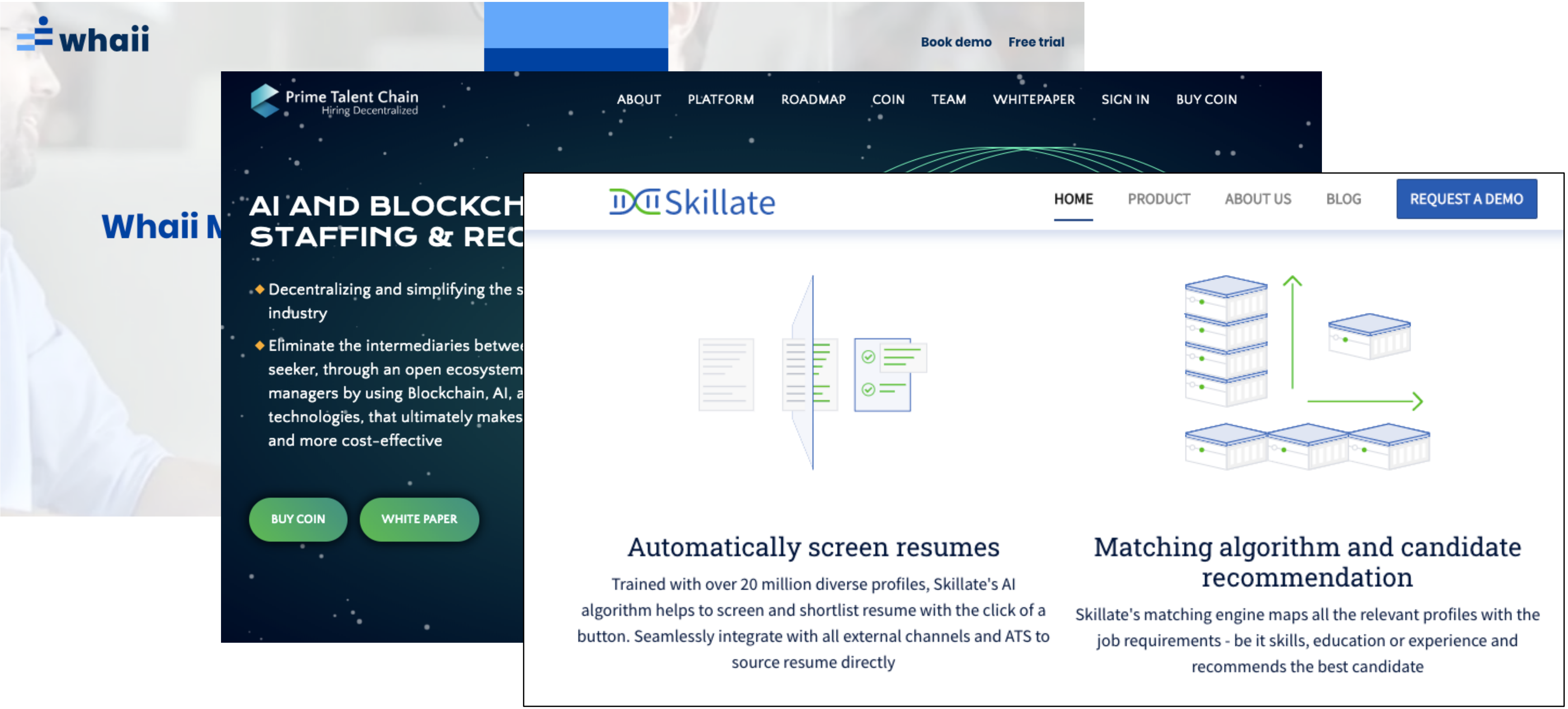
Today's Focus: Binary Classification

Distinguish 2 classes

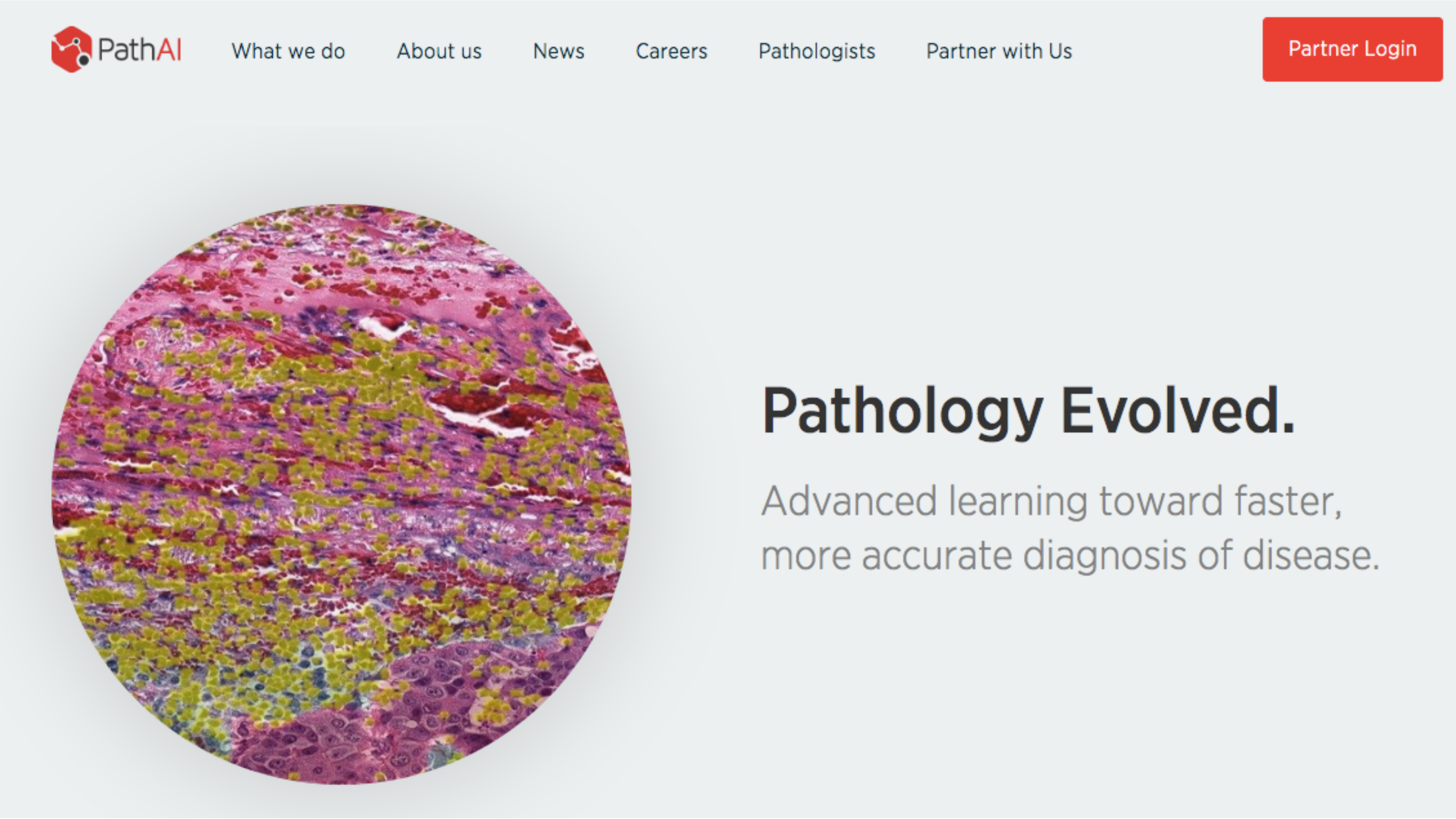
Binary Classification: Spam Detection



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: Food Quality Control

Machine Learning: Using Algorithms to Sort Fruit

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

Today’s Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neurons: Perceptron & Adaline
- Gradient descent
- Lab

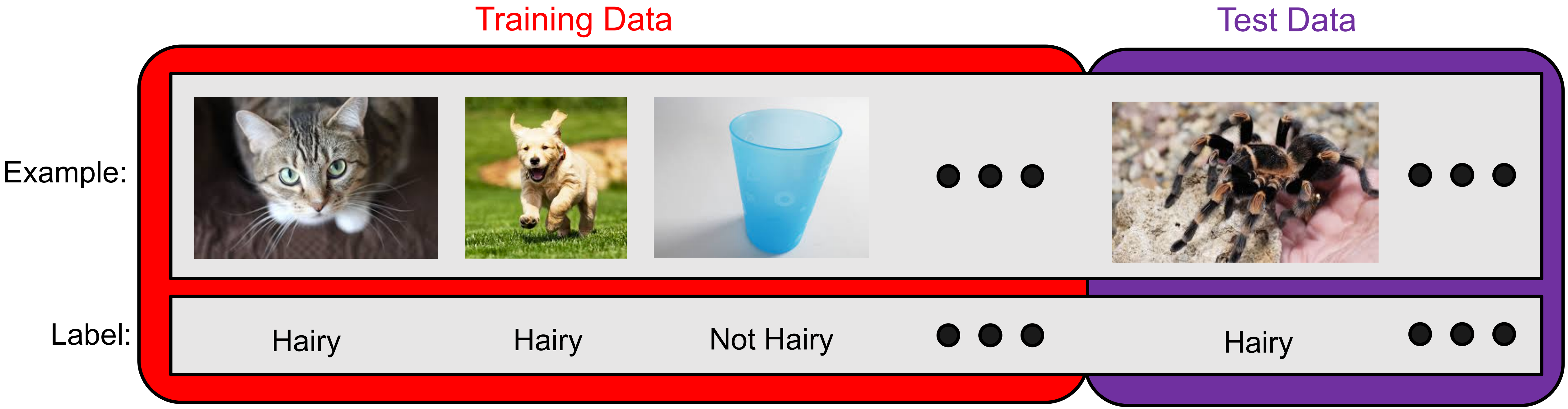
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

Example:    • • •  • • •

Label: Hairy | Hairy | Not Hairy | • • • | Hairy | • • •

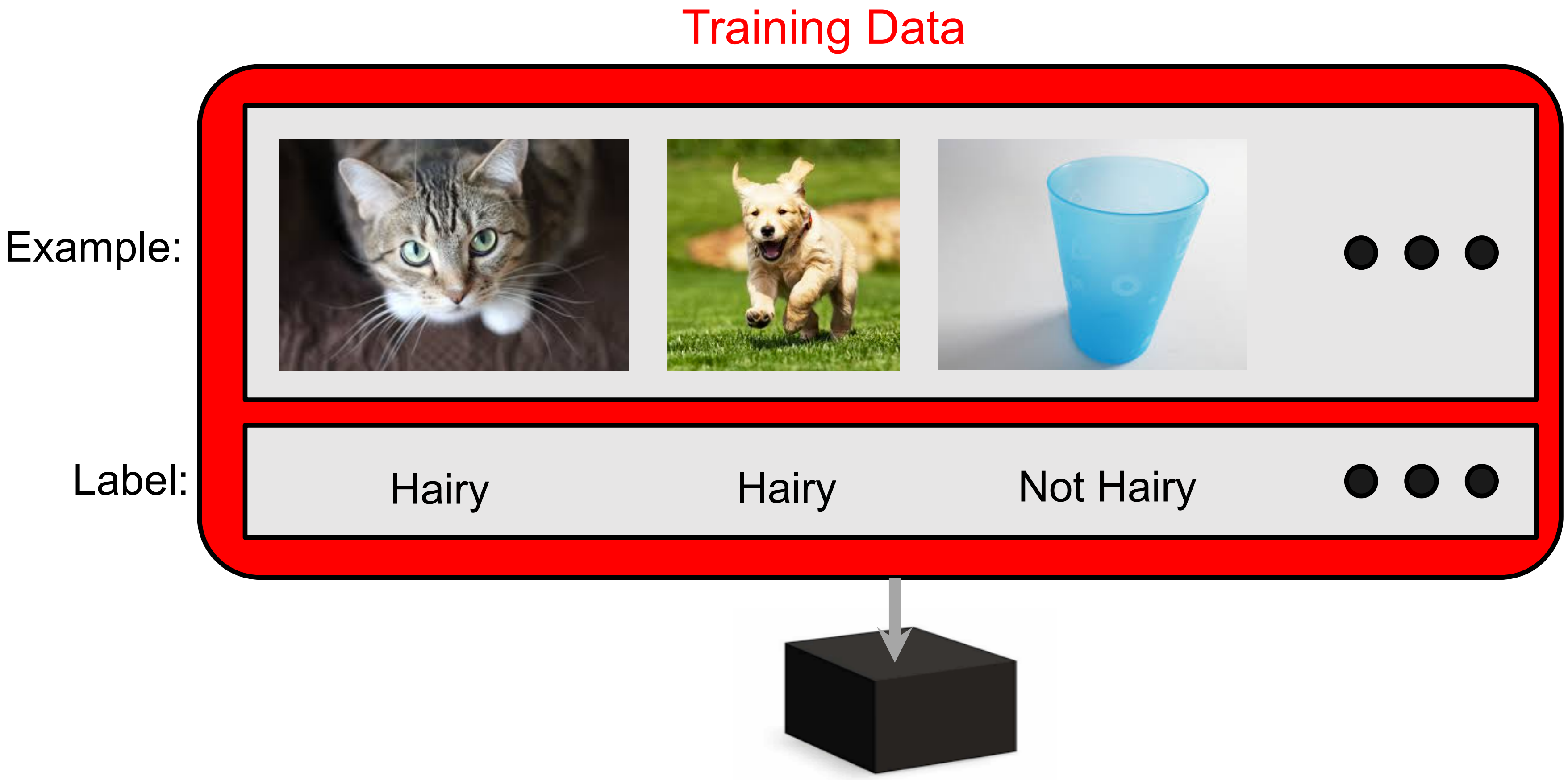
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

