Artificial Neurons and Gradient Descent

Spring 2018

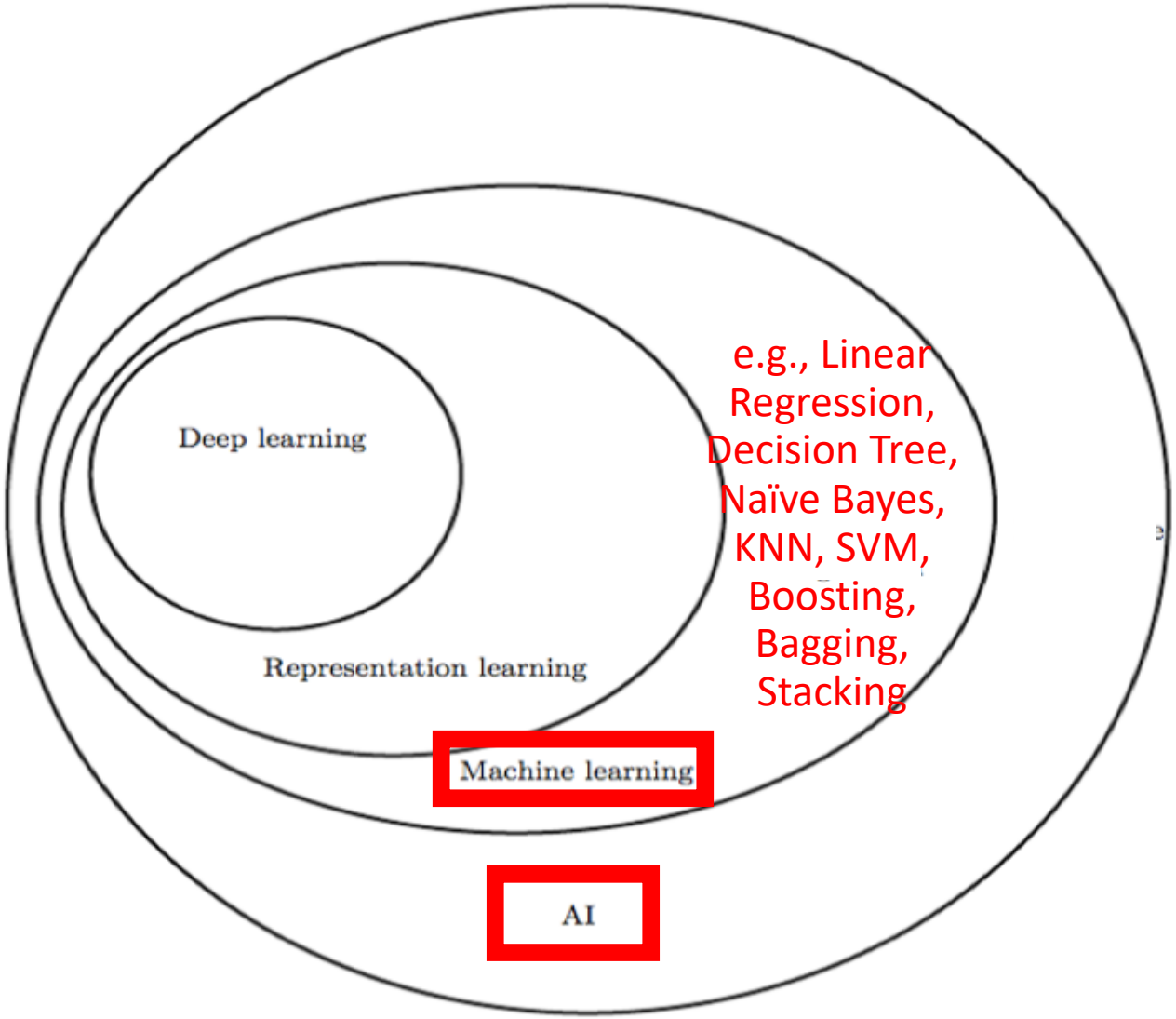
Today's Topics

- History of Modern Neural Networks and "Deep Learning"
- Single Layer Neural Network: Perceptron
- Single Layer Neural Network: Adaline
- Variants of Gradient Descent
- Lab

Today's Topics

- History of Modern Neural Networks and "Deep Learning"
- Single Layer Neural Network: Perceptron
- Single Layer Neural Network: Adaline
- Variants of Gradient Descent
- Lab

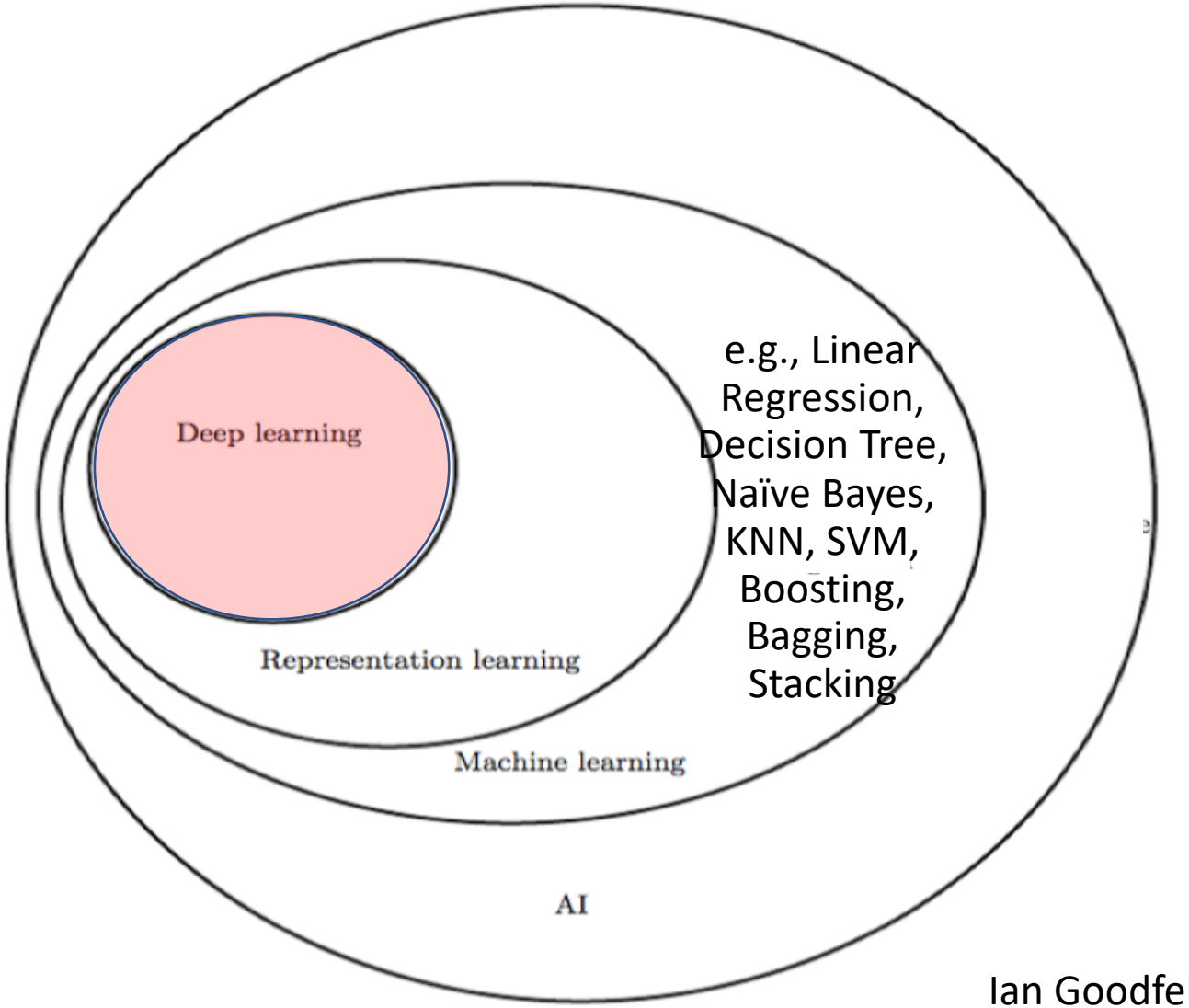
Review: AI & Relationship to Different Disciplines



- What is artificial intelligence?
  - 1956: machines that do intelligent things
- What is machine learning?
  - 1956: algorithms that learn on their own
- What machine learning algorithms have we studied so far?

llow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016.

