

CS167: Machine Learning

Perceptron (continued)
Optimization for Weight Learning

Thursday, March 28th, 2024

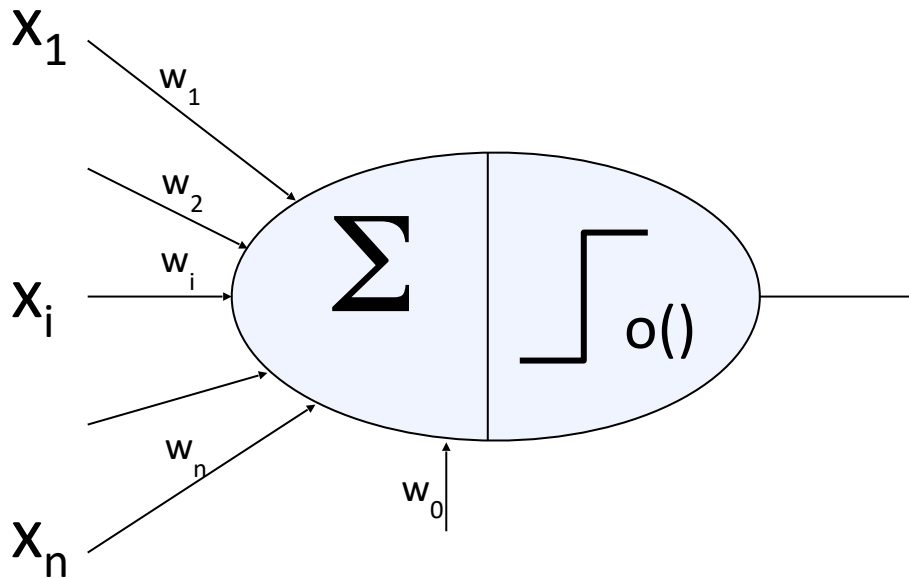


Perceptron

- The mathematical model of a *single neuron* is called a **perceptron**
- It has a two components:
 - Component 1: a linear model of the form we just saw in the previous slide

$$W_0 + W_1 * X_1 + \dots + W_n * X_n$$

- Component 2: a **step function** which will produce 1 if the function value is positive and -1 otherwise



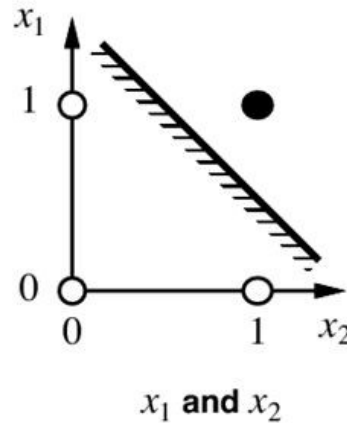
Big Question?
How do we determine the weights that best classify the data?

$$y = f(\mathbf{x}, \mathbf{w}) = o\left(\sum_{i=1, \dots, n} w_i x_i\right)$$

Perceptron can Model AND Function

- Let's consider the AND function.