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



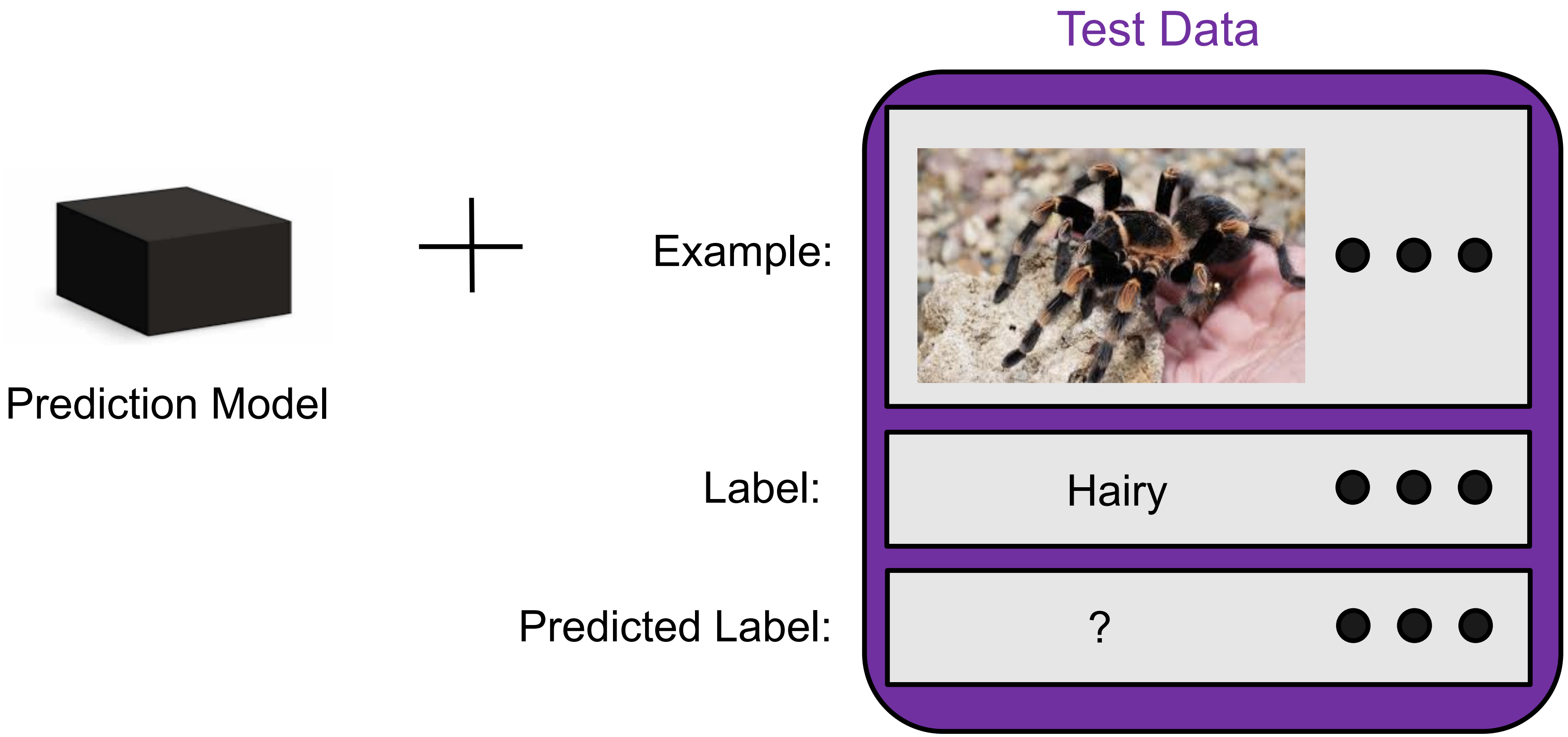
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

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



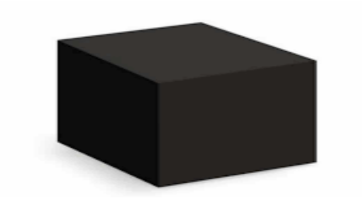
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

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

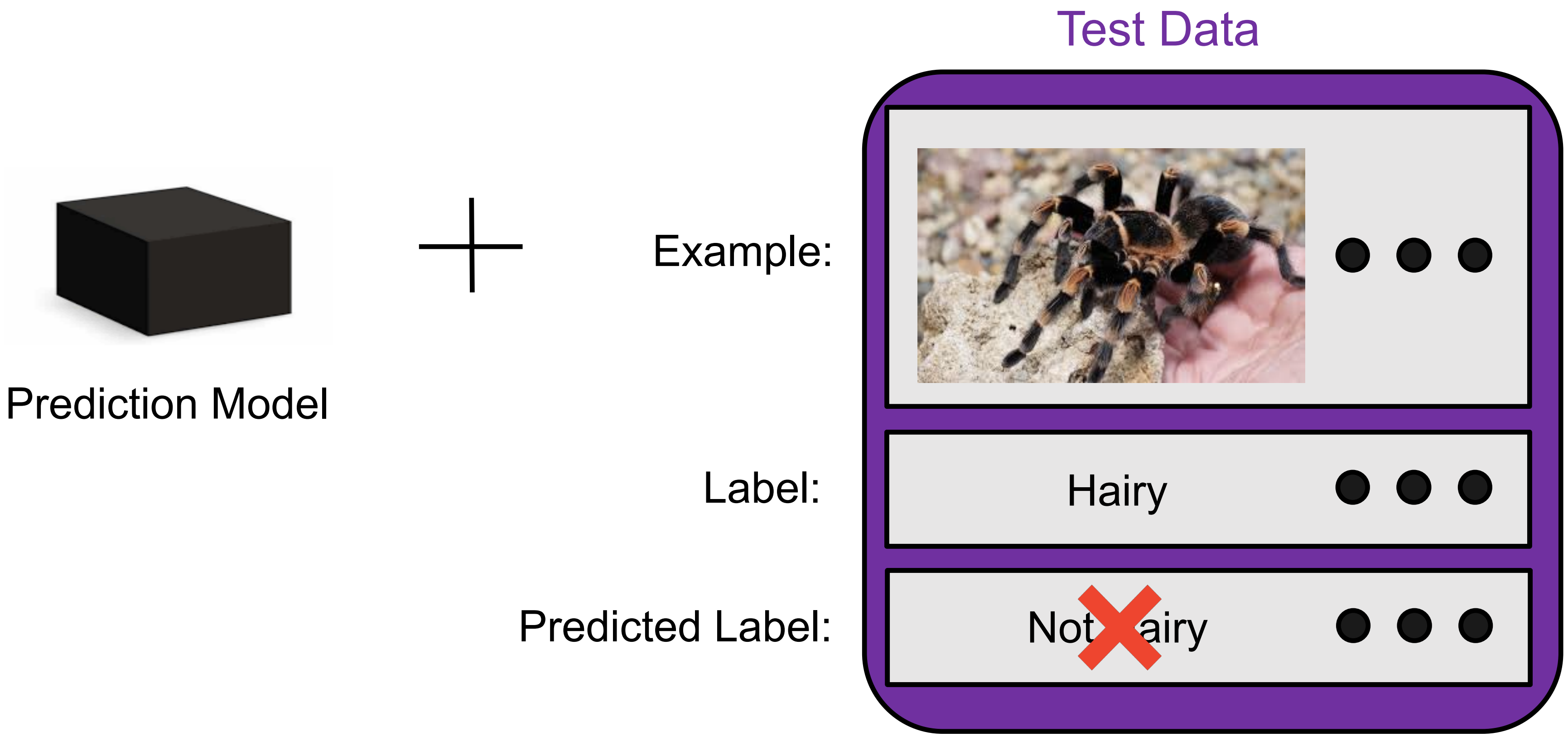
Example:  • • •

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



Evaluation Methods: Confusion Matrix

		Actual	
		Spam	Trusted
Predicted	Spam	TP	FP
	Trusted	FN	TN

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

Evaluation Methods : Descriptive Statistics

e.g.,

		Actual	
		Spam	Trusted
Predicted	Spam	50	10
	Trusted	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 (number 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.

Each student should submit a response in a Google Form (tracks attendance)

Today’s Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neurons: Perceptron & Adaline
- Gradient descent
- Lab

Inspiration: Animal's Computing Machinery

Neuron

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

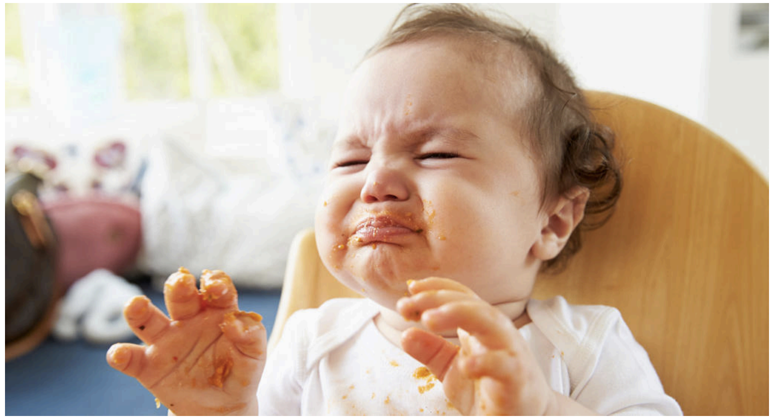
"hot"

https://www.clipart.email/clipart/dont-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/

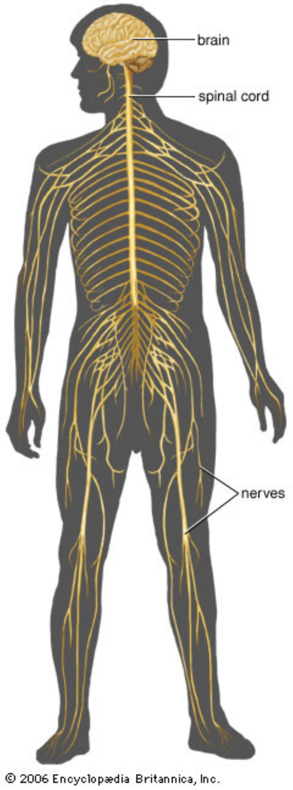
"spicy"