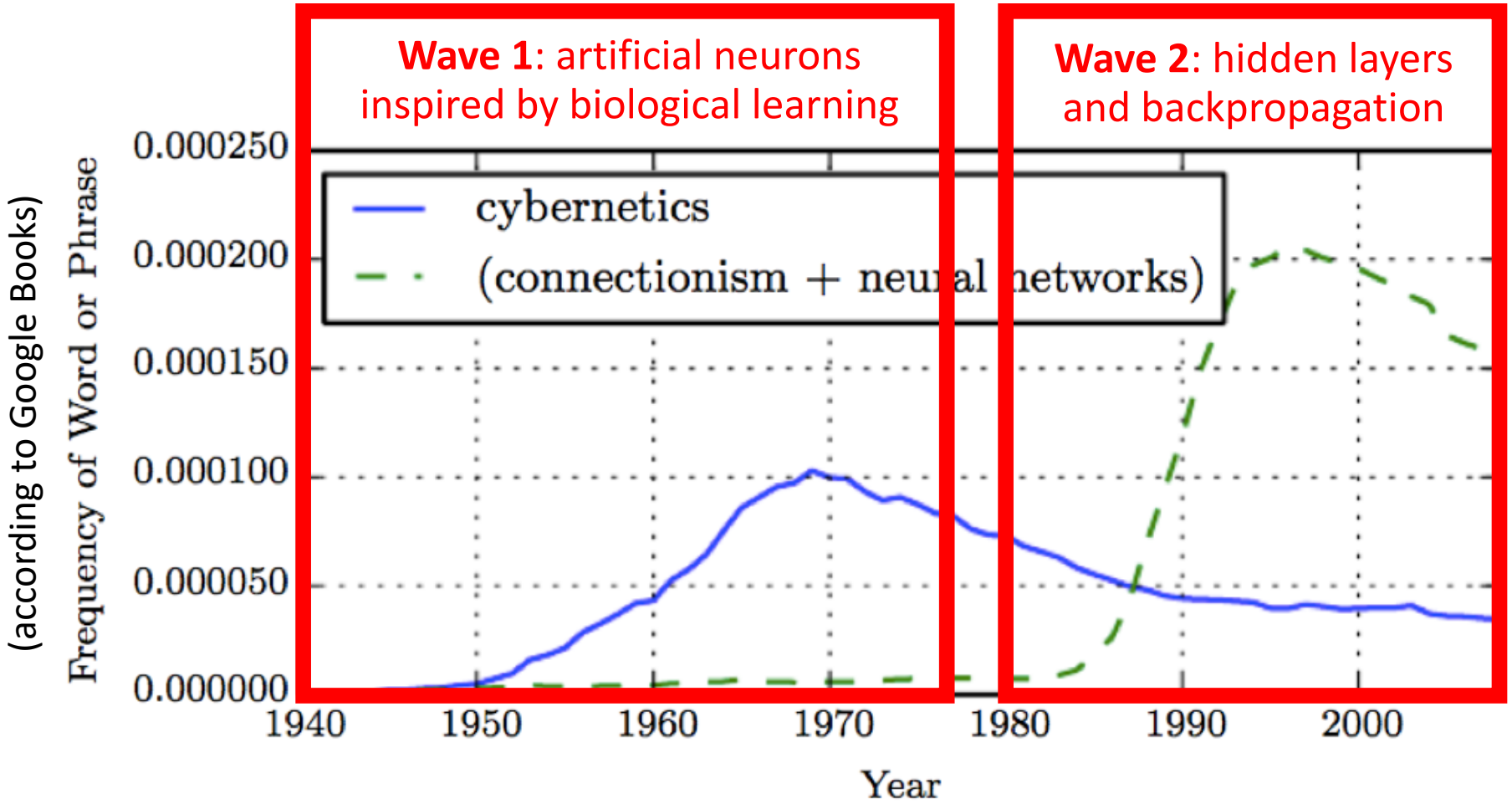
Focus in Next Few Weeks: Deep Learning



- What is artificial intelligence?
  - 1956: machines that do intelligent things
- What is machine learning?
  - 1956: algorithms that learn on their own

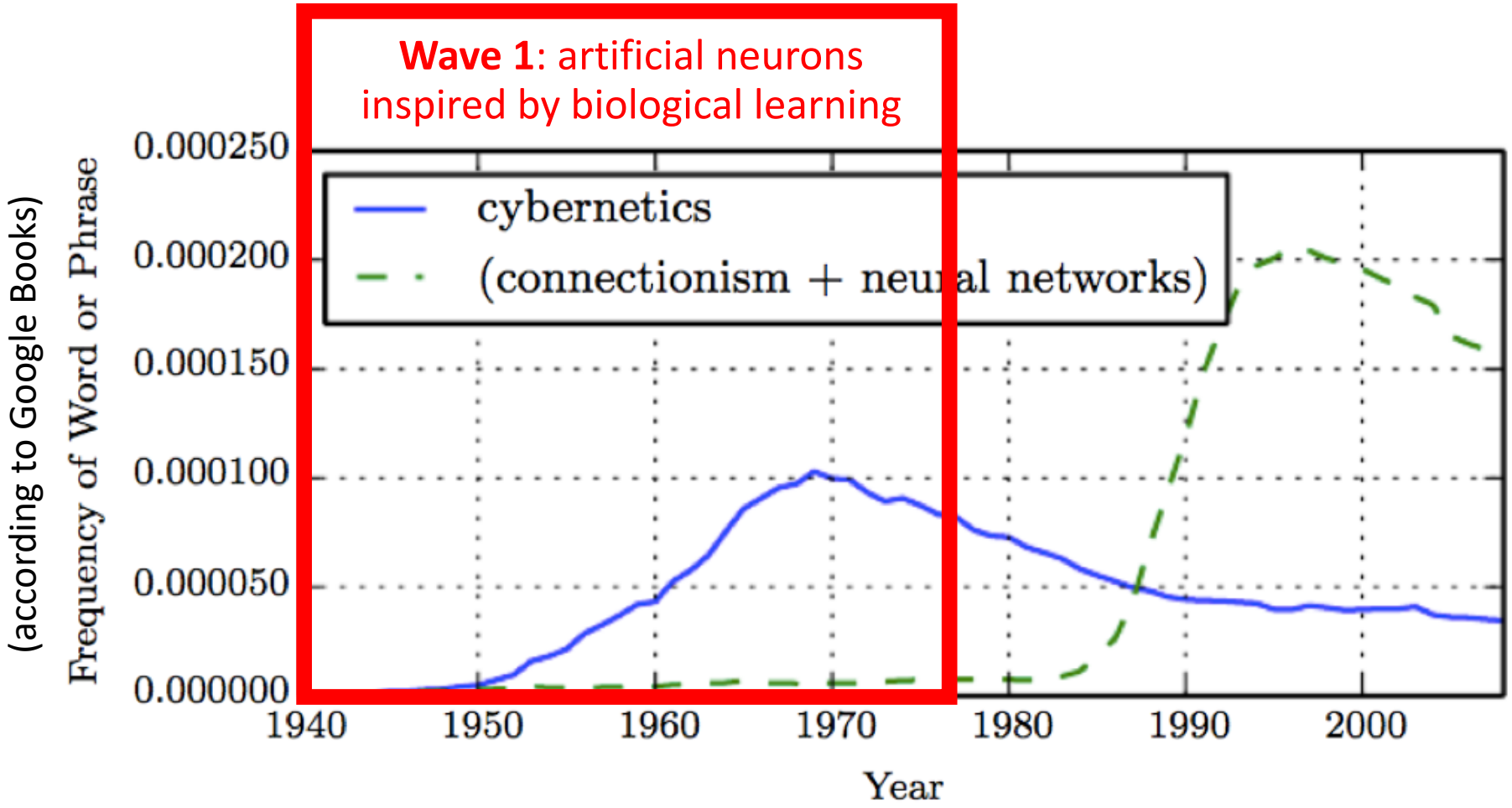
Ian Goodfellow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016.

The History of “Deep Learning”