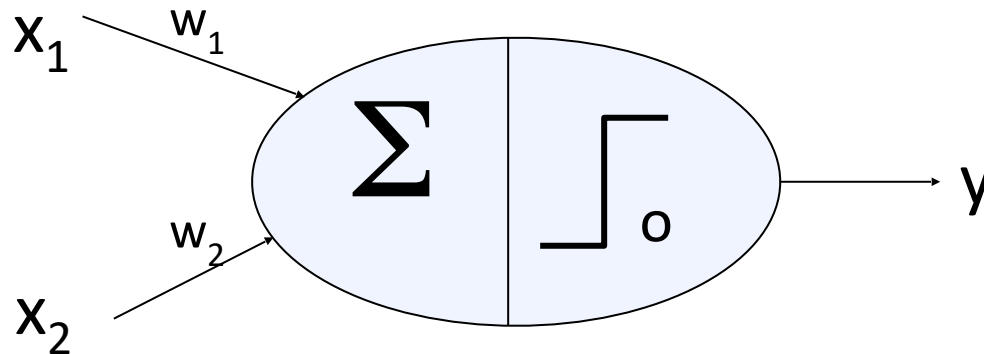
x	y	x and y
0	0	0
0	1	0
1	0	0
1	1	1



YES, because we can find this 2D line

Color code:
Empty circle denotes 0
Black circle denotes 1

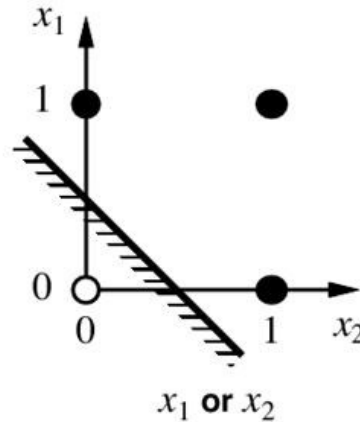
- Can a perceptron model AND function?



Perceptron can Model OR Function

- Let's consider the OR function.

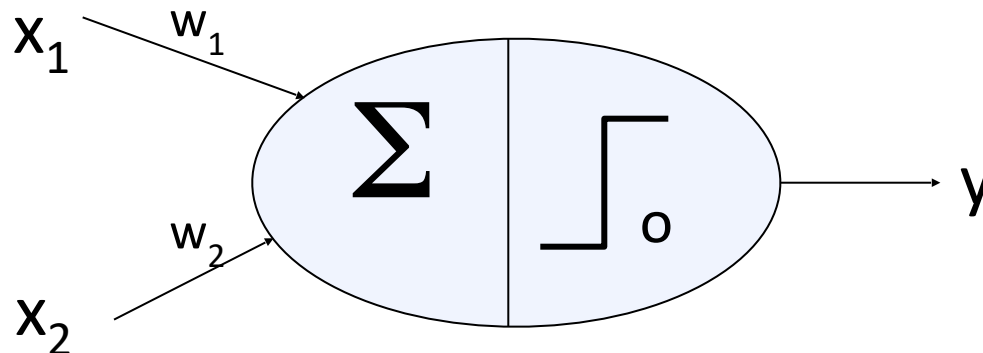
x	y	x or y
0	0	0
0	1	1
1	0	1
1	1	1



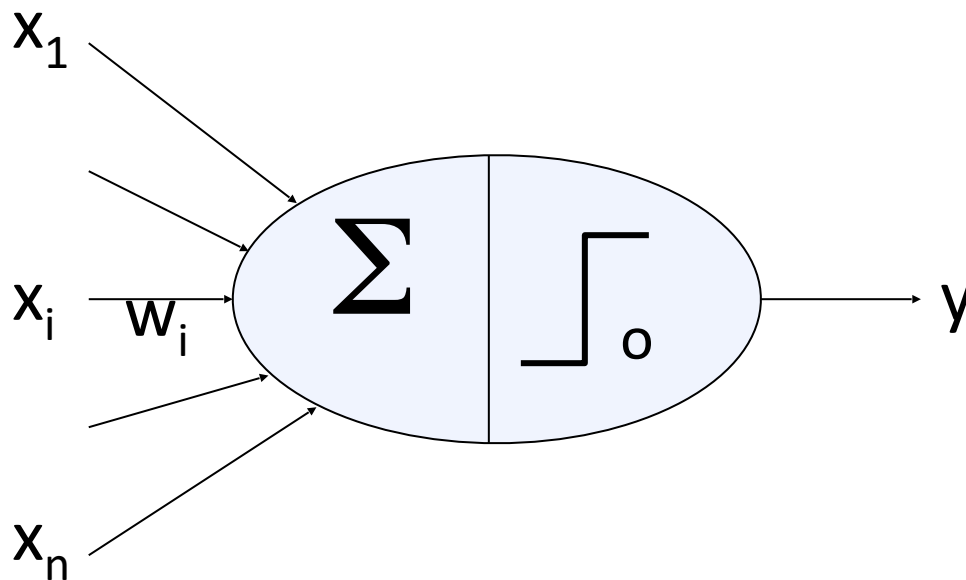
YES, because we can find this 2D line

Color code:
Empty circle denotes 0
Black circle denotes 1

- Can a perceptron model OR function?