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

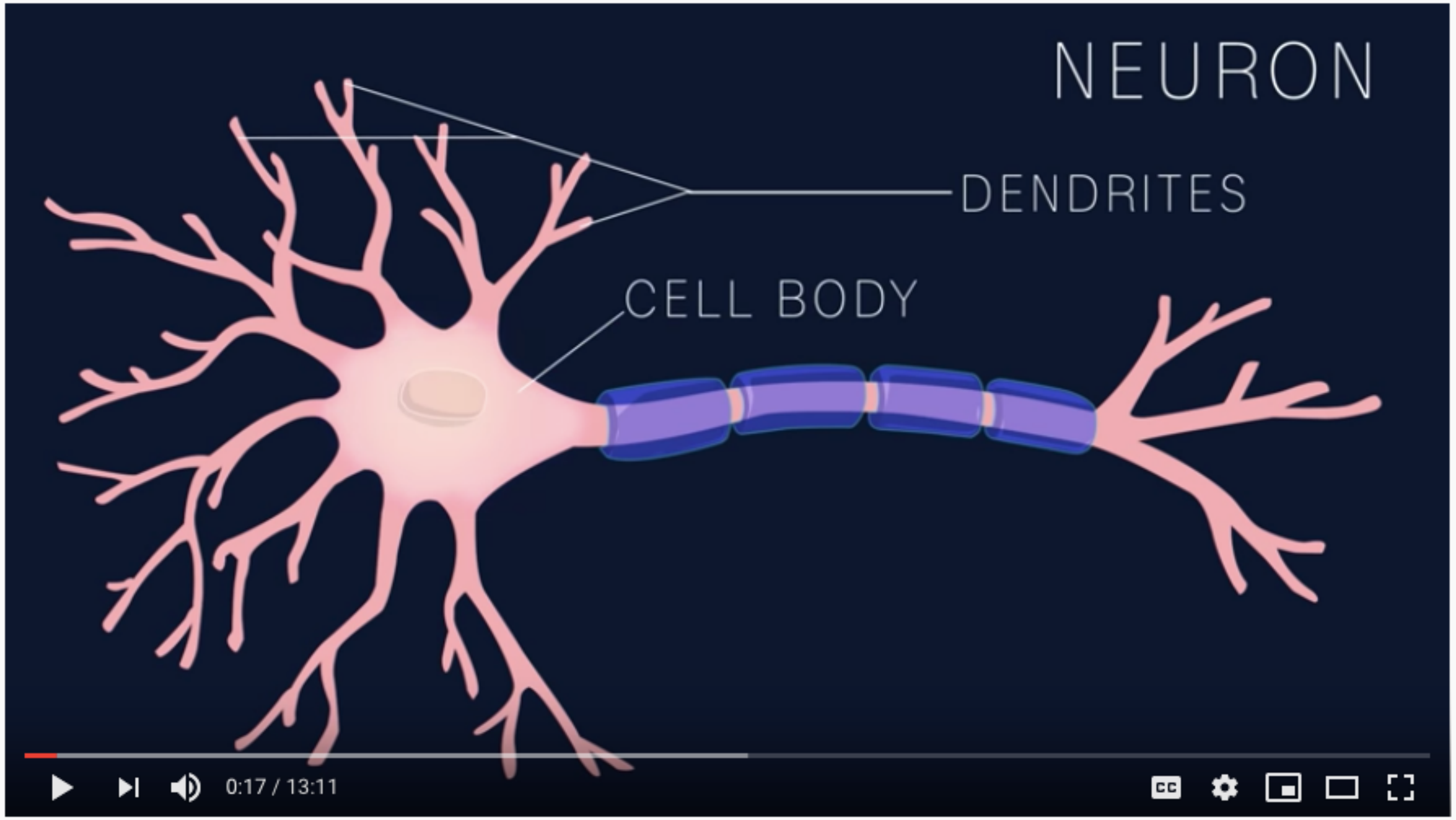


https://www.britannica.com/science/human-nervous-system

Nematode worm: 302 neurons

Human: ~100,000,000,000 neurons

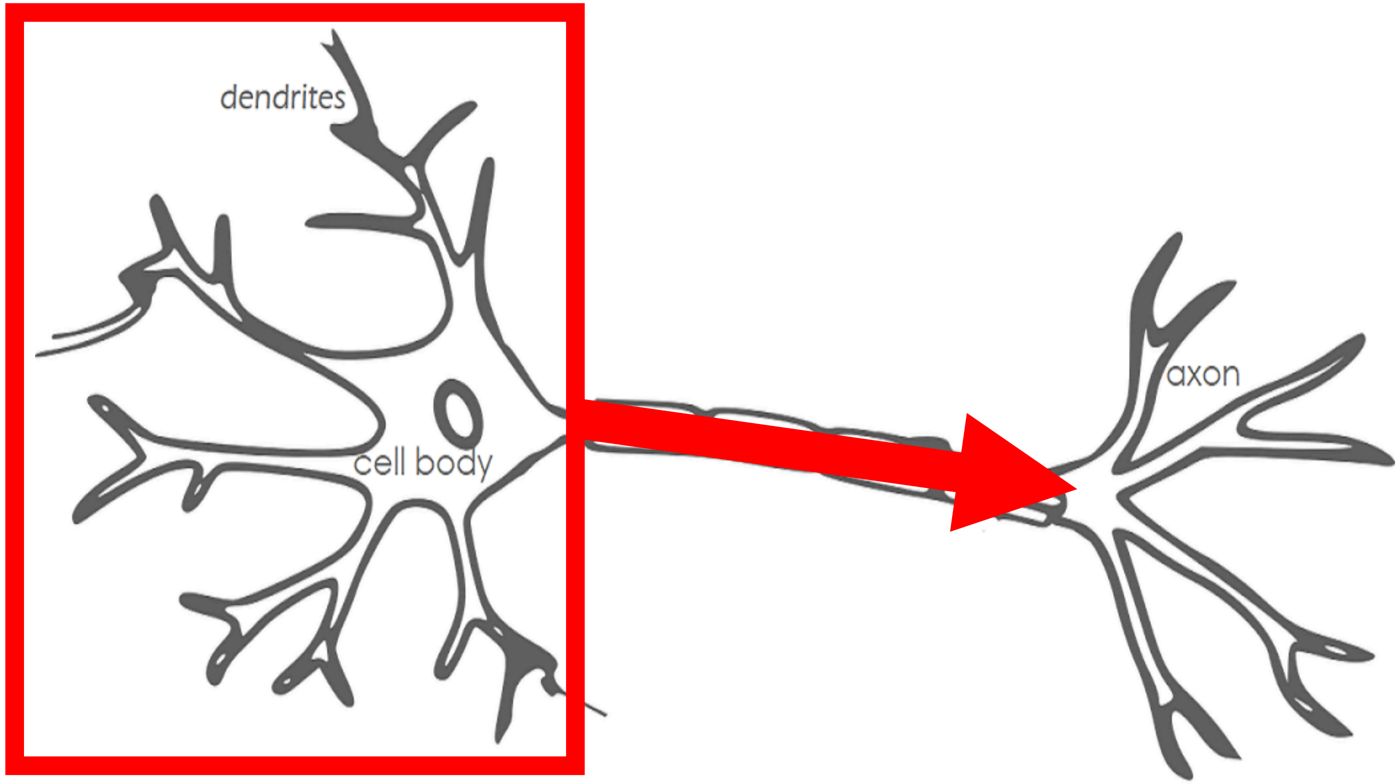
Inspiration: Animal's Computing Machinery



Action Potential in the Neuron

Demo (0-1:20): https://www.youtube.com/watch?v=oa6rvUJlg7o

Inspiration: Neuron “Firing”



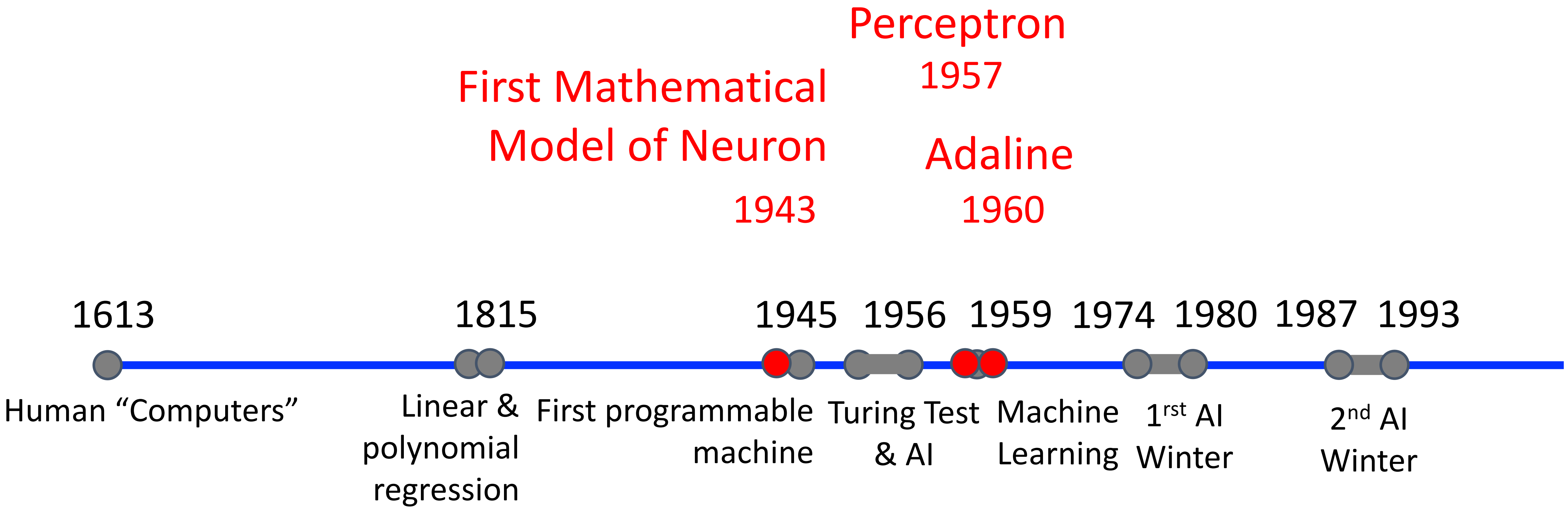
- When the input signals exceed a certain threshold within a short period of time, a neuron “fires”
- Neuron “firing” (outputs signal) is an “all-or-none” process

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

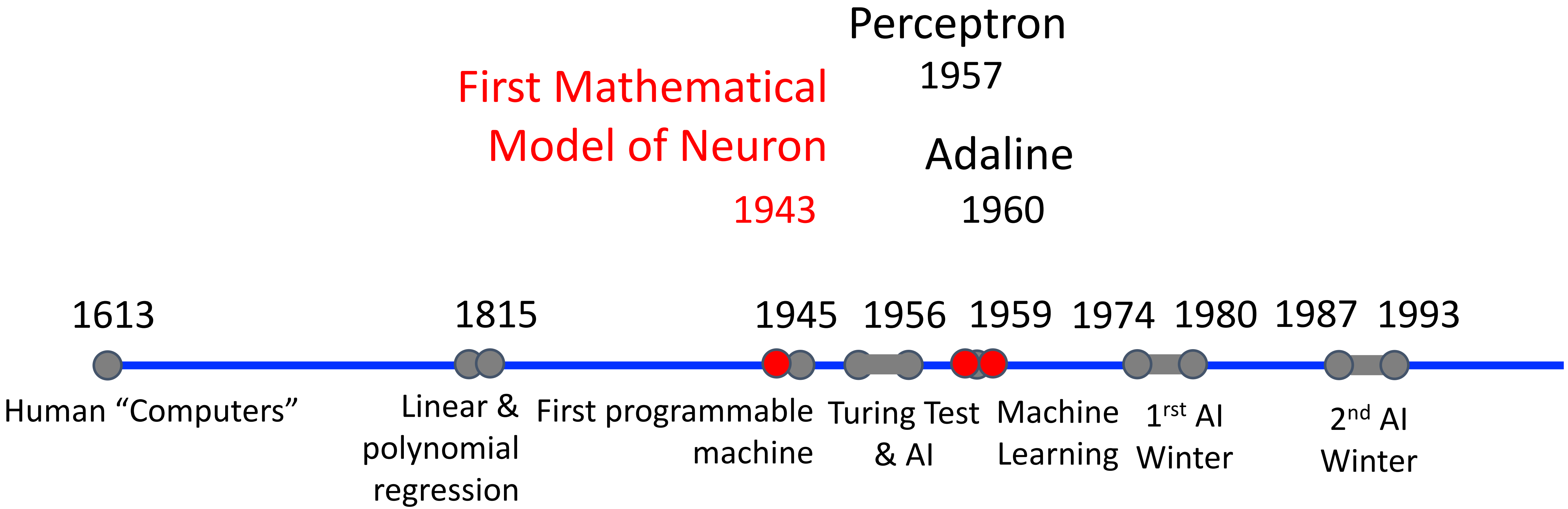
Today’s Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neurons: Perceptron & Adaline
- Gradient descent
- Lab

Artificial Neurons: Historical Context



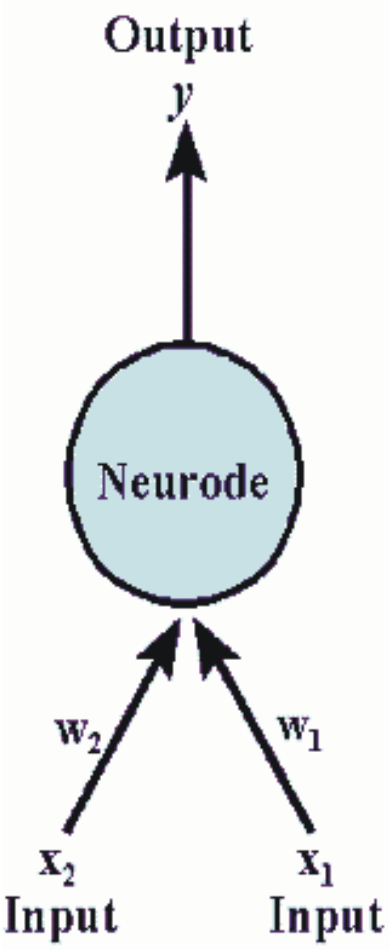
Artificial Neurons: Historical Context



Artificial Neuron: McCulloch-Pitts Neuron

Warren McCulloch
(Neurophysiologist)

Walter Pitts
(Mathematician)



Note:

- weights (W) and threshold (T) values are fixed
- inputs and weights can be only 0 or 1
- fires when combined input exceeds threshold

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

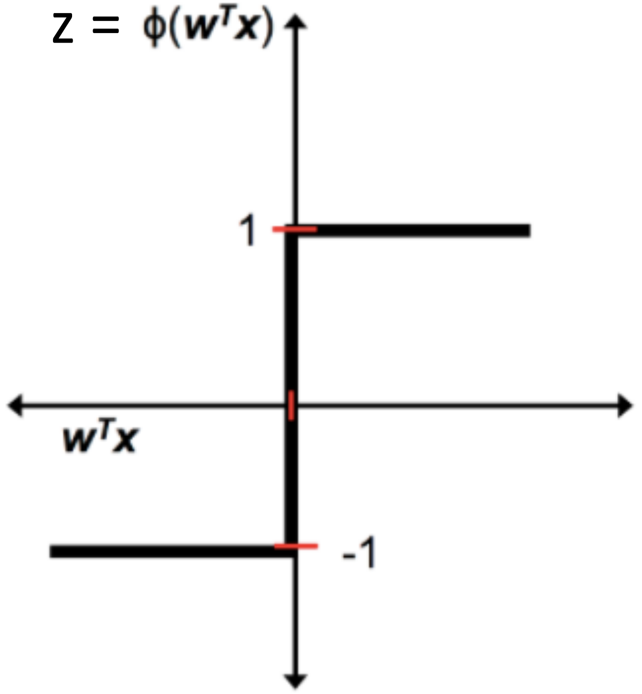
http://web.csulb.edu/~cwallis/artificialn/warren_mcculloch.html

Figure Source: https://web.csulb.edu/~cwallis/artificialn/History.htm

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

Artificial Neuron: McCulloch-Pitts Neuron

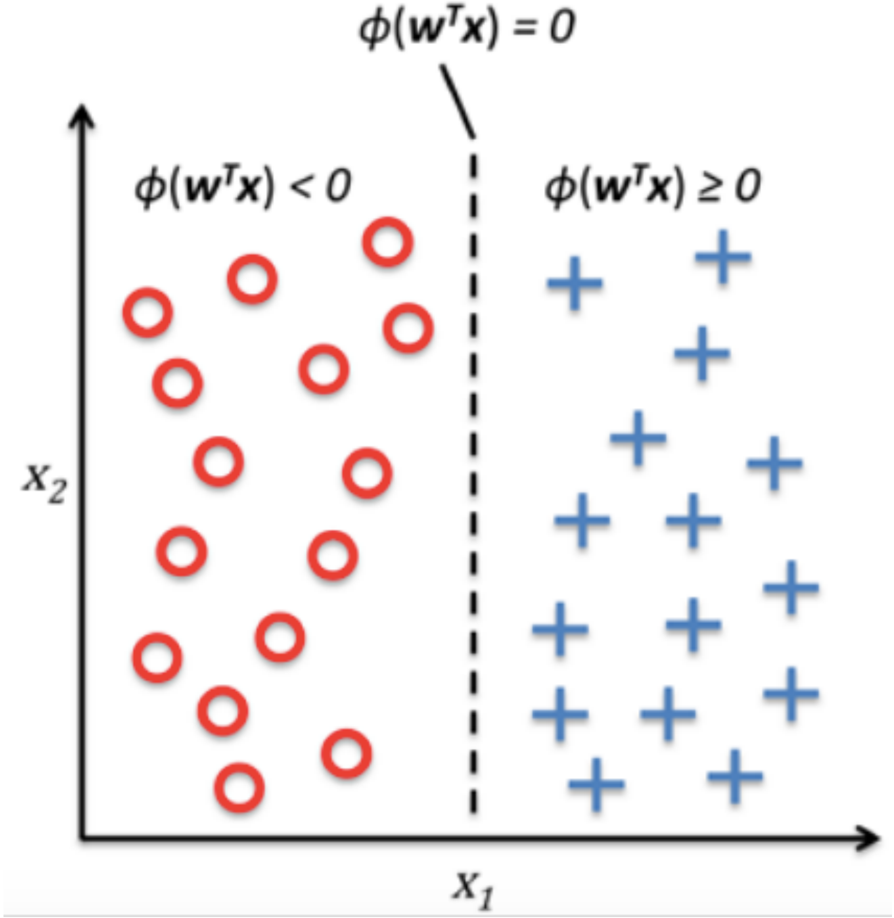
- Mathematical definition: $z = \begin{cases} 1 \text{ if } z \geq \theta \\ -1 \text{ otherwise} \end{cases}$
  - “fire” or “do not fire”
  - mimics human brain