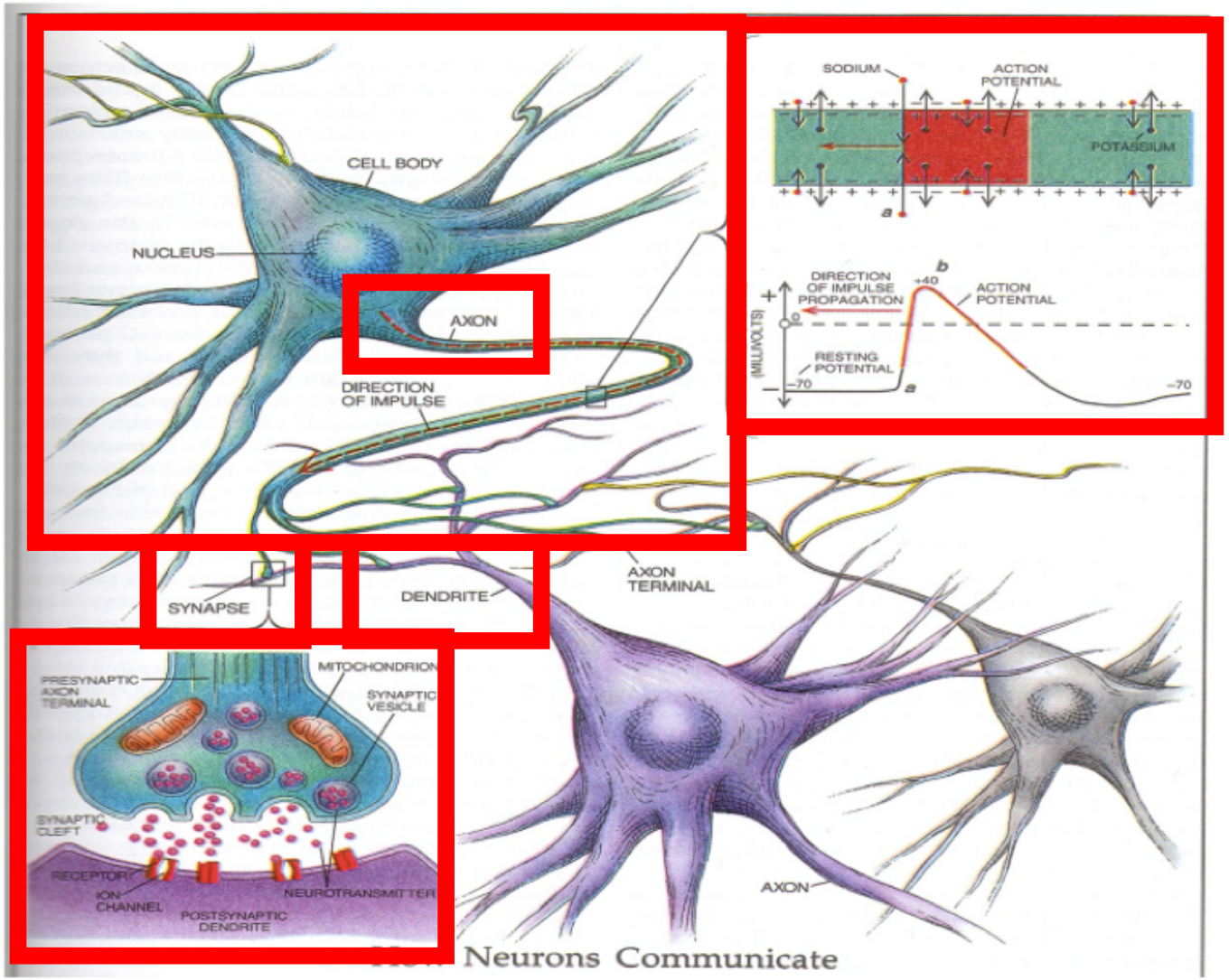


Wave 3: field emerged again as “deep learning” in 2006

Today's Topic: Artificial Neurons



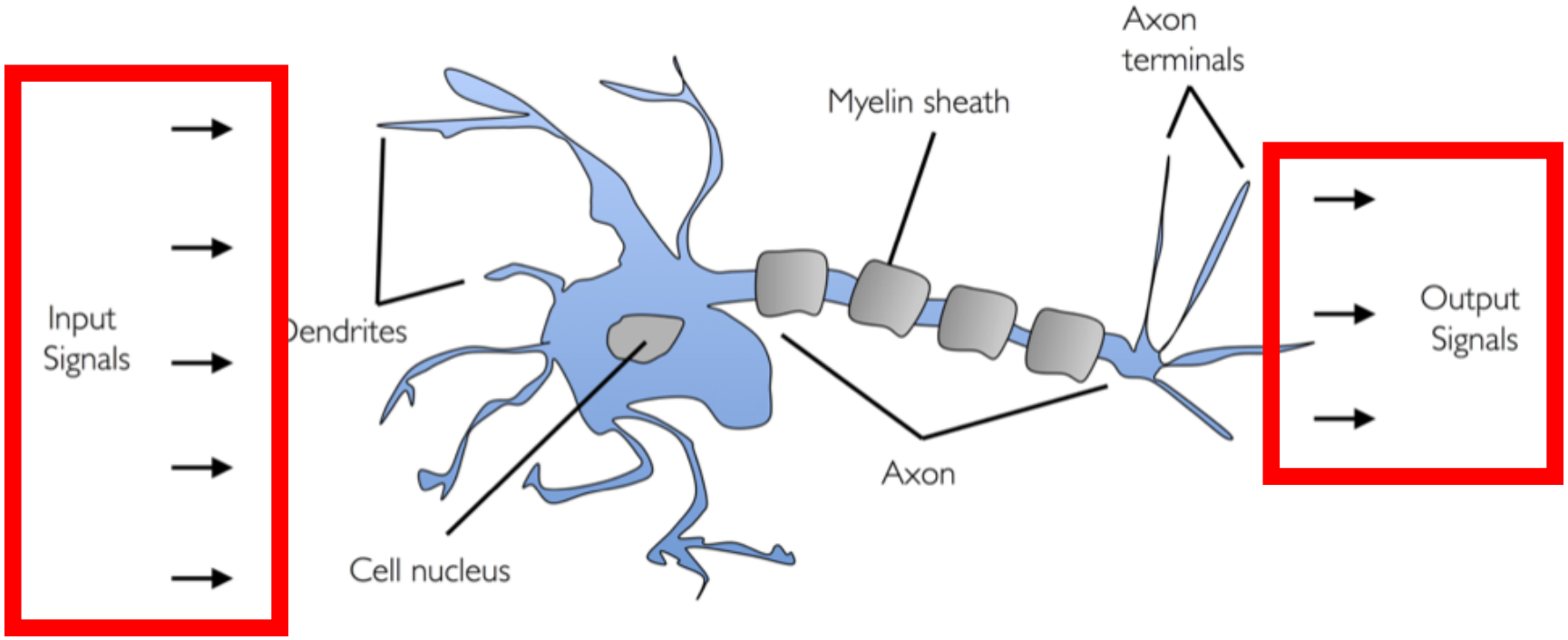
Biological Neuron: Basic Logic Unit in Brain



- What is a neuron?
  - Primary component of central nervous system for receiving, processing, and transmitting information
- How does a neuron communicate?
  - Synapse connects neurons
  - It receives information from its dendrites
  - It transmits information through its axon
  - It transmits information from its axon to another neuron's dendrites
  - It "Fires" (transmits information) ONLY when triggered above a certain voltage
- How many neurons does a human have?
  - Approximately 10^{11}

https://people.eecs.berkeley.edu/~jrs/189/lec/18.pdf

Artificial Neuron: Model of Biological Neuron



- Neuron “firing” (outputs signal) is an “all-or-none” process.
- A fixed number of input signals must be received within a short period of time for a neuron to “fire”

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

Artificial Neuron: McCulloch-Pitts Neuron

- 1943: First mathematical model of neuron proposed by…

Warren McCulloch: Neurophysiologist

Walter Pitts: Mathematician

- Recall: first computing machines created in early 1940s

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

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

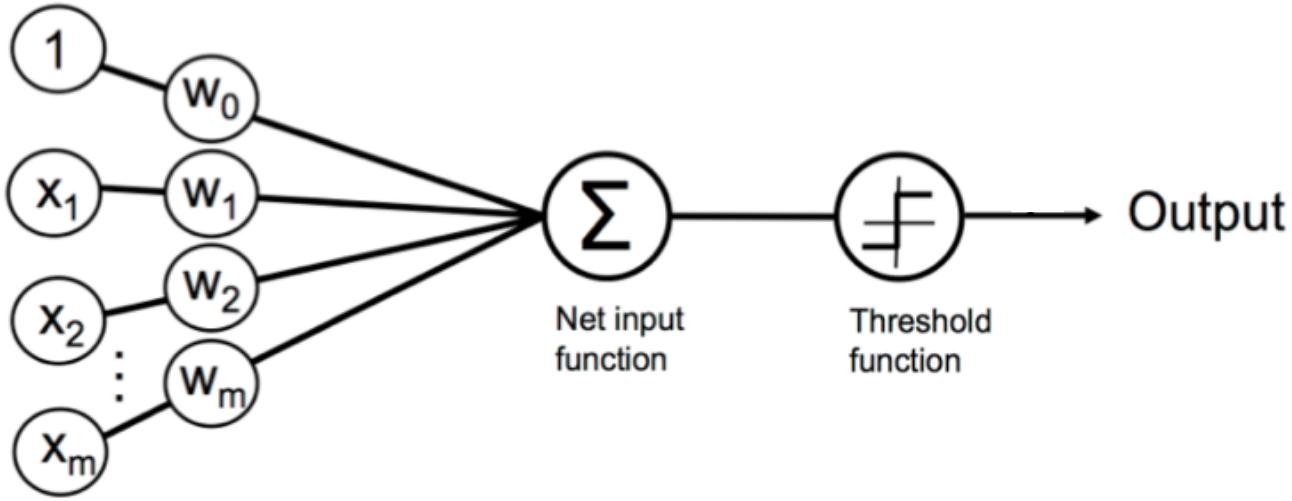
Warren McCulloch and Walter Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, 1943.

Today's Topics

- History of Modern Neural Networks and "Deep Learning"
- Single Layer Neural Network: Perceptron
- Single Layer Neural Network: Adaline
- Variants of Gradient Descent
- Lab

Artificial Neuron: Perceptron

- 1958: Linear Threshold Unit (LTU)
  - Input features
  - Model coefficients
  - Outputs weighted sum of inputs