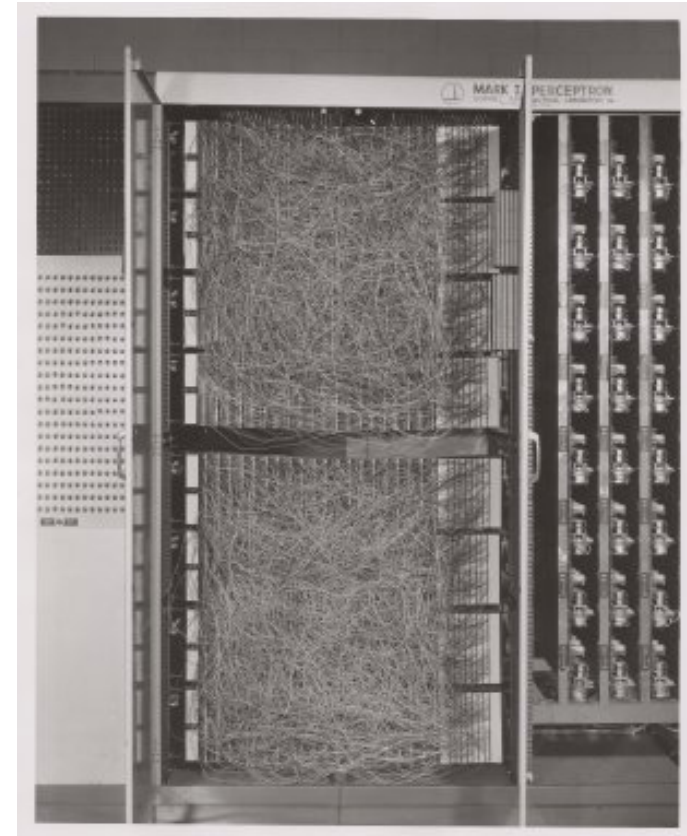
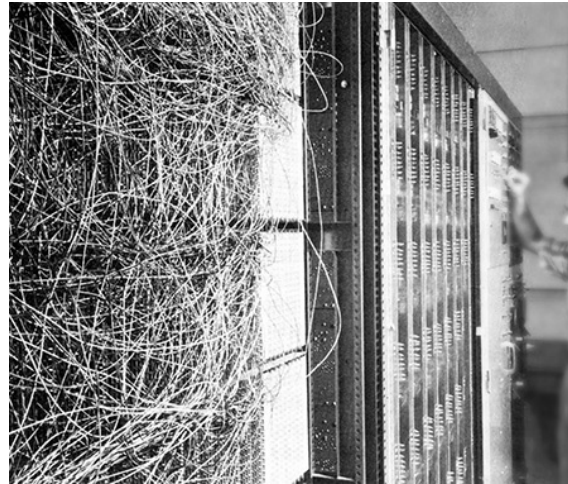


Can Perceptrons model any function?



- Perceptron can also model other boolean functions such as $(x_1 \wedge x_2 \wedge \neg x_3)$. We can tabulate all combinations of the three variables and their corresponding outputs; then find weight parameters (of the plane separating it two classes (1 and 0)) using perceptron update rule we just did
- But can a perceptron model any function?

Perceptron



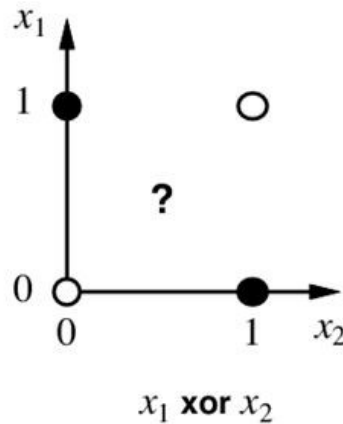
“the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”

Frank Rosenblatt, 1958

Now Let's Consider XOR Function

- Let's consider the XOR function.