Python Machine Learning; Raschkka & Mirjalili

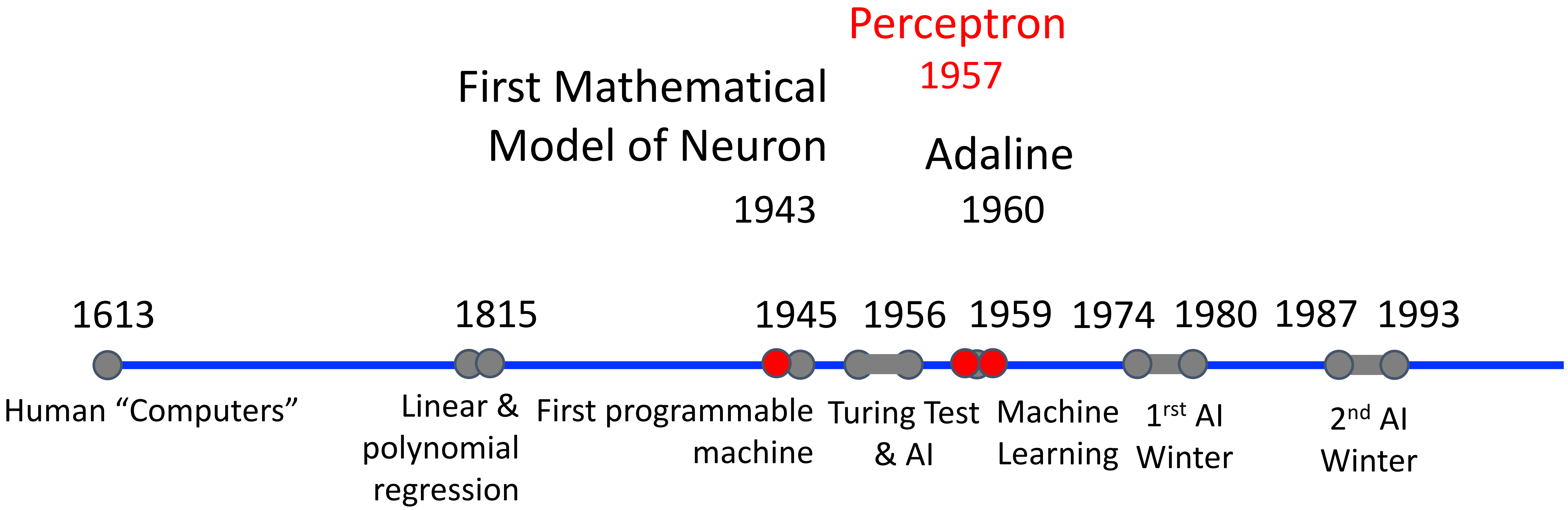
Artificial Neuron: McCulloch-Pitts Neuron

- Mathematical definition: $z = \begin{cases} 1 \text{ if } z \geq \theta \\ -1 \text{ otherwise} \end{cases}$



Python Machine Learning; Raschkka & Mirjalili

Artificial Neurons: Historical Context



Perceptron: Innovator and Vision

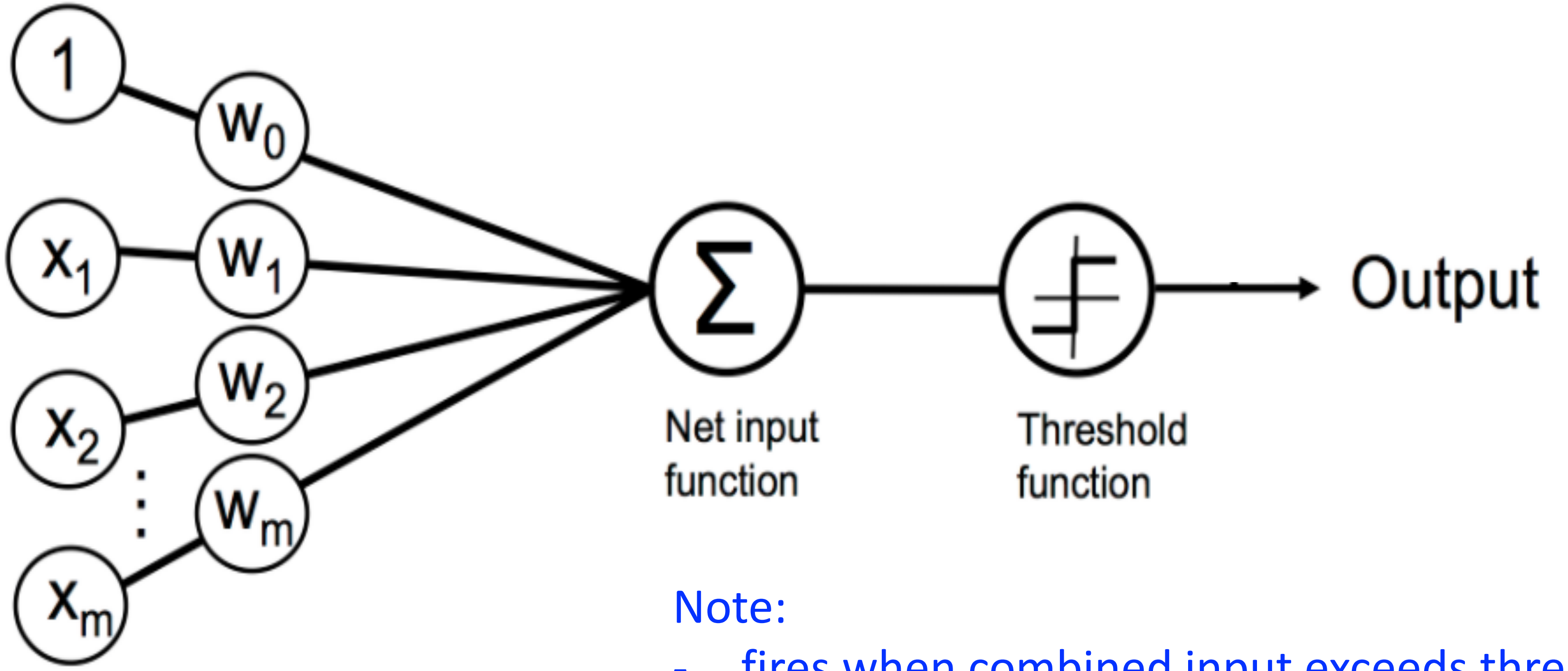
Frank Rosenblatt
(Psychologist)

"[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: Model (Linear Threshold Unit)



Note:

- fires when combined input exceeds threshold
- 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: Model (Linear Threshold Unit)

- Fires when a function exceeds threshold:

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

- Rewriting model:

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

- Where:

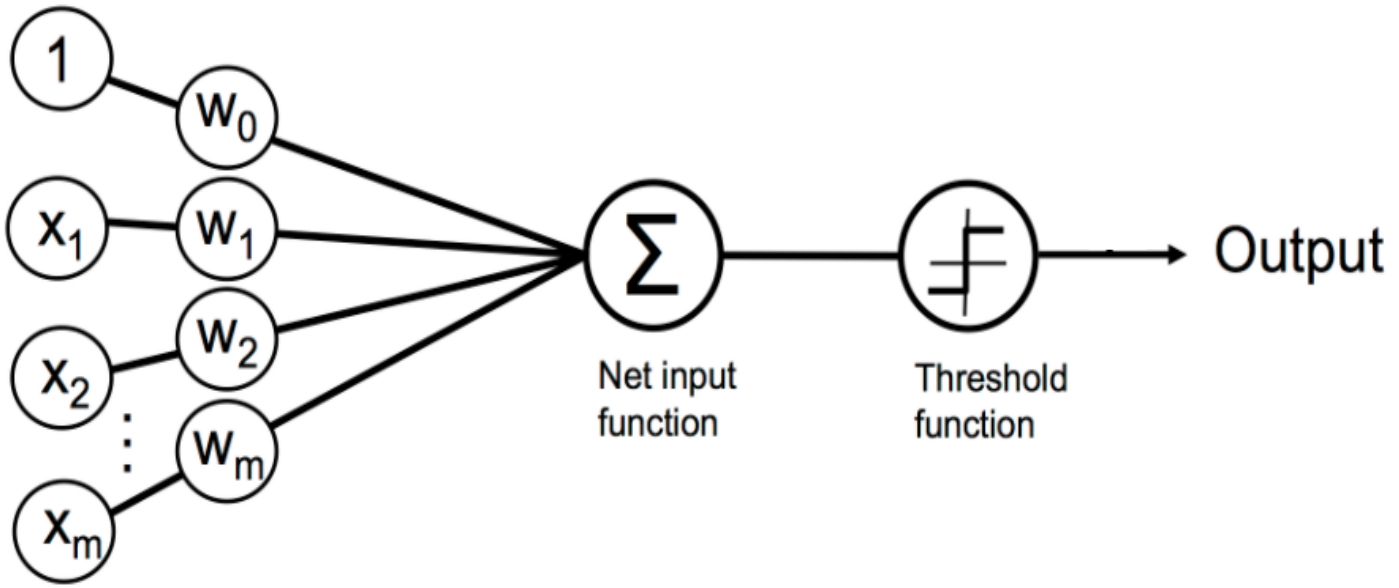
$$z = \underbrace{\boxed{w_0}}_{\boxed{-\theta}}\underbrace{\boxed{x_0}}_{\boxed{1}} + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

Bias

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)



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

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

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

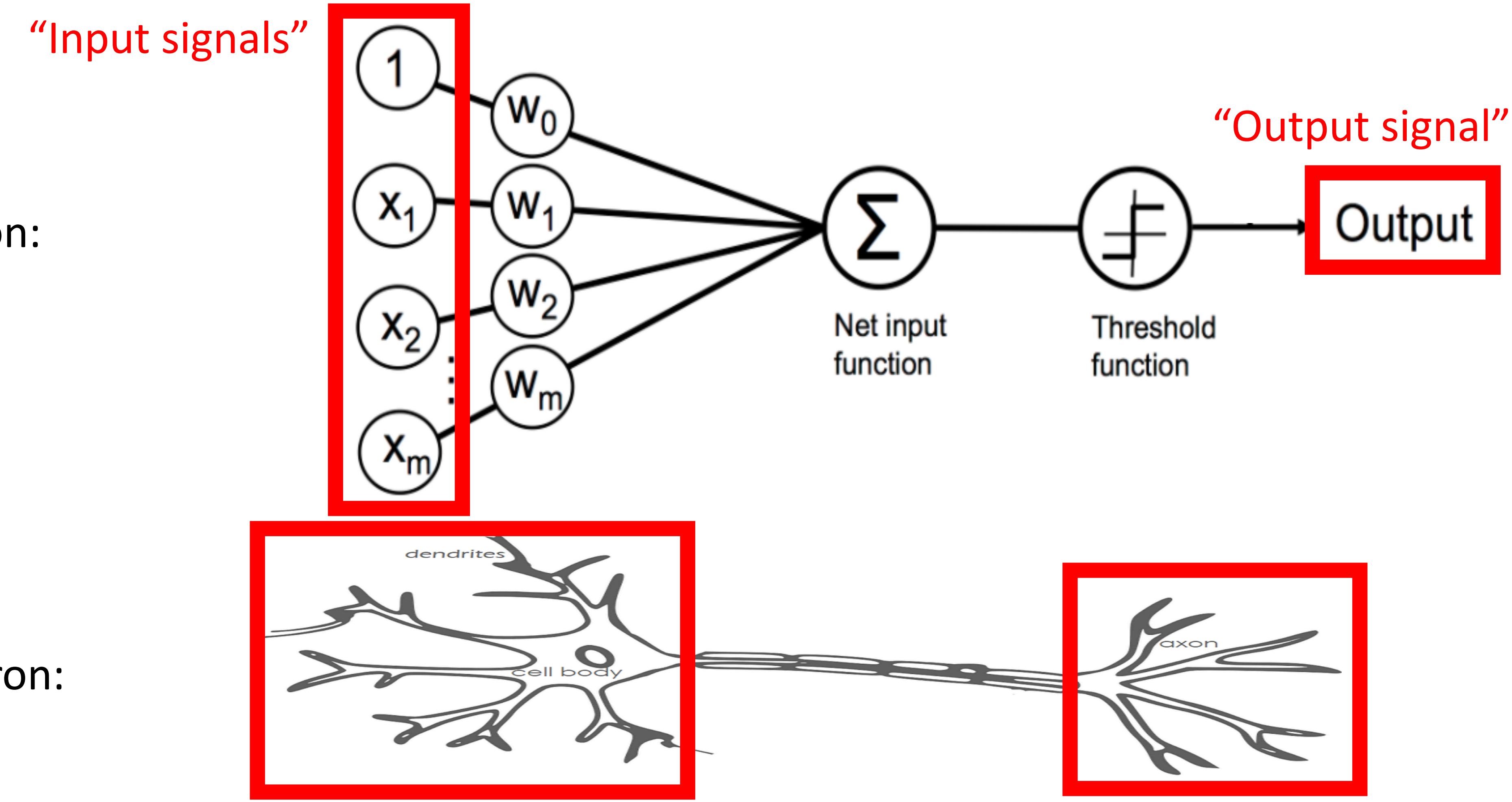
Python Machine Learning; Raschka & Mirjalili

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

Perceptron: Model (Linear Threshold Unit)

Artificial Neuron:

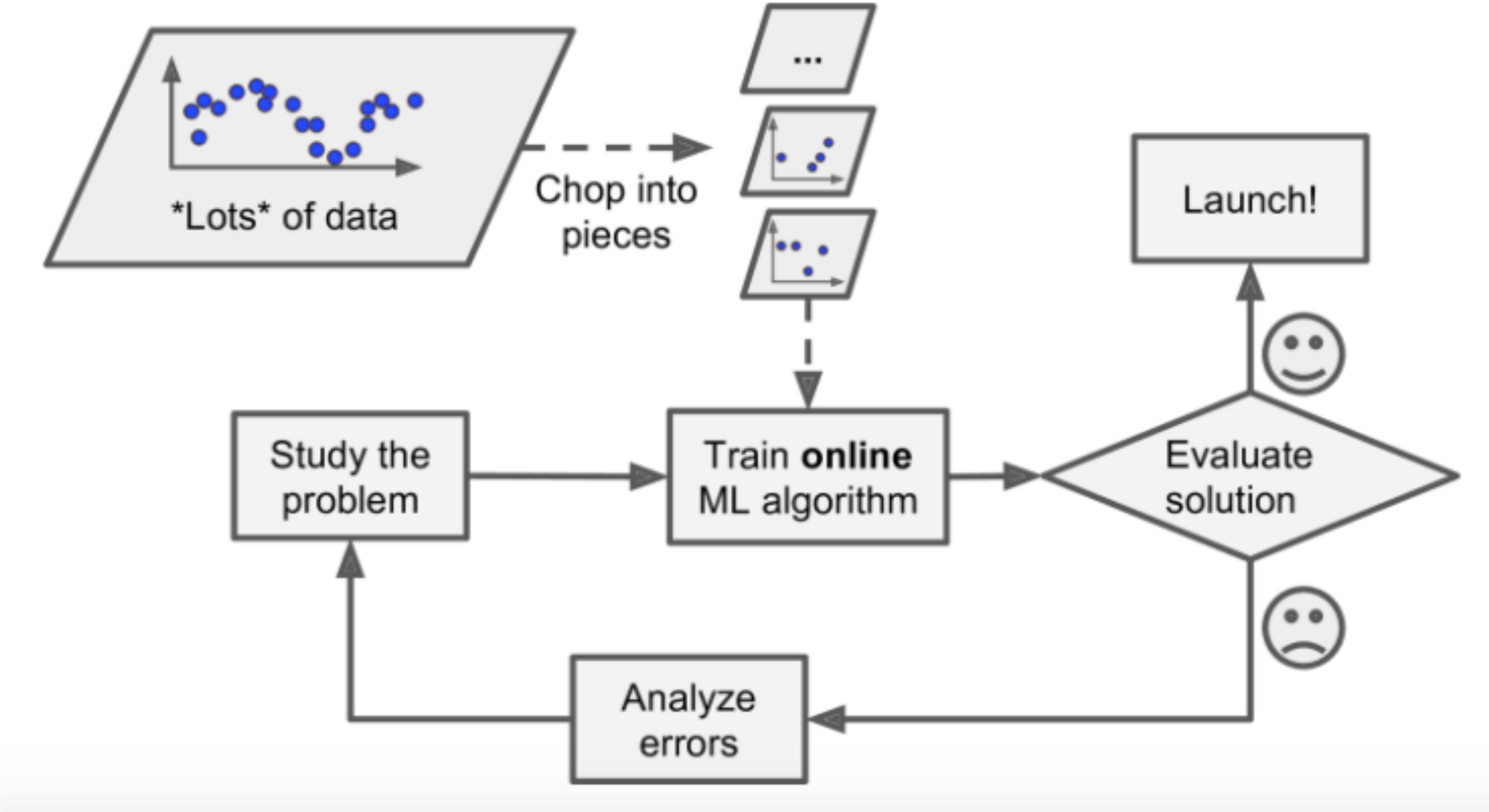
Biological Neuron:



Python Machine Learning; Raschka & Mirjalili

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

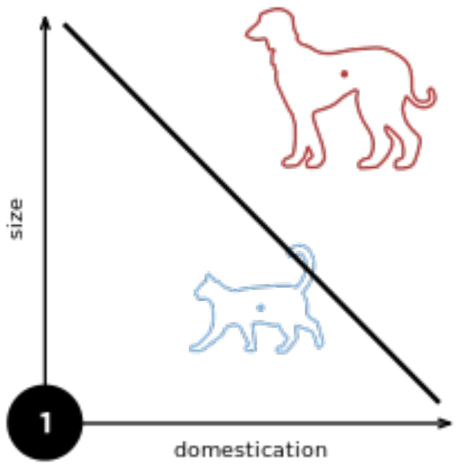
Perceptron: Learning Algorithm Approach



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

Perceptron: Learning Algorithm Approach

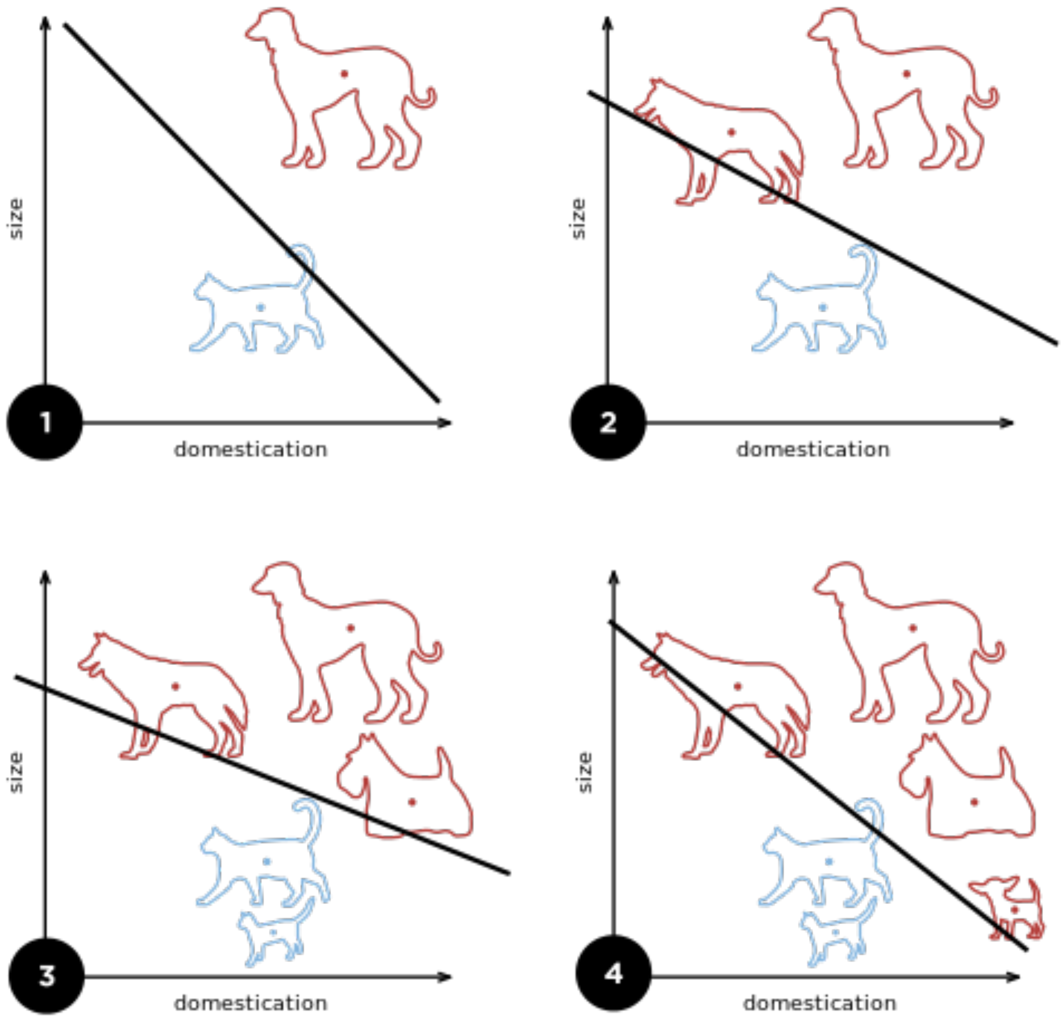
Iteratively update linear boundary with observation of each additional example:



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

Perceptron: Learning Algorithm Approach

Iteratively update linear boundary with observation of each additional example:



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

Perceptron: Learning Algorithm

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

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

Learning Rate

True Class Label

Predicted Class Label

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

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

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

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

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: $\sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}$
    2. Update weights with the following definition: $w_j := w_j + \boxed{\Delta w_j}$

equals 2 or -2 so moves weights closer to positive or negative target class

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

Learning Rate

True Class Label **Predicted Class Label**

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

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

e.g., $y^{(i)} = +1, \ \hat{y}_j^{(i)} = -1, \ \eta = 1$

If: $x_j^{(i)} = 0.5$ Then, $\Delta w_j^{(i)} =$???

$w_j := w_j + \boxed{\Delta w_j}$

equals 2 or -2 so moves weights closer to positive or negative target class

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

Learning Rate

True Class Label

Predicted Class Label

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

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

e.g., $y^{(i)} = +1,\ \hat{y}_j^{(i)} = -1,\ \eta = 1$

If: $x_j^{(i)} = 0.5$ Then, $\Delta w_j^{(i)} = (1 - -1)0.5 = (2)0.5 = 1$

- Increases weight so activation will be more positive for the sample next time
- Thus more likely to classify the sample as +1 next time

$$w_j := w_j + \boxed{\Delta w_j}$$

equals 2 or -2 so moves weights closer to positive or negative target class

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

Learning Rate

True Class Label **Predicted Class Label**

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

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

e.g., $y^{(i)} = +1,\ \hat{y}_j^{(i)} = -1,\ \eta = 1$

If: $x_j^{(i)} = 2$ Then, $\Delta w_j =$???

$w_j := w_j + \boxed{\Delta w_j}$

equals 2 or -2 so moves weights closer to positive or negative target class

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

Learning Rate

True Class Label **Predicted Class Label**

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

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

e.g., $y^{(i)} = +1, \ \hat{y}_j^{(i)} = -1, \ \eta = 1$

If: $x_j^{(i)} = 2$ Then, $\Delta w_j = (1 - -1)2 = (2)2 = 4$

- Increases weight to a larger extent to be more positive for the sample next time
- Thus more likely to classify the sample as +1 next time

$$w_j := w_j + \boxed{\Delta w_j}$$

equals 2 or -2 so moves weights closer to positive or negative target class

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

Learning Rate

True Class Label **Predicted Class Label**

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

Perceptron: Learning Algorithm (e.g., 2D dataset)

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: $w_j := w_j + \boxed{\Delta w_j}$

$$\boxed{\begin{aligned} \Delta w_0 &= \eta\ (\text{target}^{(i)} - \text{output}^{(i)}) \\ \Delta w_1 &= \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_1^{(i)} \\ \Delta w_2 &= \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_2^{(i)} \end{aligned}}$$

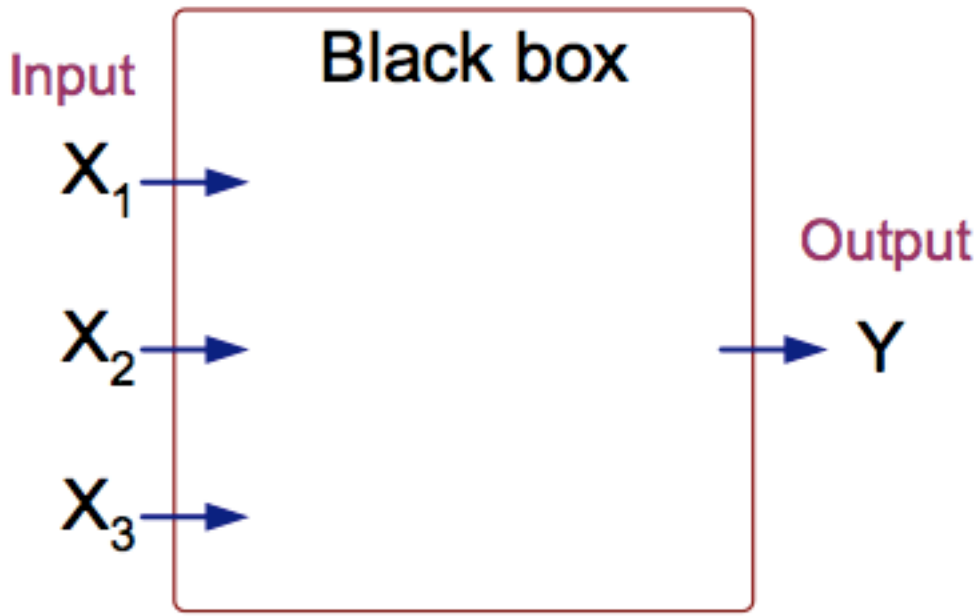
All weights updated simultaneously

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

Perceptron: Learning Algorithm 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	

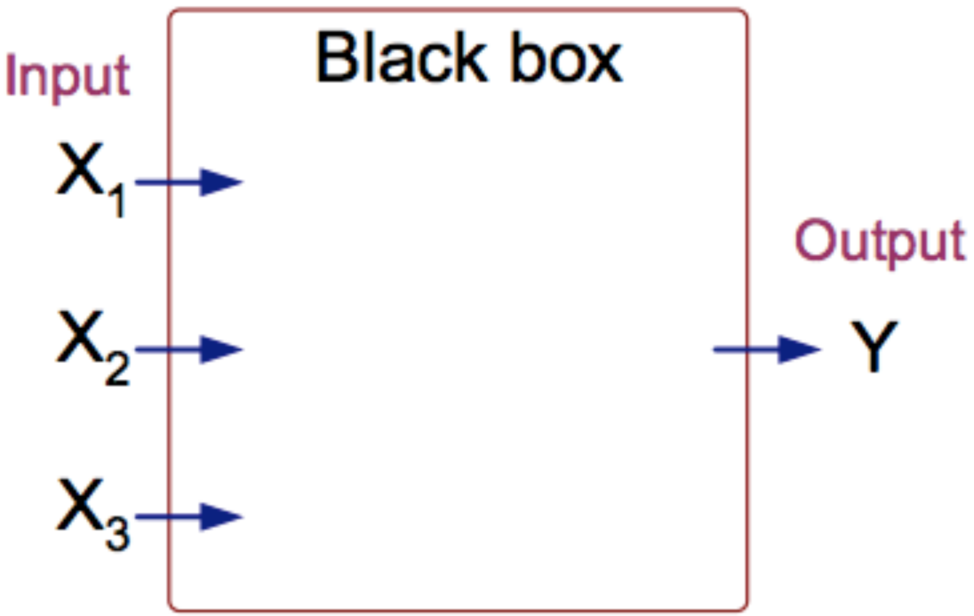


https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Perceptron: Learning Algorithm 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: Learning Algorithm Example - First 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

Predicted
?

w_0	w_1	w_2	w_3
0	0	0	0

Perceptron: Learning Algorithm Example - First 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

Predicted
1

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: Learning Algorithm Example - First Sample

- Update weights: $w_j = w_j + \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_j^{(i)}$; 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: Learning Algorithm Example - First Sample

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

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

Perceptron: Learning Algorithm Example - First Sample

- Update weights: $w_j = w_j + \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_j^{(i)}$; 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
-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)*0 = 0

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

Perceptron: Learning Algorithm Example - Second 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: Learning Algorithm Example - Second 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
-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: Learning Algorithm Example - Second Sample

- Update weights: $w_j = w_j + \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_j^{(i)}$; 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: Learning Algorithm Example - Second Sample

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

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

Perceptron: Learning Algorithm Example - Second Sample

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

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

Perceptron: Learning Algorithm Example - One Epoch (All Examples)

- $w_j = w_j + \eta\,(\text{target}^{(i)} - \text{output}^{(i)})\,x_j^{(i)}$; 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: Learning Algorithm Example - Six Epochs (All Examples)