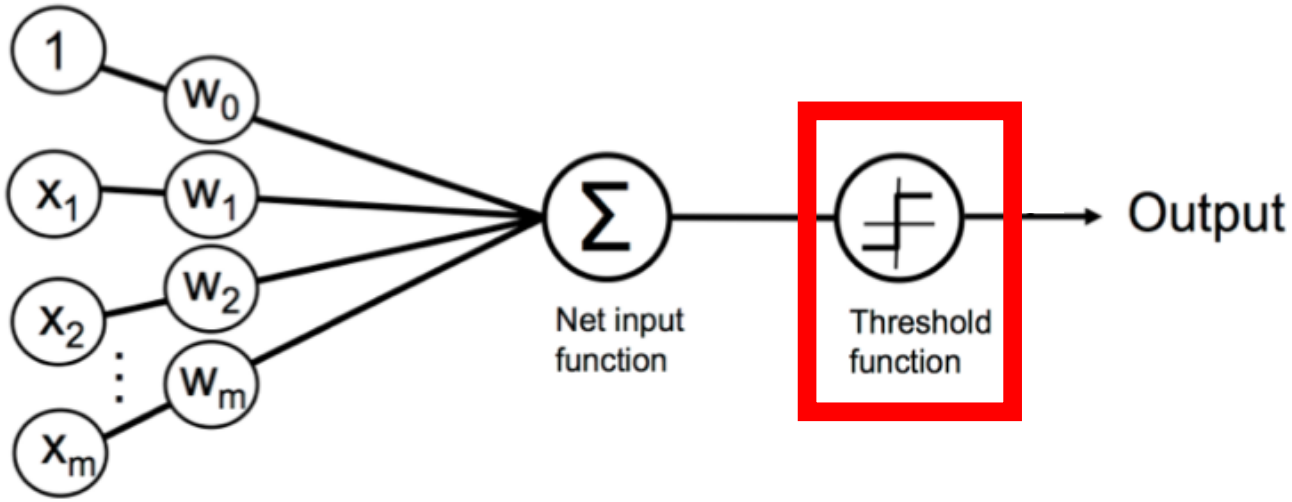


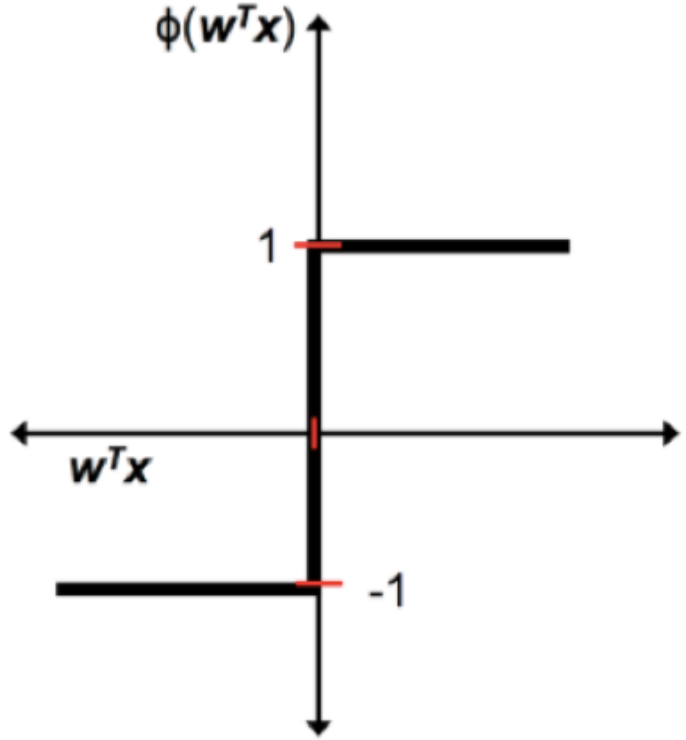
Python Machine Learning; Raschka & Mirjalili

Frank Rosenblatt, The **perceptron**: a probabilistic model for information storage and organization in the brain, 1958.

Artificial Neuron: Perceptron

- Formal definition of artificial neuron: $\phi(w^T x) = \begin{cases} 1 \text{ if } \phi(w^T x) \geq \theta \\ -1 \text{ otherwise} \end{cases}$
- Decision function is a unit step function:
  - "fire" or "do not fire"
  - mimics human brain



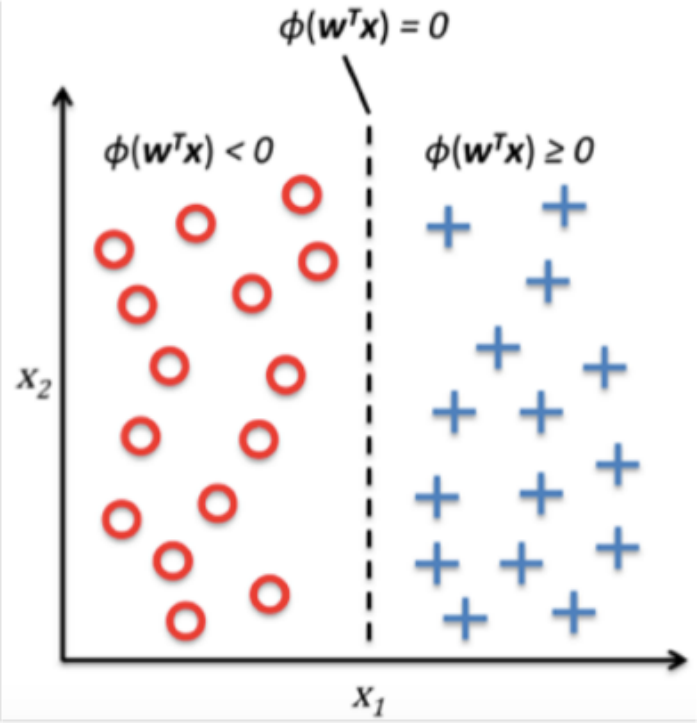


Python Machine Learning; Raschka & Mirjalili

Artificial Neuron: Perceptron

- Formal definition of artificial neuron: $\phi(\boldsymbol{w}^T\boldsymbol{x}) = \begin{cases} 1 \text{ if } \phi(\boldsymbol{w}^T\boldsymbol{x}) \geq 0 \\ -1 \text{ otherwise} \end{cases}$
- Rewrite to: $w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}$

$-\theta$ 1



Python Machine Learning; Raschka & Mirjalili

Perceptron Learning Algorithm

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

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

Learning Rate

True Class Label

Predicted Class Label

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

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

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

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

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

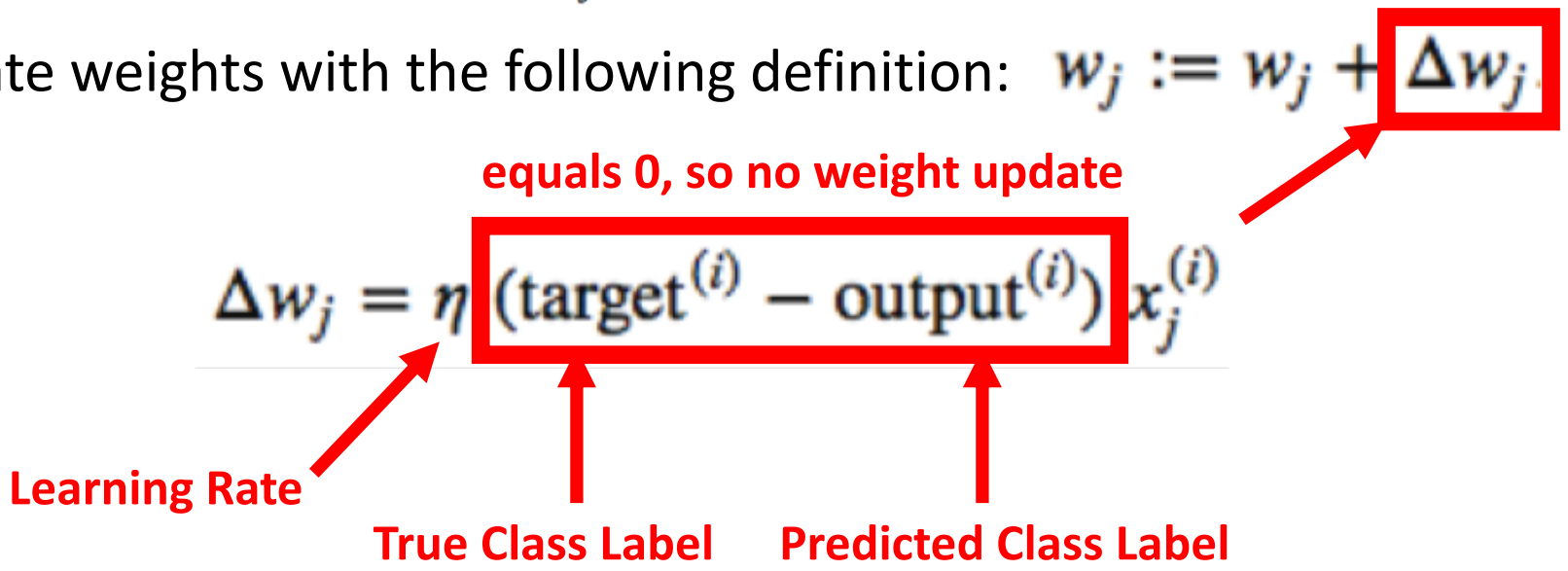
$$\Delta w_2 = \eta\ (\text{target}^{(i)} - \text{output}^{(i)})\ x_2^{(i)}$$

All weights updated simultaneously

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

Learning: What Happens to Weights When Perceptron Predicts **Correct** Class Label?