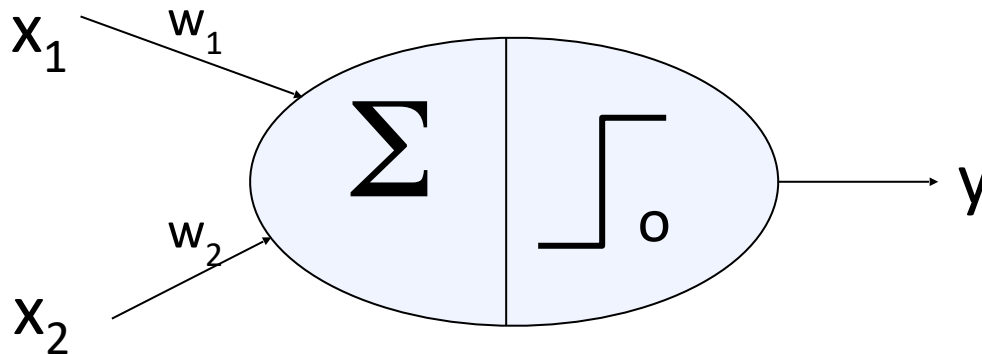
x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0



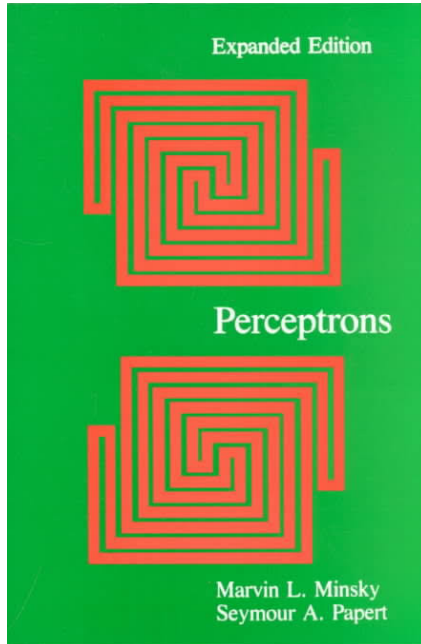
NO, because we can't find a single 2D line that separates the samples

Color code:
Empty circle denotes 0
Black circle denotes 1

- Can a perceptron model XOR function?

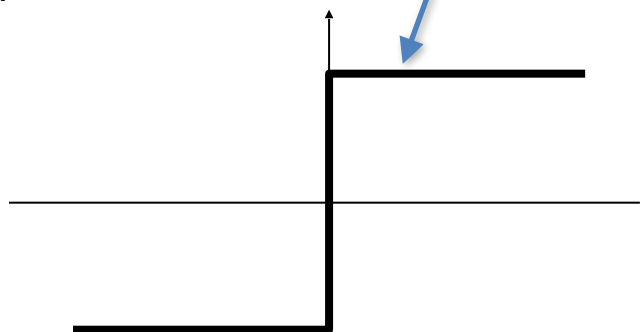
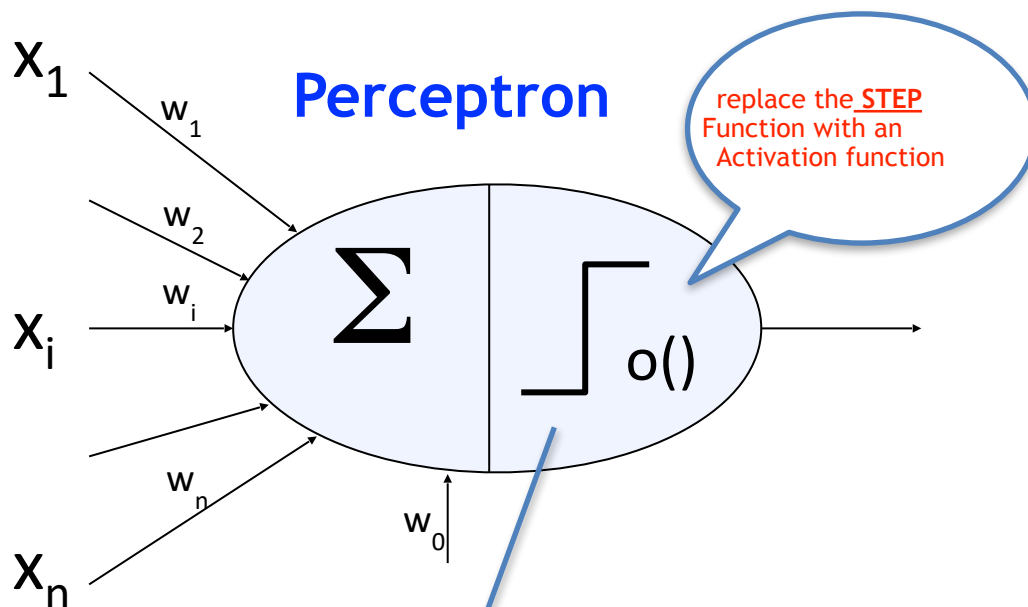