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

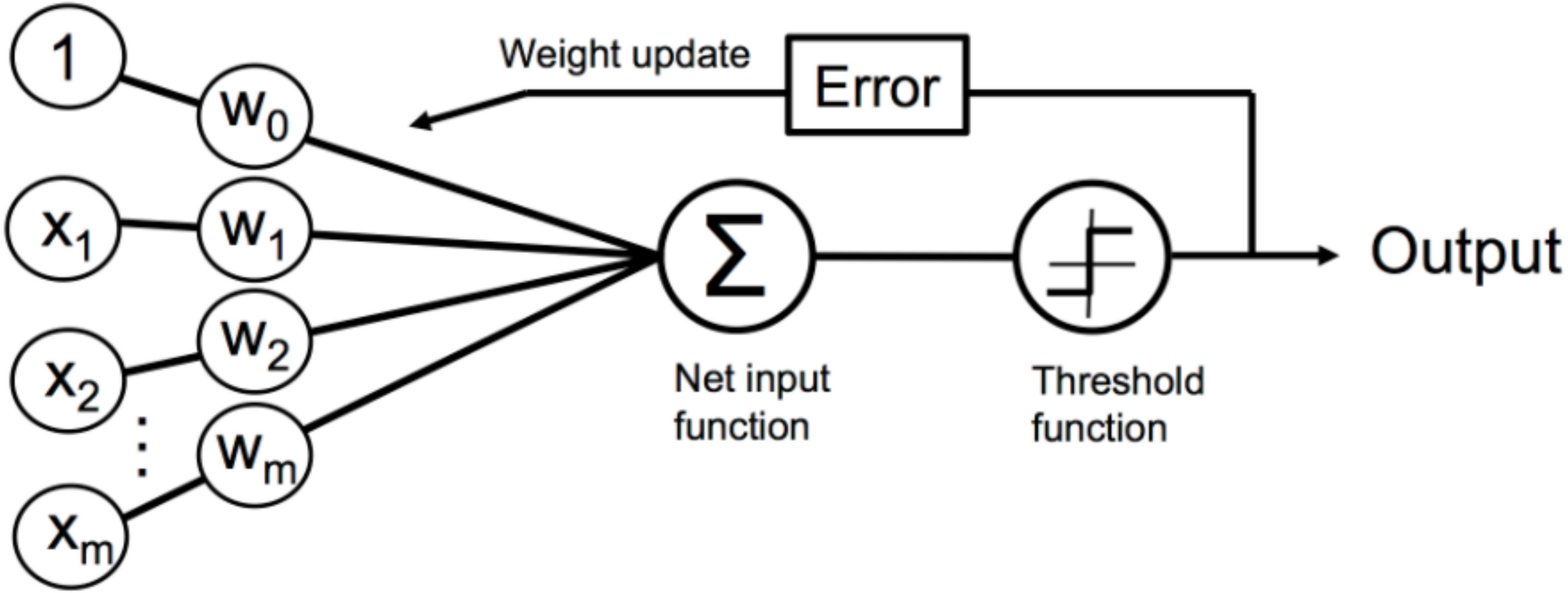
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

	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

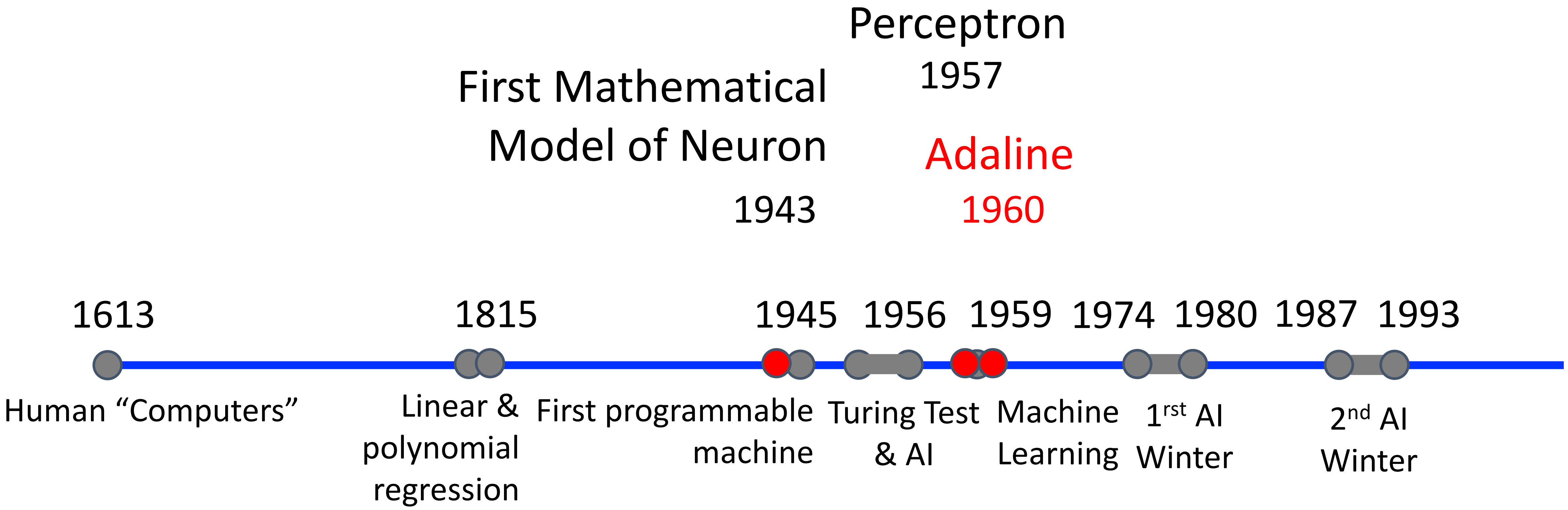


Python Machine Learning; Raschka & Mirjalili

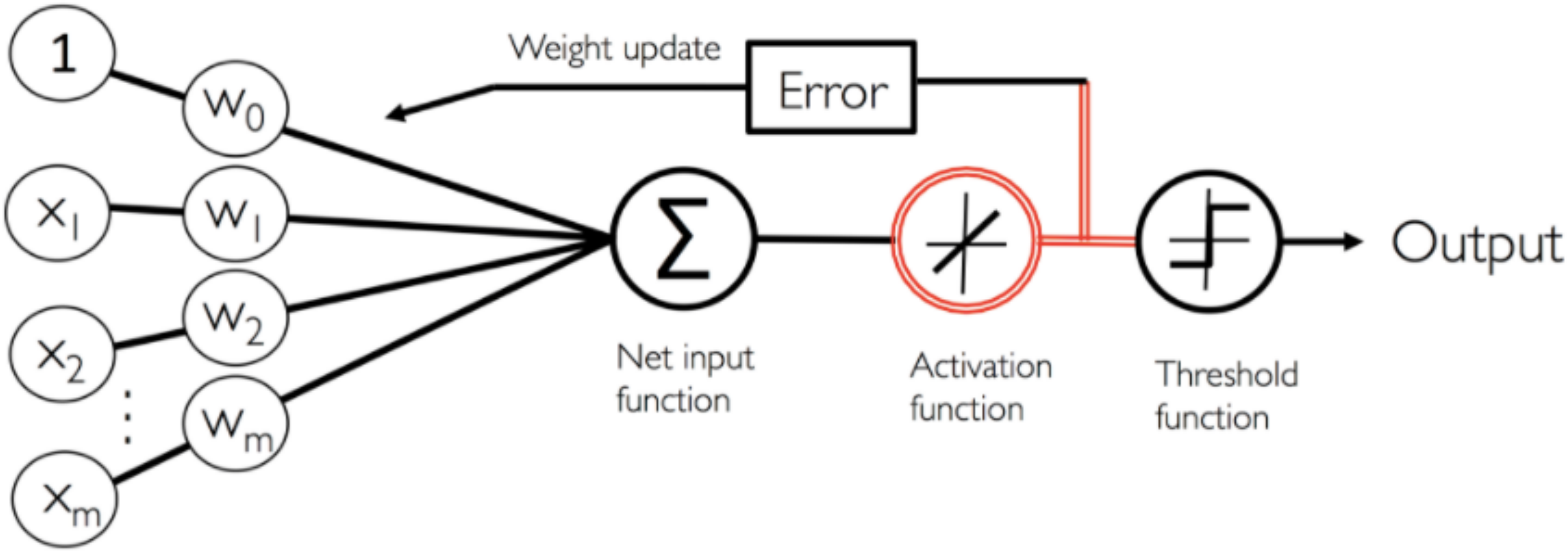
Perceptron: Learning Algorithm
- What are the Hyperparameters?

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

Artificial Neurons: Historical Context



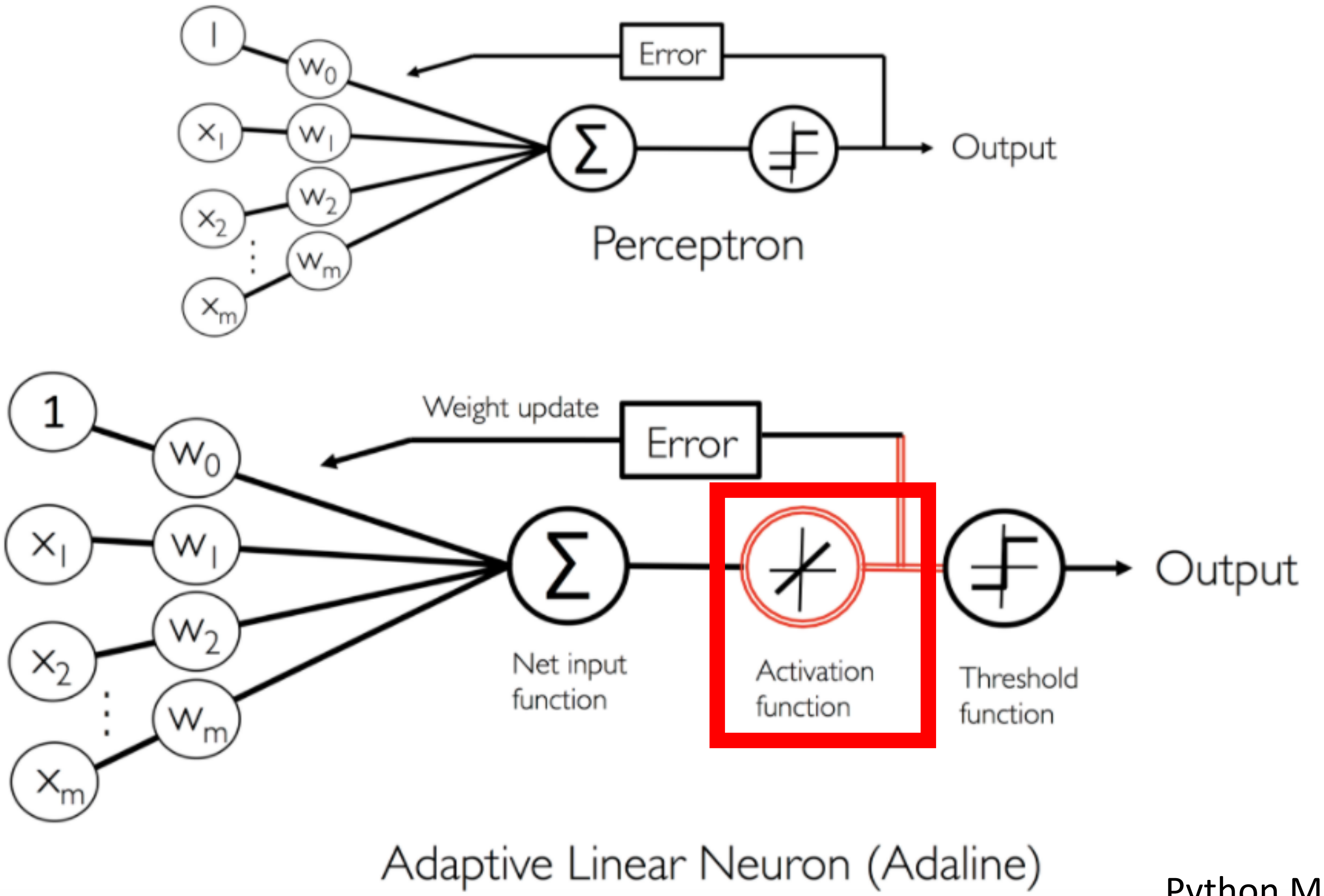
Adaline (**ADA**ptive **LI**near **NE**uron)



Python Machine Learning; Raschka & Mirjalili

Bernard Widrow and Ted Hoff, An Adaptive "Adaline" Neuron Using Chemical "Memistors", 1960

Adaline: Difference to Perceptron



Python Machine Learning; Raschka & Mirjalili

Adaline: Learning Algorithm

1. Initialize the weights to 0 or small random numbers.
2. For k epochs (passes over the training set)
    1. For each training sample
        1. Compute the predicted output value y
        2. Compare predicted to actual output and compute "weight update" value
        3. Update the "weight update" value
    2. Update weights with $\boxed{\text{accumulated "weight update" values}}$

Unlike Perceptron, does not make updates per sample

http://rasbt.github.io/mlxtend/user_guide/general_concepts/linear-gradient-derivative/

Adaline: Learning Algorithm

1. Initialize the weights to 0 or small random numbers.
2. For k epochs (passes over the training set)
    1. For each training sample
        1. Compute the predicted output value y
        2. Compare predicted to actual output and compute "weight update" value
        3. Update the "weight update" value
    2. Update weights with accumulated "weight update" values: $w_j := w_j + \boxed{\Delta w_j}$

Key Idea: this is differentiable!!!

$$J(\mathbf{w}) = \frac{1}{2}\boxed{\sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2}$$

$$\boxed{\Delta \mathbf{w} = -\eta \nabla J(\mathbf{w})}$$

Mathematical Simplification

Sum of squared errors

Learning Rate

Take step away from gradient

http://rasbt.github.io/mlxtend/user_guide/general_concepts/linear-gradient-derivative/

Adaline: Learning Algorithm
- Derivation of Equation to Update Weights

$$\frac{\partial J}{\partial w_j}$$

$$= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2$$

$$= \frac{1}{2} \frac{\partial}{\partial w_j} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2$$

$$= \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \phi(z)_A^{(i)}\right)$$

$$= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \sum_i \left(w_j^{(i)} x_j^{(i)}\right)\right)$$

$$= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)\left(-x_j^{(i)}\right)$$

$$= -\sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) x_j^{(i)}$$

http://rasbt.github.io/mlxtend/user_guide/general_concepts/linear-gradient-derivative/

Adaline: Learning Algorithm
- Derivation of Equation to Update Weights

$$\frac{\partial J}{\partial w_j}$$

$$= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2$$

$$= \frac{1}{2} \frac{\partial}{\partial w_j} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)^2$$

$$= \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \phi(z)_A^{(i)}\right)$$

$$= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \sum_i \left(w_j^{(i)} x_j^{(i)}\right)\right)$$

$$= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)}\right)\left(-x_j^{(i)}\right)$$