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



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

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

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

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

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

Learning Rate

True Class Label

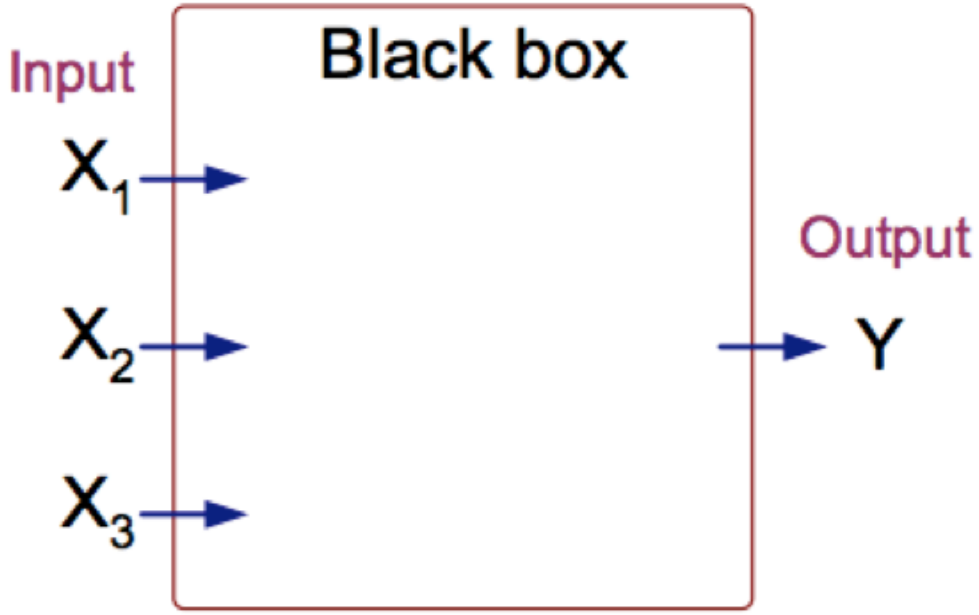
Predicted Class Label

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

Learning Example: Example Data

- 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	

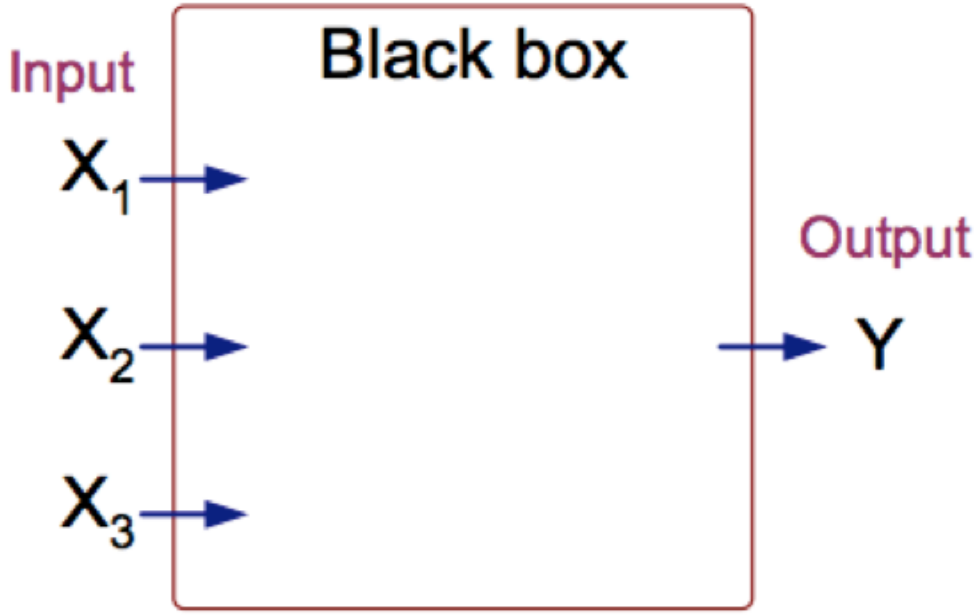


https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Learning Example: Example Data

- 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

Learning Example: Learn with **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

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

Learning Example: Learn with **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

Learning Example: Learn with **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

Learning Example: Learn with **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
?	?	?	?

$\Delta w_0 = 0.1(-1-1)*1 = -0.2$

$\Delta w_1 = 0.1(-1-1)*1 = -0.2$

$\Delta w_2 = 0.1(-1-1)*0 = 0$

$\Delta w_3 = 0.1(-1-1)*0 = 0$

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

Learning Example: Learn with **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

$\Delta w_0 = 0.1(-1-1)*1 = -0.2$

$\Delta w_1 = 0.1(-1-1)*1 = -0.2$

$\Delta w_2 = 0.1(-1-1)*0 = 0$

$\Delta w_3 = 0.1(-1-1)*0 = 0$

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

Learning Example: Learn with **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

Learning Example: Learn with **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

Learning Example: Learn with **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

Learning Example: Learn with **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
?	?	?	?

$\Delta w_0 = 0.1(1--1)*1 = 0.2$

$\Delta w_1 = 0.1(1--1)*1 = 0.2$

$\Delta w_2 = 0.1(1--1)*0 = 0$

$\Delta w_3 = 0.1(1--1)*1 = 0.2$

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

Learning Example: Learn with **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

$\Delta w_0 = 0.1(1--1)*1 = 0.2$

$\Delta w_1 = 0.1(1--1)*1 = 0.2$

$\Delta w_2 = 0.1(1--1)*0 = 0$

$\Delta w_3 = 0.1(1--1)*1 = 0.2$

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

Learning 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

Learning Example: 6 Epochs (All Examples)

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

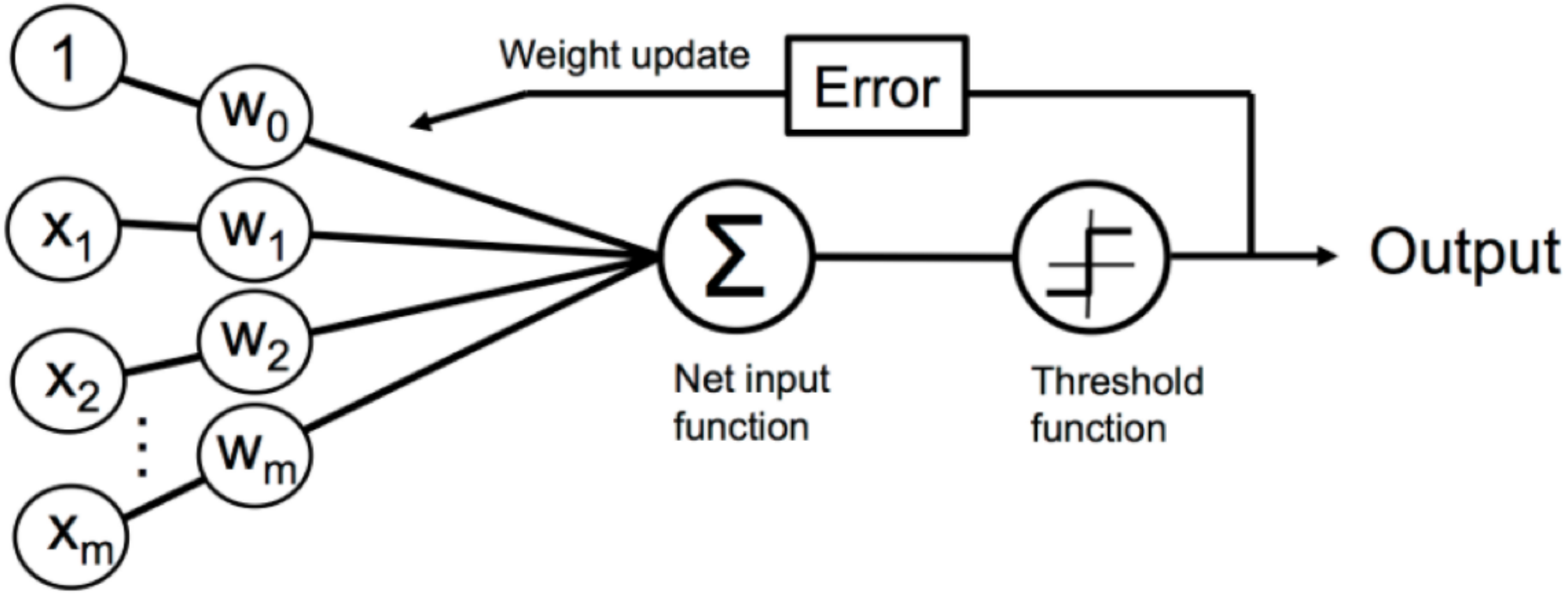
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

	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 Pictorially



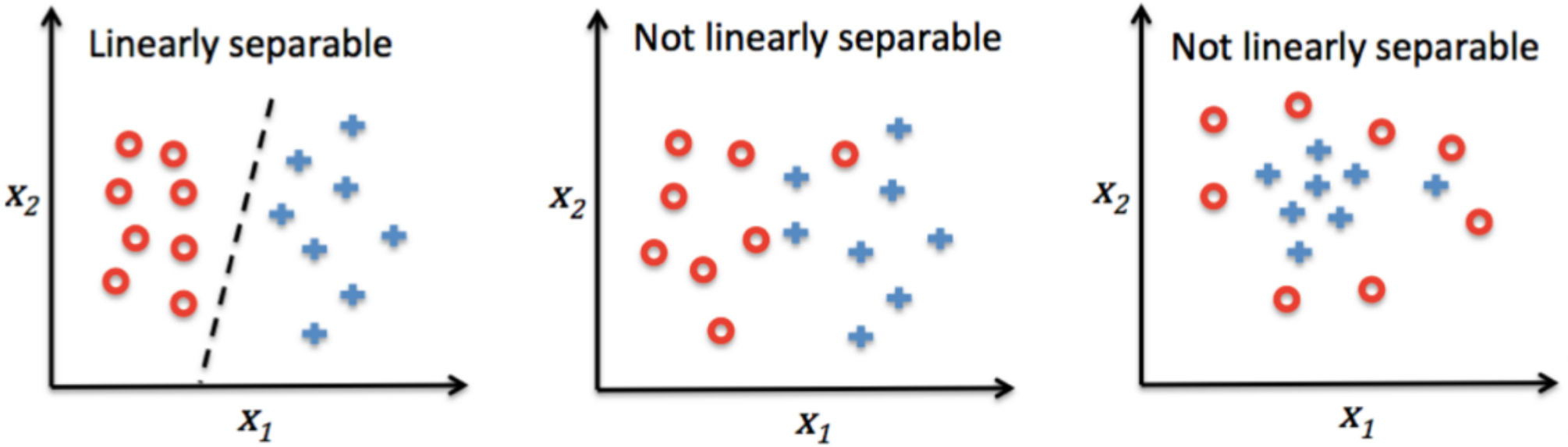
Python Machine Learning; Raschka & Mirjalili

Perceptron Learning Algorithm: What are the Hyperparameters?