Perceptron



Marvin Minsky and Seymour Papert showed that they couldn't even learn XOR in 1969, which is why all the hype about the perceptron faded away

Modification of Perceptron

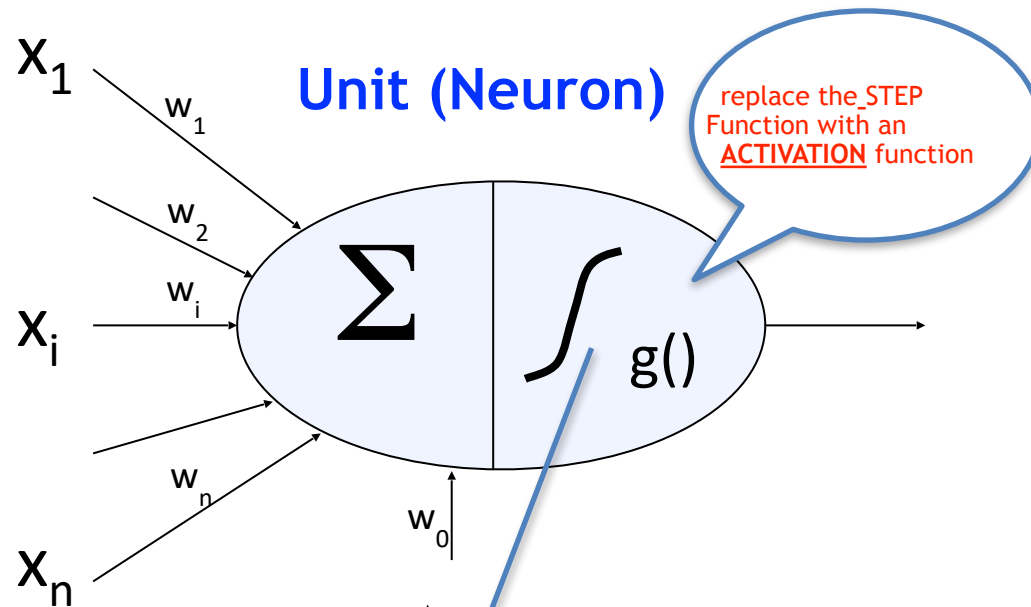


this step function outputs +1 if the function value is positive and -1 otherwise

It is not smooth and **not differentiable**

$$y = f(\mathbf{x}, \mathbf{w}) = o\left(\sum_{i=1, \dots, n} w_i x_i\right)$$

Make a Neuron with a Differentiable Function



this new function is referred to as an activation function. eg, sigmoid activation function