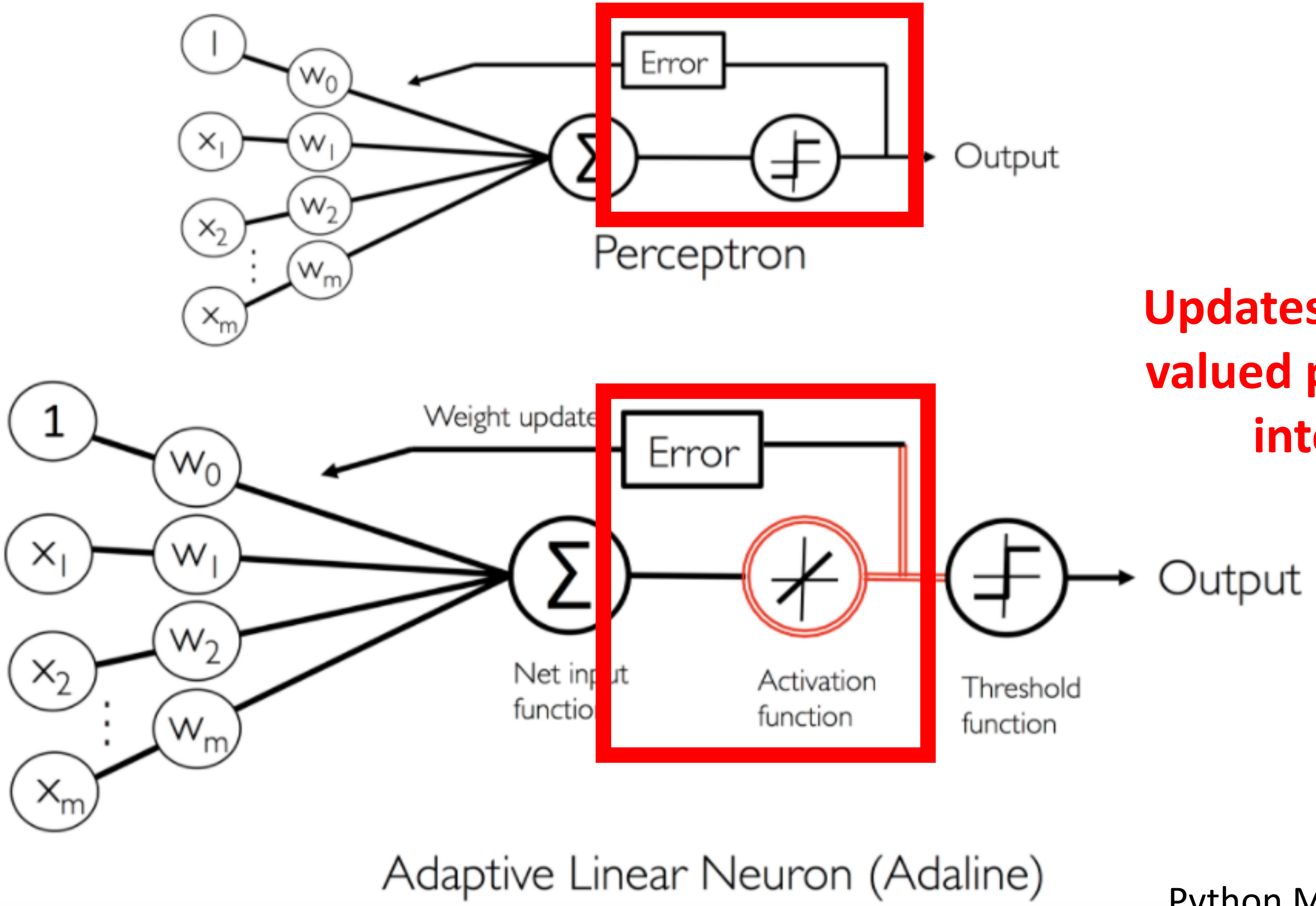
$$= -\boxed{\sum_i} \left(y^{(i)} - \boxed{\phi(z)_A^{(i)}}\right) x_j^{(i)}$$

Updates based on all samples

Updates based on continuous valued prediction!

http://rasbt.github.io/mlxtend/user_guide/general_concepts/linear-gradient-derivative/

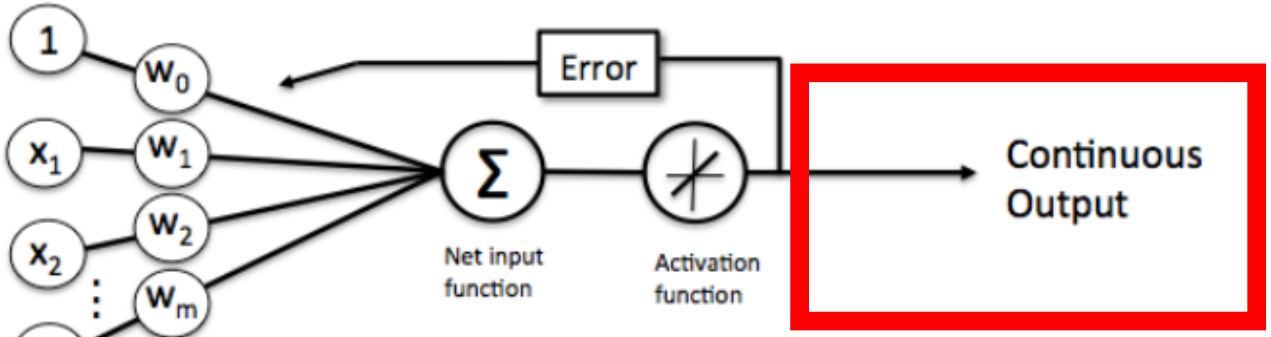
Adaline: Difference to Perceptron



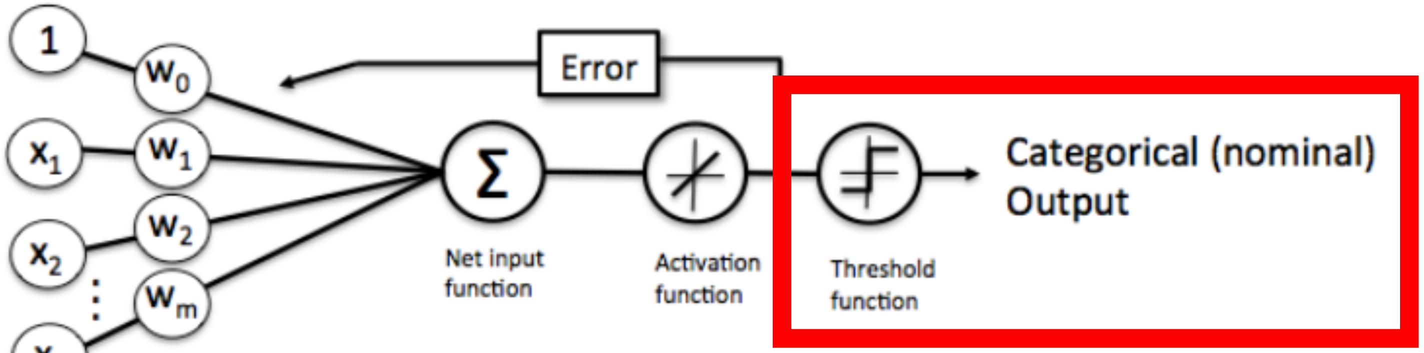
Updates based on continuous valued prediction rather than integer predictions!

Python Machine Learning; Raschka & Mirjalili

Adaline: Comparison to Linear Regression



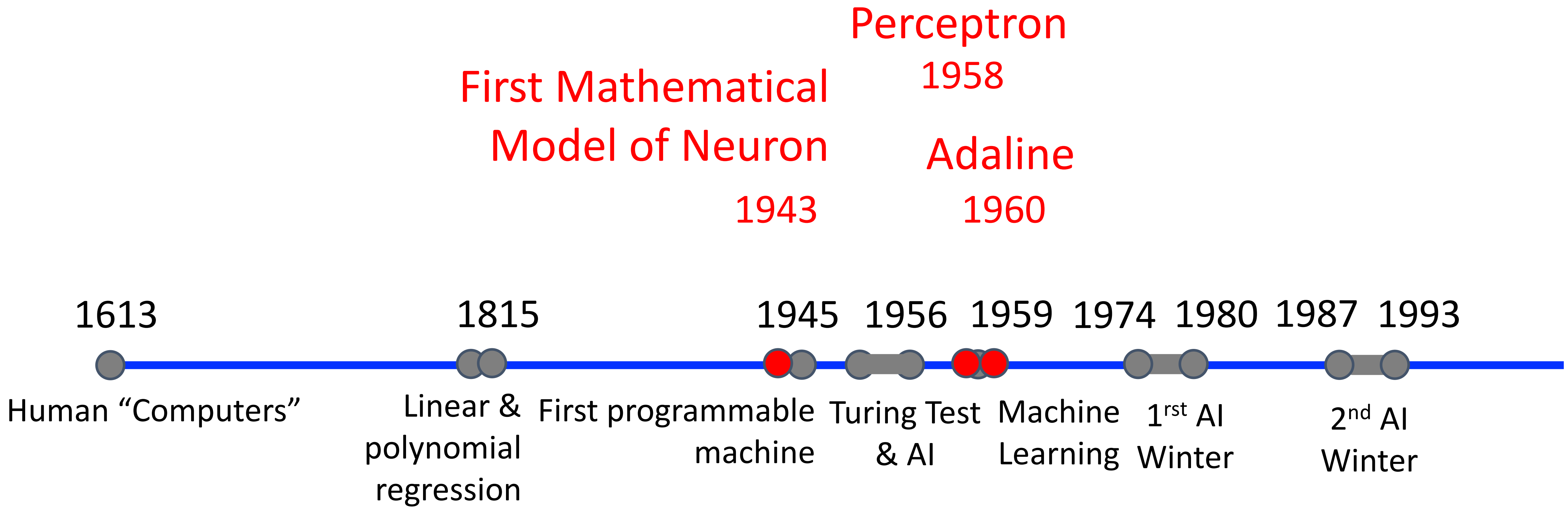
Linear Regression



Adaptive Linear Neuron (Adaline)

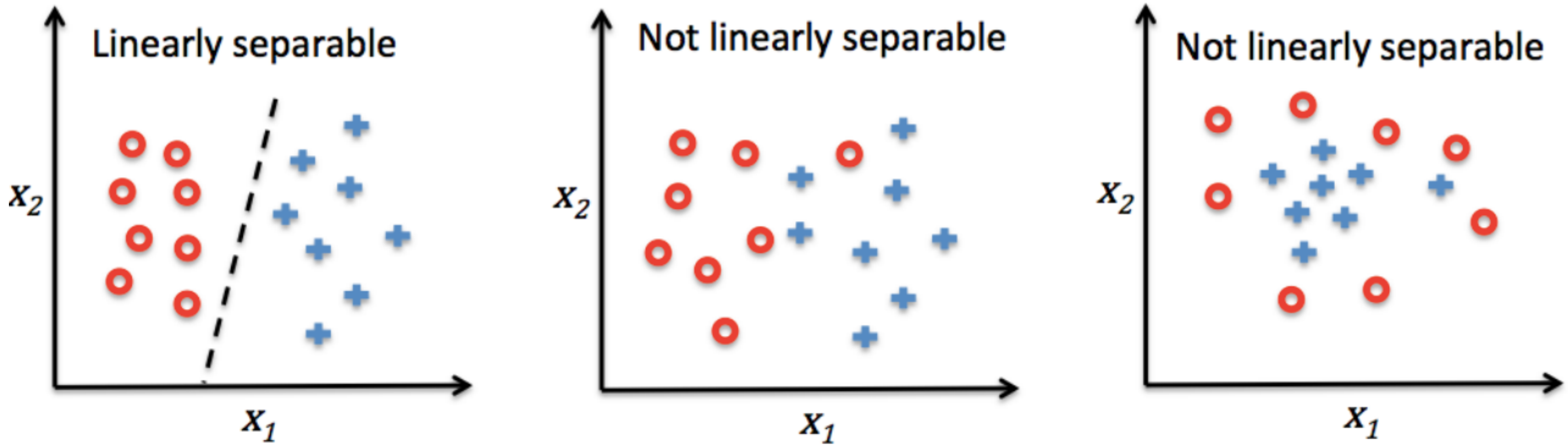
http://rasbt.github.io/mlxtend/user_guide/general_concepts/linear-gradient-derivative/

Artificial Neurons: Historical Context



Artificial Neurons: Limitations

1. Assumes Data is Linearly Separable



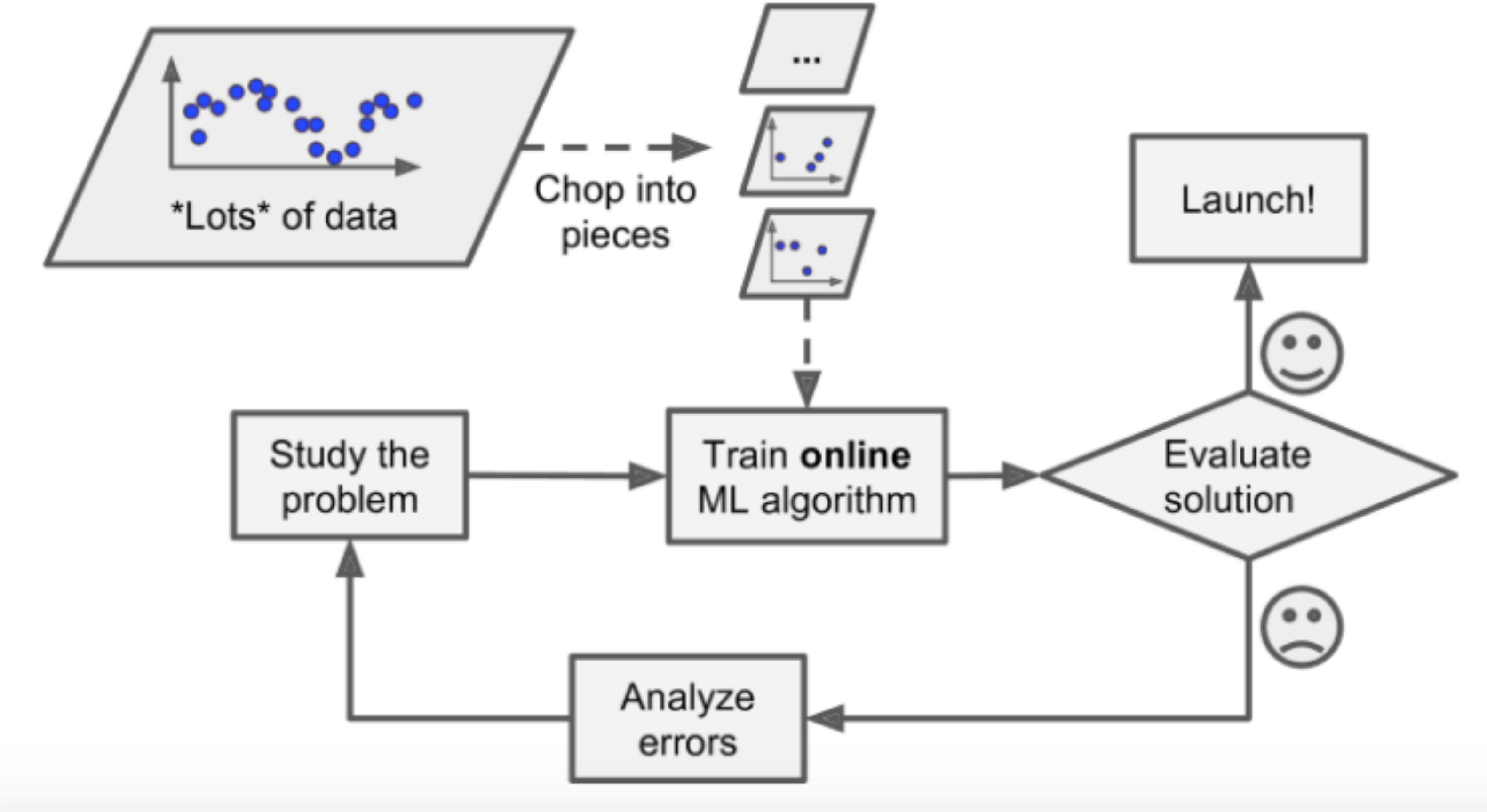
2. Results depend on initial values of weights
3. Despite clear weaknesses, artificial neurons are the foundation of today’s state-of-art machine learning algorithms