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

Limitations

1. Assumes Data is Linearly Separable



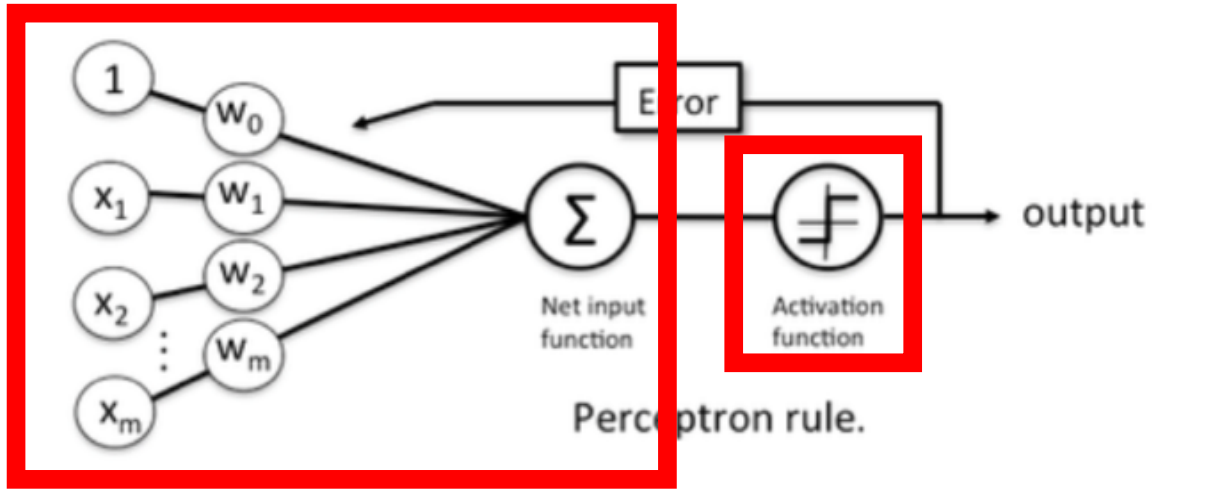
2. Results depend on initial values of weights

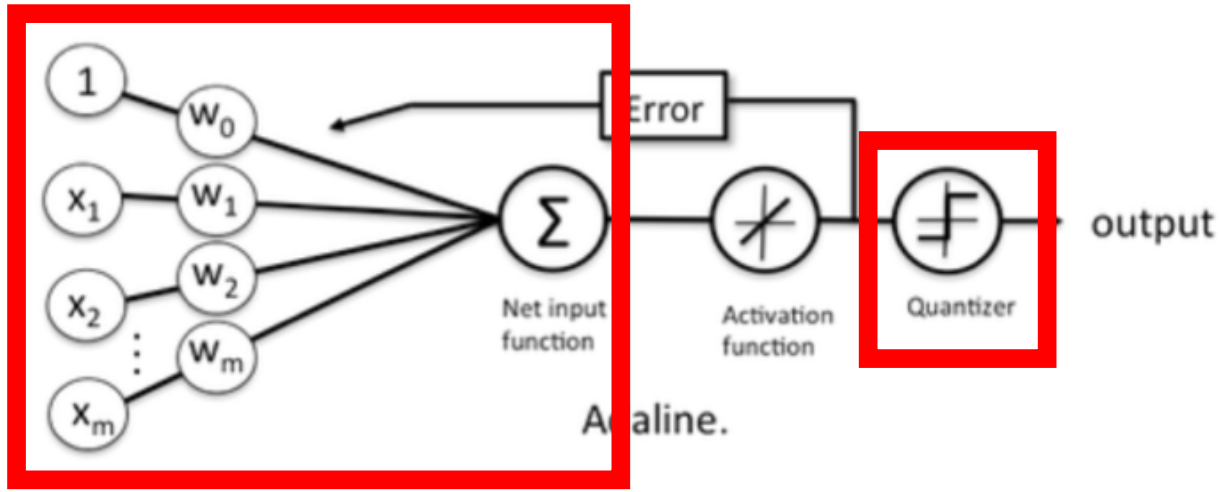
Python Machine Learning; Raschka & Mirjalili

Today's Topics

- History of Modern Neural Networks and "Deep Learning"
- Single Layer Neural Network: Perceptron
- Single Layer Neural Network: Adaline
- Variants of Gradient Descent
- Lab

Adaline (**ADA**ptive **LI**near **NE**uron): Similarity to Perceptron



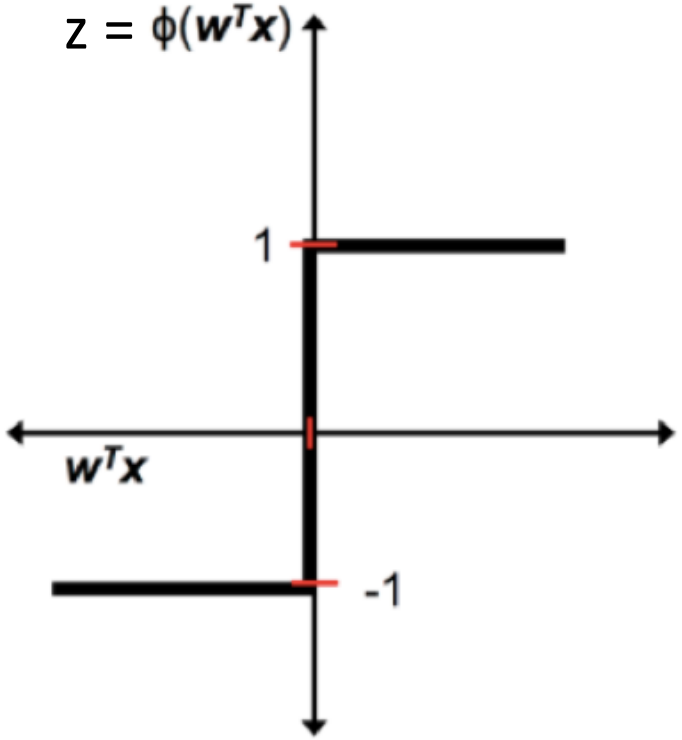


Python Machine Learning; Raschka & Mirjalili

Bernard Widrow and Ted Hoff, An Adaptive “Adaline” Neuron Using Chemical “Memistors”, 1960.

Adaline: Similarity to Perceptron

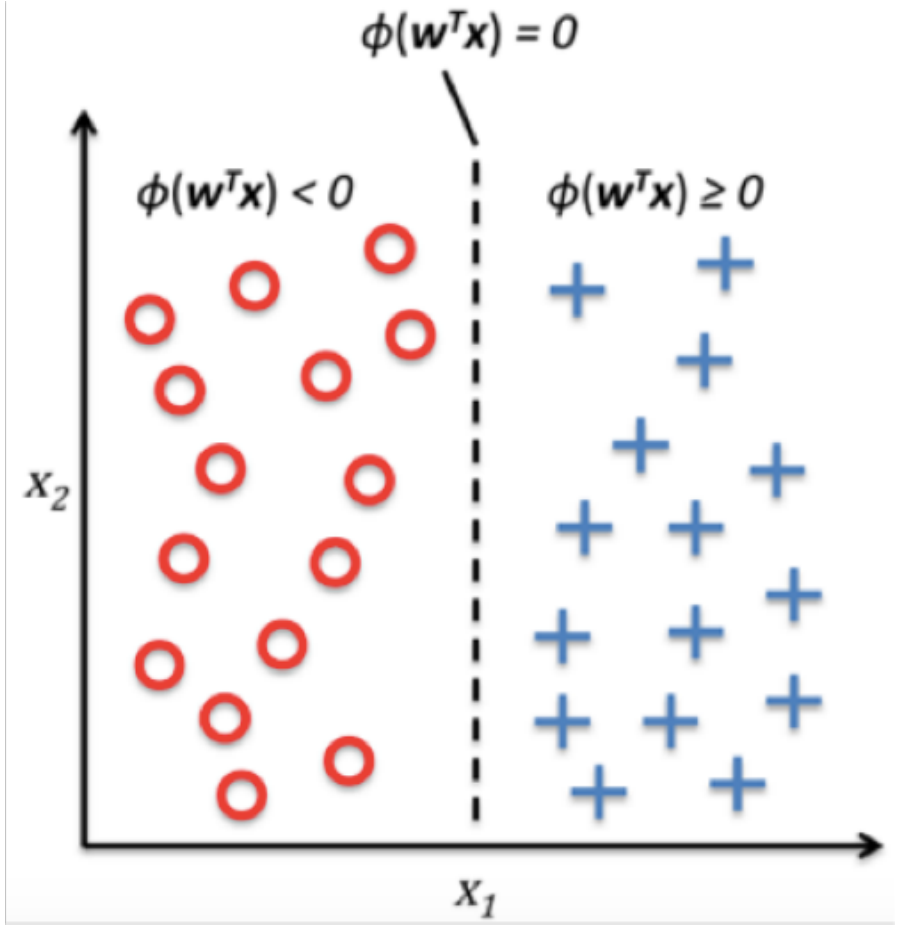
- Formal definition of artificial neuron: $z = \begin{cases} 1 \text{ if } z \geq \theta \\ -1 \text{ otherwise} \end{cases}$
- Decision function is a unit step function:
  - "fire" or "do not fire"
  - mimics human brain