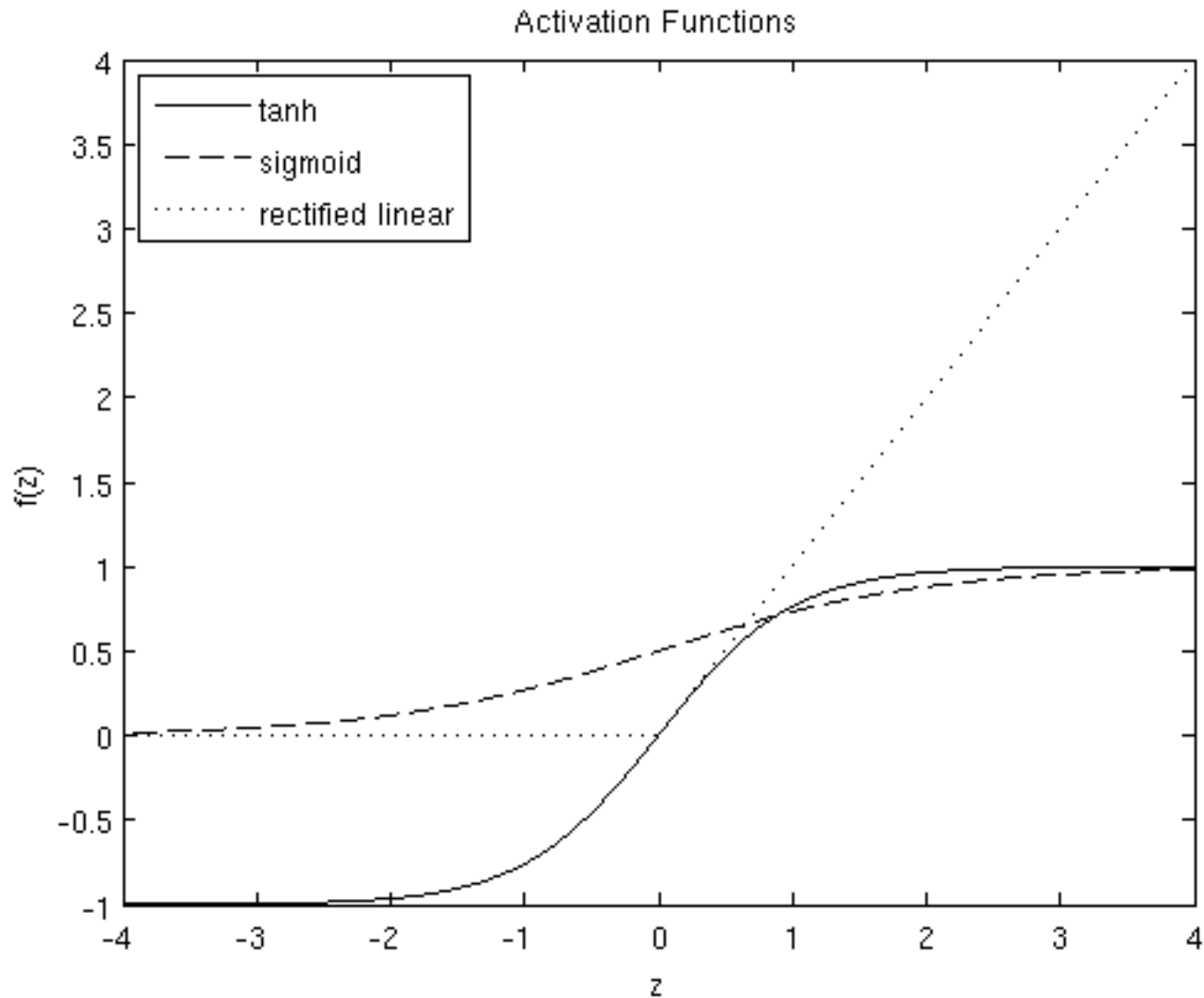
$$g\left(\sum_{i=1} w_i x_i\right) = \frac{1}{1 + \exp\left(-\sum_{i=1} w_i x_i\right)}$$