Python Machine Learning; Raschka & Mirjalili

Today’s Topics

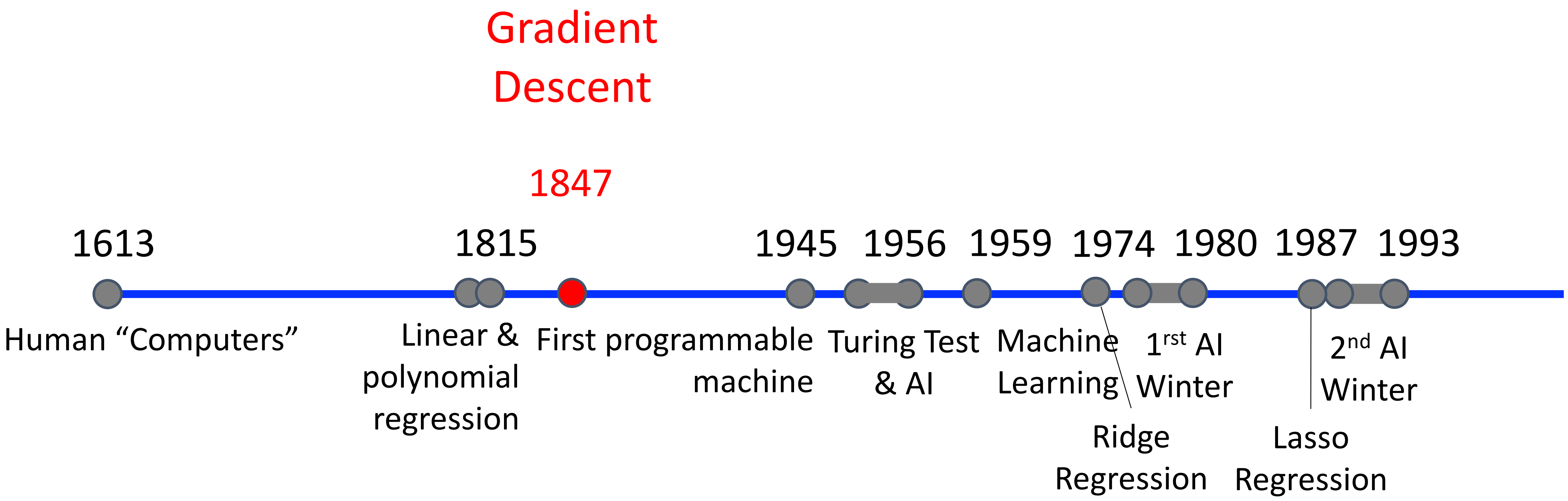
- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neurons: Perceptron & Adaline
- Gradient descent
- Lab

Learning Algorithm for Adaline:
- Gradient Descent (Optimization)



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

Gradient Descent (Optimization)



Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn linear model for converting kilometers to miles

Miles → Kilometers = miles x constant → Kilometers

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant → Error = Guess - Correct

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant → Error = Guess - Correct

Gradient Descent – Intuition

- Repeat:
  1. $\boxed{\text{Guess}}$
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

\$10 → Shekels = dollars x constant → Error = Guess - Correct

- Idea: iteratively adjust constant (i.e., model parameter) to try to reduce the error

Gradient Descent Algorithms

- Approach: solve mathematical problems by updating estimates of the solution via an iterative process to "optimize" a function
  - e.g., minimize or maximize an objective function f(x) by altering x

Start

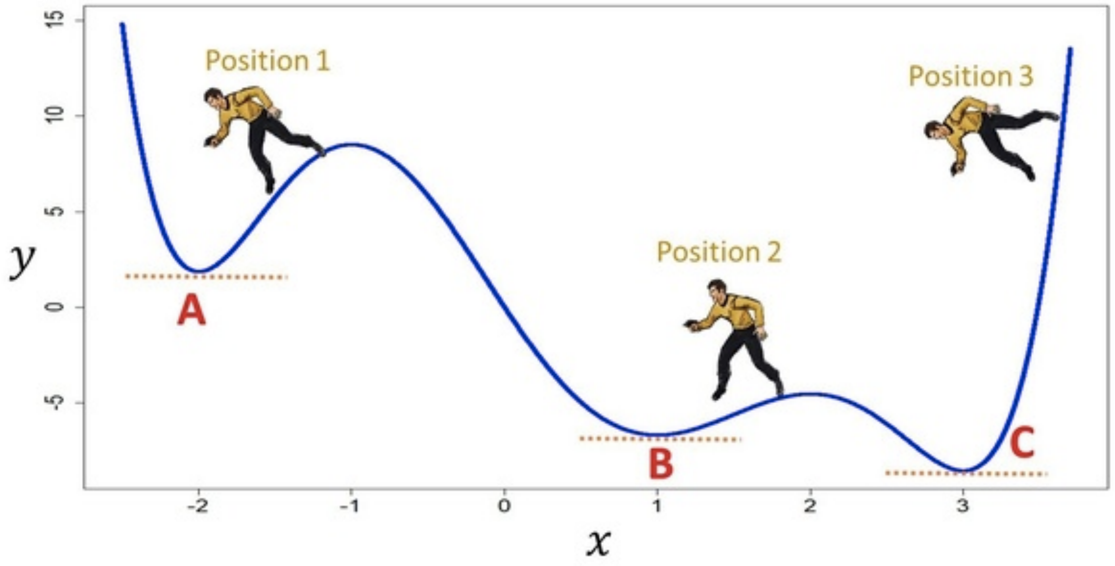
End Point (Minimum)

<u>Analogy</u>
Hiking to the bottom of a mountain range… blindfolded (or for a person who is blind)!

- When **minimizing** the objective function, it also is often called interchangeably the **cost function**, **loss function**, or **error function**.

Approach: Employ Calculus Concepts

- Idea: use derivatives!
  - Derivatives tells us how to change the input x to make a small change to the output f(x)
  - Functions with multiple inputs rely on a partial derivative for each input
- Gradient descent:
  - Iteratively update f(x) by moving x in small steps with the opposite sign of the derivative



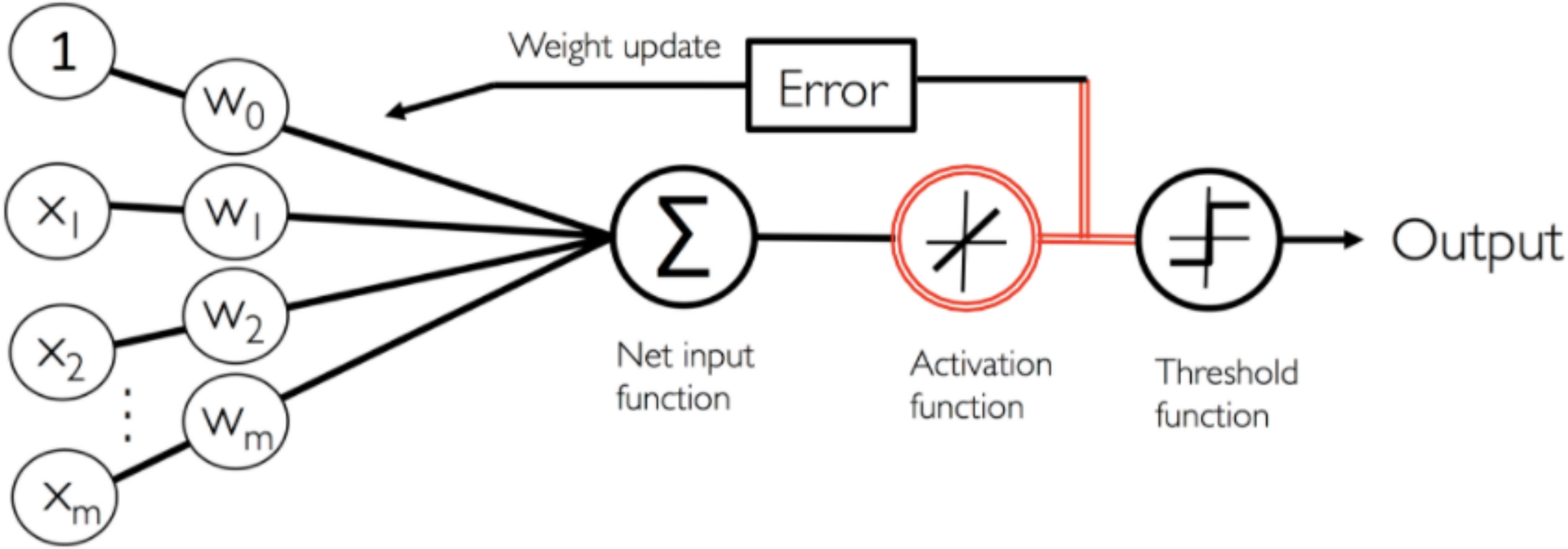
Which letter is the global minimum?

Which letter(s) are local minima?

Louis Augustin Cauchy: Compte Rendu `a l'Acad´emie des Sciences of October 18, 1847

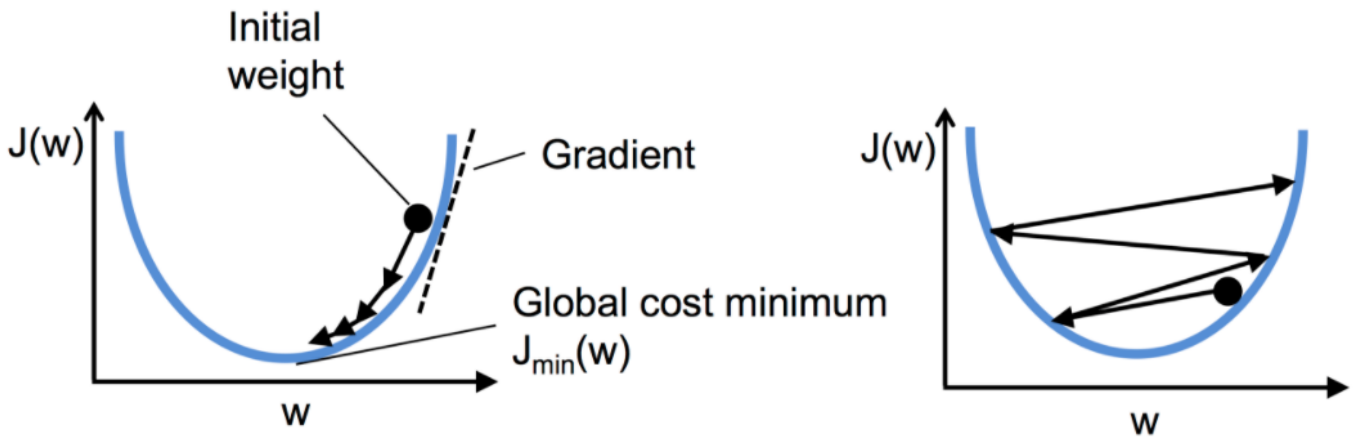
Gradient Descent – Relationship to Adaline

- What was trying to be minimized for Adaline?



Python Machine Learning; Raschka & Mirjalili

Gradient Descent: Influence of Learning Rate



- Learning Rate: amount new evidence is prioritized when updating weights
- What happens when learning rate is too small?
  - Convergence to good solution will be slow!
- What happens when learning rate is too large?
  - May not be able to converge to a good solution
- How to address the cons of different learning rates?
  - Gradually reduce learning rate over time

https://github.com/rasbt/python-machine-learning-book-2nd-edition/blob/master/code/ch02/ch02.ipynb

Batch Gradient Descent (BGD)

- For each step (update), use calculations over ***all training examples***
- What are strengths of this approach?
  - Does not bounce too much
- What are weaknesses of this approach?
  - Very slow or infeasible when dataset is large
- Which algorithm uses this?
  - Adaline

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

Stochastic Gradient Descent (SGD)

- For each step (update), use calculations from ***one training example***
- What are strengths of this approach?
  - Each iteration is fast to compute
  - Can train using huge datasets (stores one instance in memory at each iteration)
- What are weaknesses of this approach?
  - Updates will bounce a lot

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

Mini-batch Gradient Descent

- For each step (update), use calculations over ***subset of training examples***
- What are strengths of this approach?
  - Bounces less erratically when finding model parameters than SGD
  - Can train using huge datasets (store some instances in memory at each iteration)
- What are weaknesses of this approach?
  - Very slow or infeasible when dataset is large
- Which algorithm uses this?
  - To be explored in future classes

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

Today's Topics

- Binary classification applications
- Evaluating classification models
- Biological neurons: inspiration
- Artificial neurons: Perceptron & Adaline
- Gradient descent
- Lab

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