Python Machine Learning; Raschkka & Mirjalili

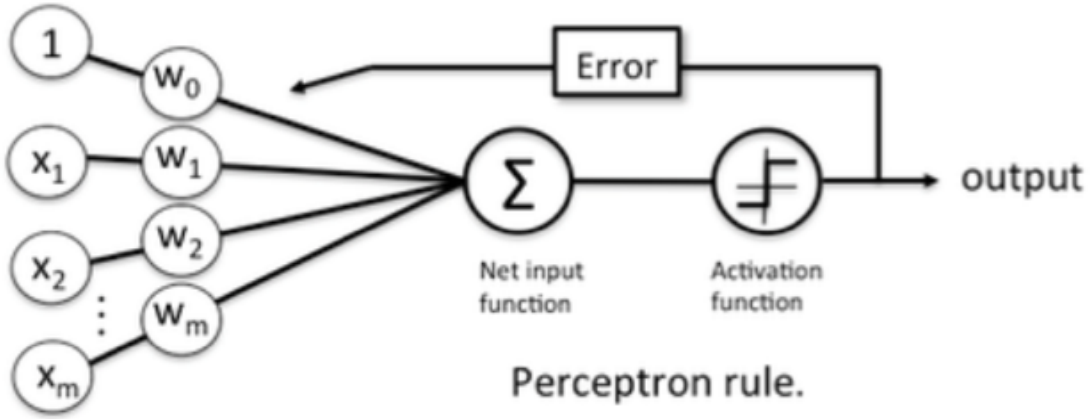
Adaline: Similarity to Perceptron

- Formal definition of artificial neuron: $z = \begin{cases} 1 \text{ if } z \geq \theta \\ -1 \text{ otherwise} \end{cases}$

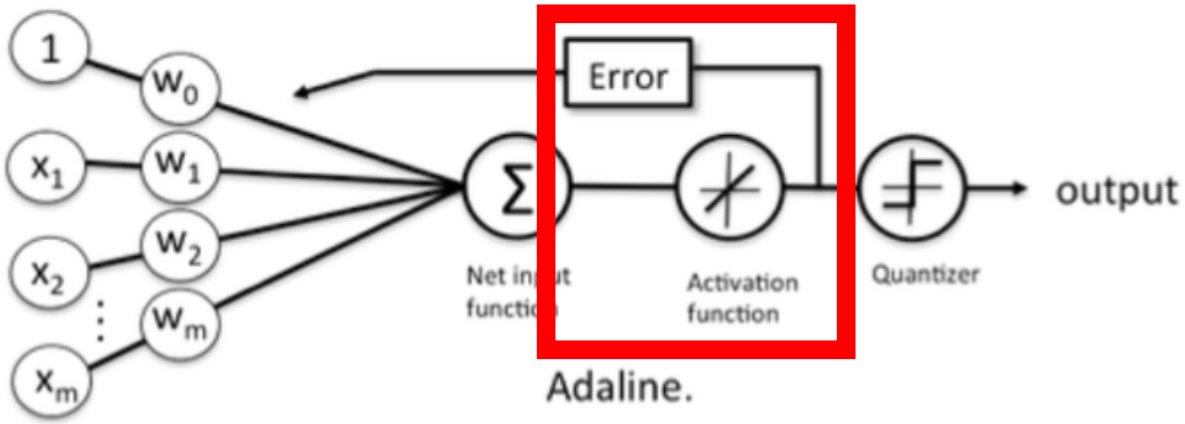


Python Machine Learning; Raschkka & Mirjalili

Adaline: Difference to Perceptron



Perceptron rule.



Adaline.

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

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

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: $w_j := w_j + \Delta w_j$

Key Idea: this is differentiable!!!

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

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

Mathematical Simplification

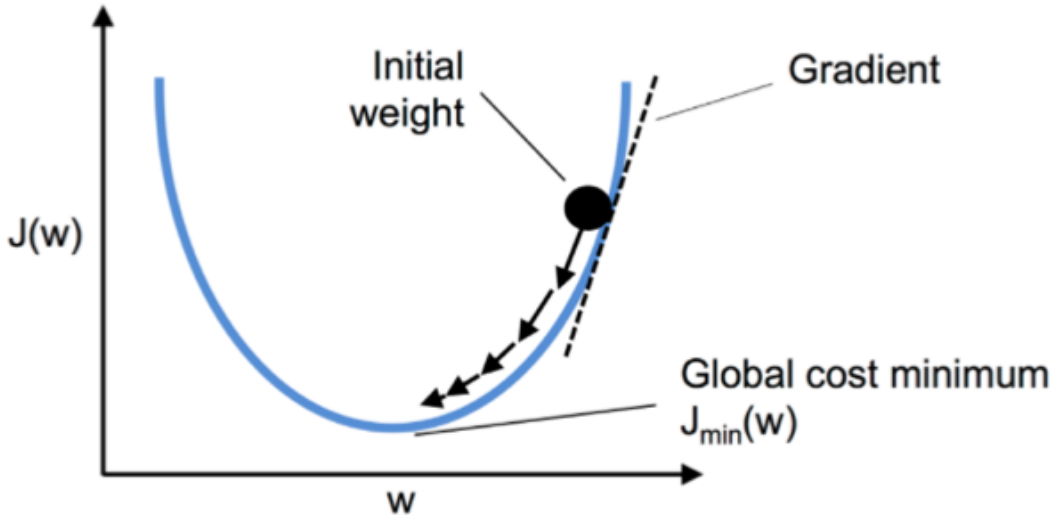
Sum of squared errors

Learning Rate

Take step away from gradient

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

Adaline Learning Algorithm



2. Update weights with accumulated "weight update" values: $w_j := w_j + \Delta w_j$

Key Idea: this is differentiable!!!

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

Mathematical Simplification

Sum of squared errors

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

Learning Rate

Take step away from gradient

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

Derivation of Equation to Update Weights

$$
\begin{aligned}
&\frac{\partial J}{\partial w_j} \\
&= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right)^2 \\
&= \frac{1}{2} \frac{\partial}{\partial w_j} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right)^2 \\
&= \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \phi(z)_A^{(i)} \right) \\
&= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \sum_i \left(w_j^{(i)} x_j^{(i)} \right) \right) \\
&= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right) (-x_j^{(i)}) \\
&= - \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right) x_j^{(i)}
\end{aligned}
$$

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

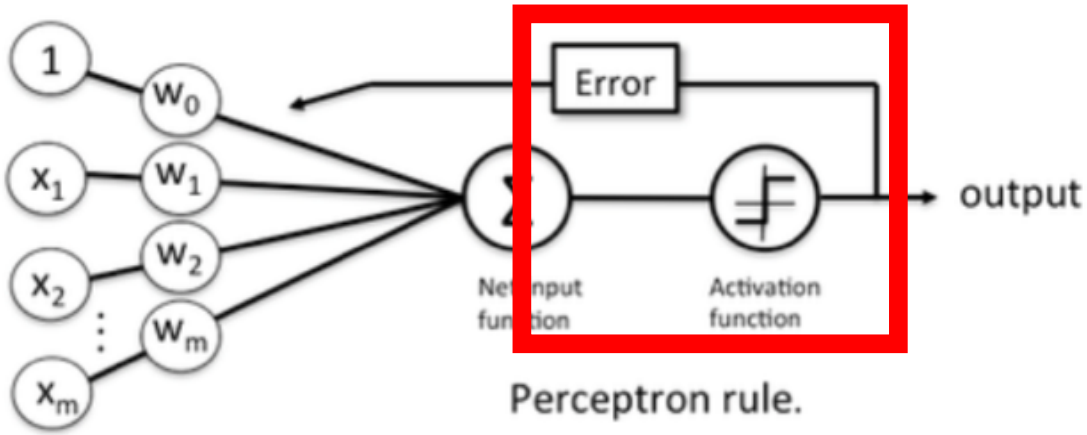
Adaline Learning Algorithm

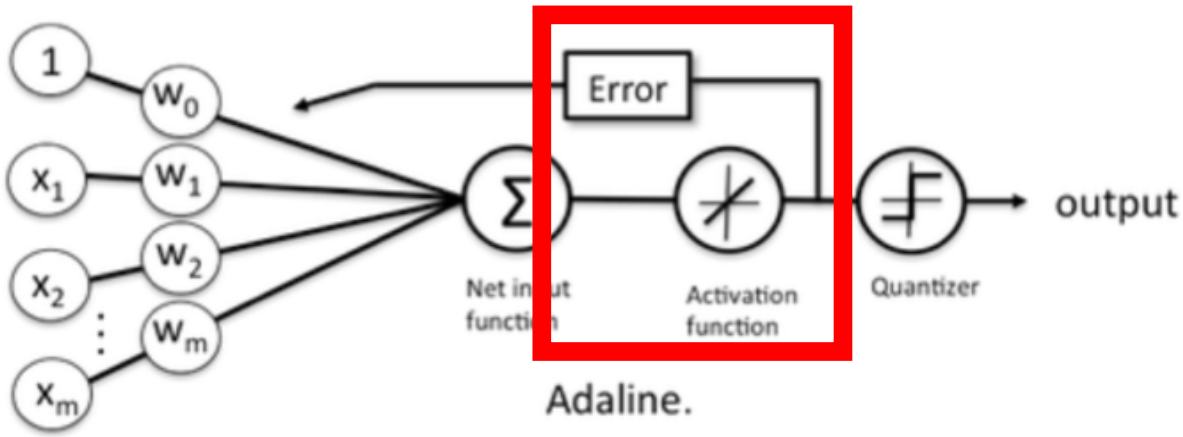
$$
\begin{aligned}
&\frac{\partial J}{\partial w_j} \\
&= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right)^2 \\
&= \frac{1}{2} \frac{\partial}{\partial w_j} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right)^2 \\
&= \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \phi(z)_A^{(i)} \right) \\
&= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right) \frac{\partial}{\partial w_j} \left(y^{(i)} - \sum_i \left(w_j^{(i)} x_j^{(i)} \right) \right) \\
&= \sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right) (-x_j^{(i)}) \\
&= -\sum_i \left(y^{(i)} - \phi(z)_A^{(i)} \right) x_j^{(i)}
\end{aligned}
$$

Updates based on continuous valued prediction!