It is **smooth** and **differentiable**

$$y = f(\mathbf{x}, \mathbf{w}) = g\left(\sum_{i=1, \dots, n} w_i x_i\right)$$

Common Activation Functions

Each activation function is smooth and differentiable

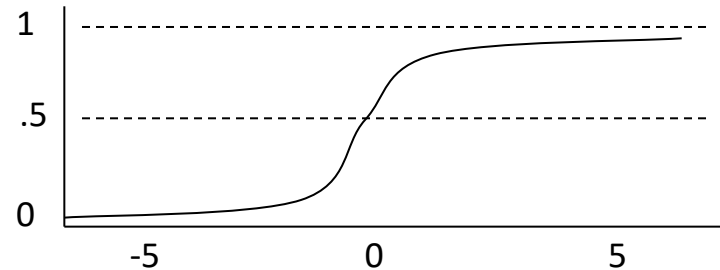


Sigmoid (a.k.a. Logistic Function) Activation

Sigmoid:

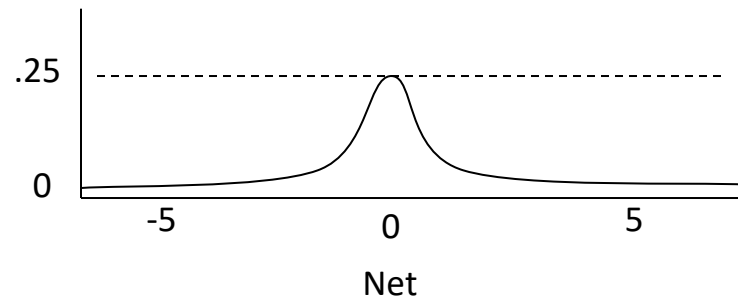
$$g(u) = \frac{1}{1 + \exp^{-u}}$$

It is **smooth** and **differentiable**



Differentiation of sigmoid:

$$\frac{dg(u)}{du} = g'(u) = g(u)(1 - g(u))$$



Today's Agenda

- Mathematical Modeling of Weight Parameters Learning
- Intuitive Understanding of Optimization (minimization/maximization)
- Finding Moving Direction in Loss Surface (Gradient Calculation)
- Gradient Descent
- Stochastic Gradient Descent (SGD)

Learning Weight Parameters with this Modified Neuron Model

- Instead of our simple **Perceptron Update Rule**, we can now use a better learning algorithm to learn the weight parameters ($w_0, w_1, w_2, \dots, w_n$)
- But what is this new weight parameter learning algorithm?

Learning Weight Parameters with this Modified Neuron Model

boston-housing dataset training features

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.10793	0.0	8.56	0	0.520	6.195	54.4	2.7778	5	384	20.9	13.00
1.34284	0.0	19.58	0	0.605	6.066	100.0	1.7573	5	403	14.7	6.43
4.81213	0.0	18.10	0	0.713	6.701	90.0	2.5975	24	666	20.2	16.42