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

Adaline: Difference to Perceptron



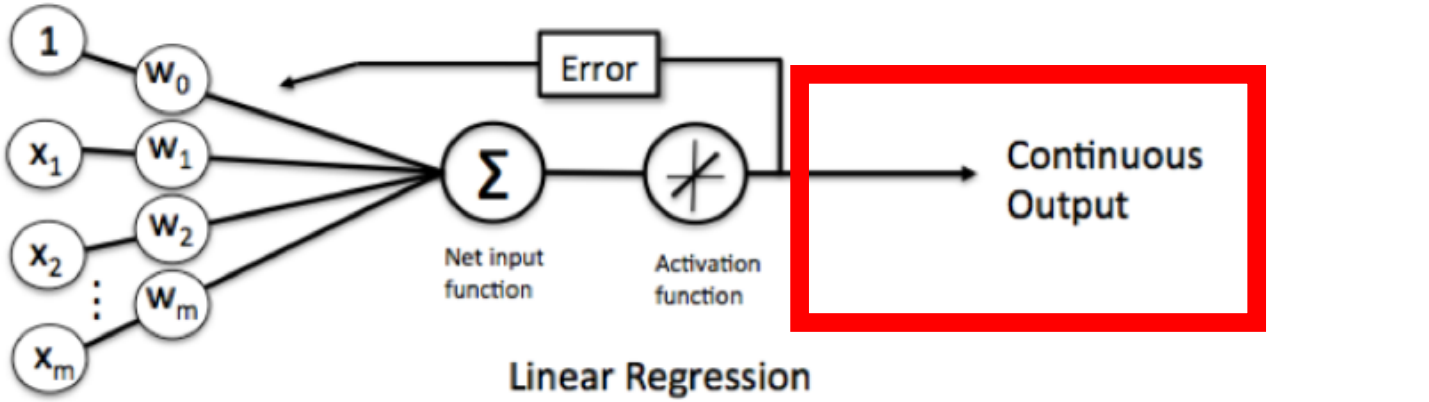


Updates based on continuous valued prediction rather than integer predictions!

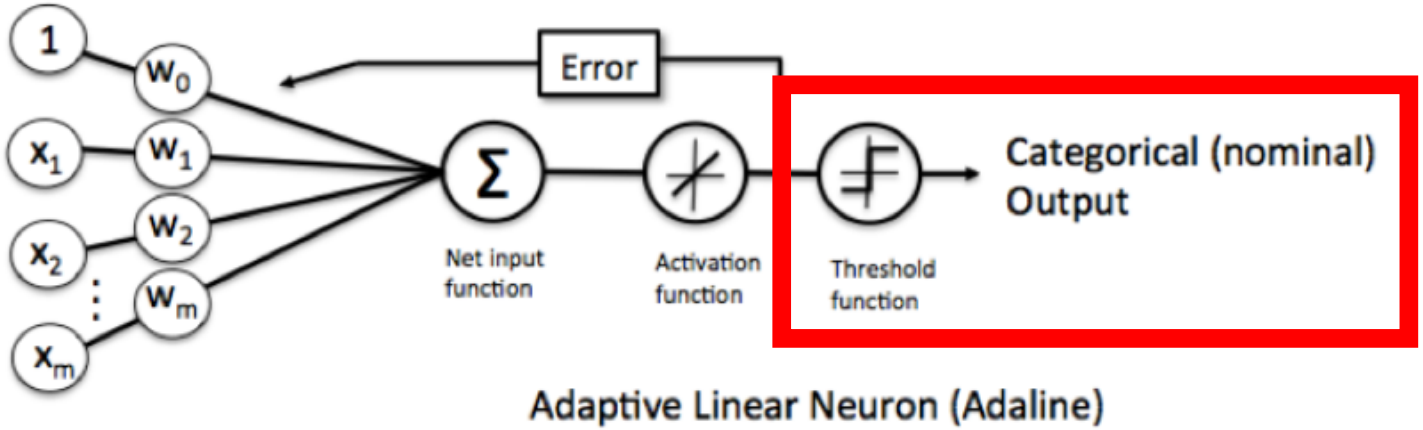
Python Machine Learning; Raschka & Mirjalili

Bernard Widrow and Ted Hoff, An Adaptive “Adaline” Neuron Using Chemical “Memistors”, 1960.

Adaline: Similarity and Difference to Linear Regression



Linear Regression



Adaptive Linear Neuron (Adaline)

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

Today's Topics

- History of Modern Neural Networks and "Deep Learning"
- Single Layer Neural Network: Perceptron
- Single Layer Neural Network: Adaline
- Variants of Gradient Descent
- Lab

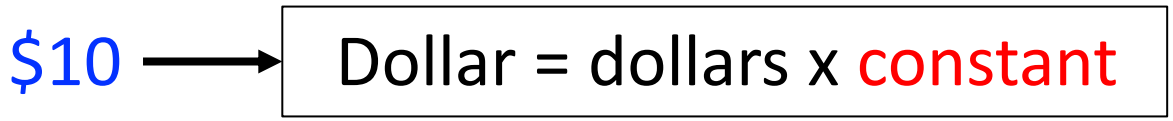
Gradient Descent Intuition

- 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 1$ to one yuan

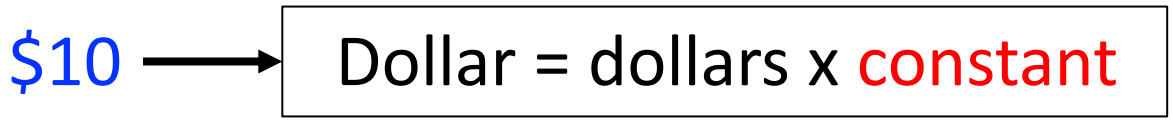


Gradient Descent Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert 1$ to one yuan

Gradient Descent Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert 1$ to one yuan

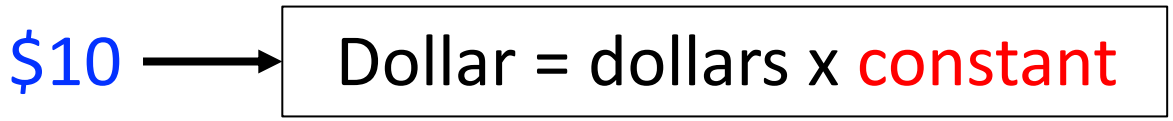


Gradient Descent Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert 1$ to one yuan

Gradient Descent Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert 1$ to one yuan

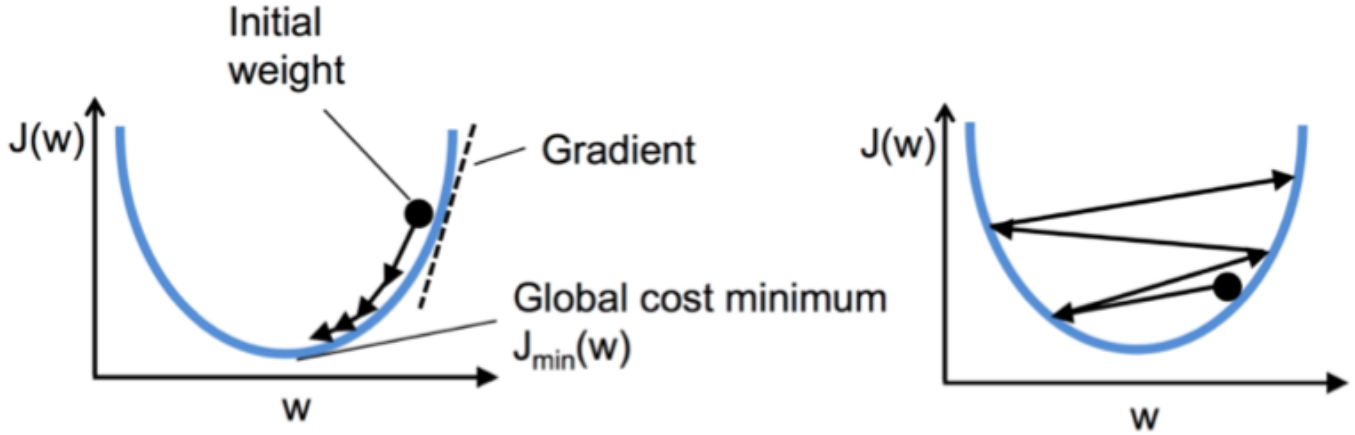


Gradient Descent Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert 1$ to one yuan

- Idea: iteratively adjust constant (i.e., model parameter) to try to make error get smaller

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

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
- Which algorithm uses this?
  - Perceptron

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

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

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

- History of Modern Neural Networks and "Deep Learning"
- Single Layer Neural Network: Perceptron
- Single Layer Neural Network: Adaline
- Variants of Gradient Descent
- Lab