$\mathbf{x}^i =$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

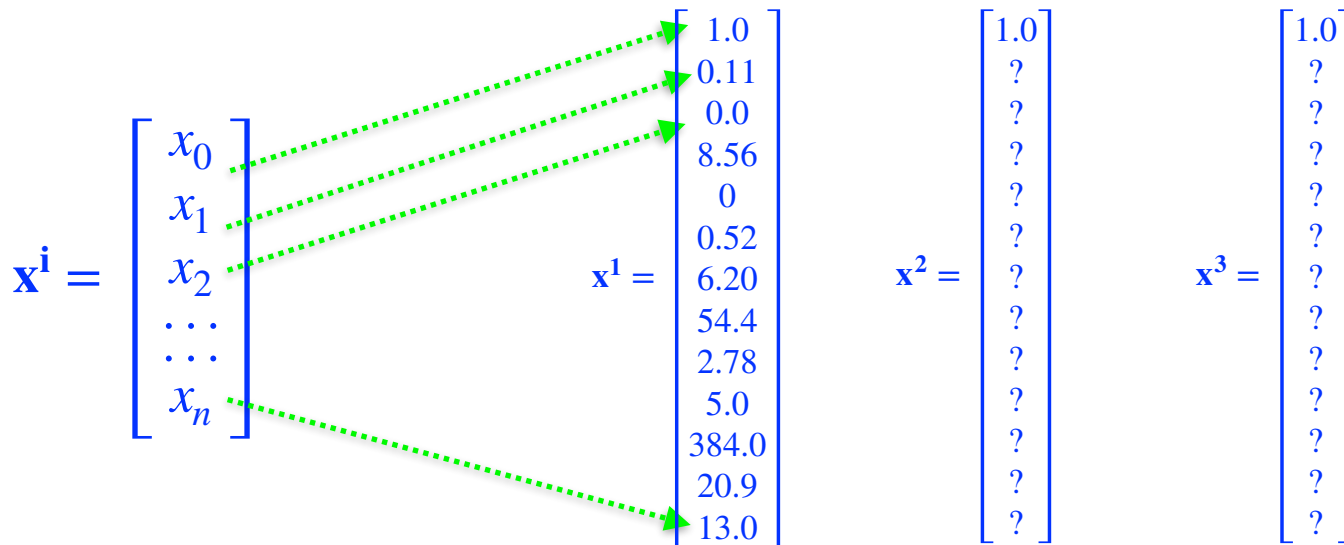
training labels

MEDV
21.7
24.3
16.4

y^i

Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



boston-housing dataset training split

take a training example

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.10793	0.0	8.56	0	0.520	6.195	54.4	2.7778	5	384	20.9	13.00
1.34284	0.0	19.58	0	0.605	6.066	100.0	1.7573	5	403	14.7	6.43
4.81213	0.0	18.10	0	0.713	6.701	90.0	2.5975	24	666	20.2	16.42

Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input

$$\mathbf{x}^i = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

$$\mathbf{x}^1 = \begin{bmatrix} 1.0 \\ 0.11 \\ 0.0 \\ 8.56 \\ 0 \\ 0.52 \\ 6.20 \\ 54.4 \\ 2.78 \\ 5.0 \\ 384.0 \\ 20.9 \\ 13.0 \end{bmatrix}$$

$$\mathbf{x}^2 = \begin{bmatrix} 1.0 \\ 1.34 \\ 0.0 \\ 19.58 \\ 0 \\ 0.61 \\ 6.07 \\ 100.0 \\ 1.76 \\ 5.0 \\ 403.0 \\ 14.7 \\ 6.43 \end{bmatrix}$$

$$\mathbf{x}^3 = \begin{bmatrix} 1.0 \\ 4.81 \\ 0.0 \\ 18.1 \\ 0 \\ 0.71 \\ 6.70 \\ 90.0 \\ 2.60 \\ 24.0 \\ 666.0 \\ 20.2 \\ 16.42 \end{bmatrix}$$

take a training
example

boston-housing dataset training split

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.10793	0.0	8.56	0	0.520	6.195	54.4	2.7778	5	384	20.9	13.00
1.34284	0.0	19.58	0	0.605	6.066	100.0	1.7573	5	403	14.7	6.43
4.81213	0.0	18.10	0	0.713	6.701	90.0	2.5975	24	666	20.2	16.42

Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input

$$\mathbf{x}^i = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

$$\mathbf{x}^1 = \begin{bmatrix} 1.0 \\ 0.11 \\ 0.0 \\ 8.56 \\ 0 \\ 0.52 \\ 6.20 \\ 54.4 \\ 2.78 \\ 5.0 \\ 384.0 \\ 20.9 \\ 13.0 \end{bmatrix}$$

$$\mathbf{x}^2 = \begin{bmatrix} 1.0 \\ 1.34 \\ 0.0 \\ 19.58 \\ 0 \\ 0.61 \\ 6.07 \\ 100.0 \\ 1.76 \\ 5.0 \\ 403.0 \\ 14.7 \\ 6.43 \end{bmatrix}$$

$$\mathbf{x}^3 = \begin{bmatrix} 1.0 \\ 4.81 \\ 0.0 \\ 18.1 \\ 0 \\ 0.71 \\ 6.70 \\ 90.0 \\ 2.60 \\ 24.0 \\ 666.0 \\ 20.2 \\ 16.42 \end{bmatrix}$$

y^i

take a training
ground truth labels

training labels

MEDV
21.7
24.3
16.4

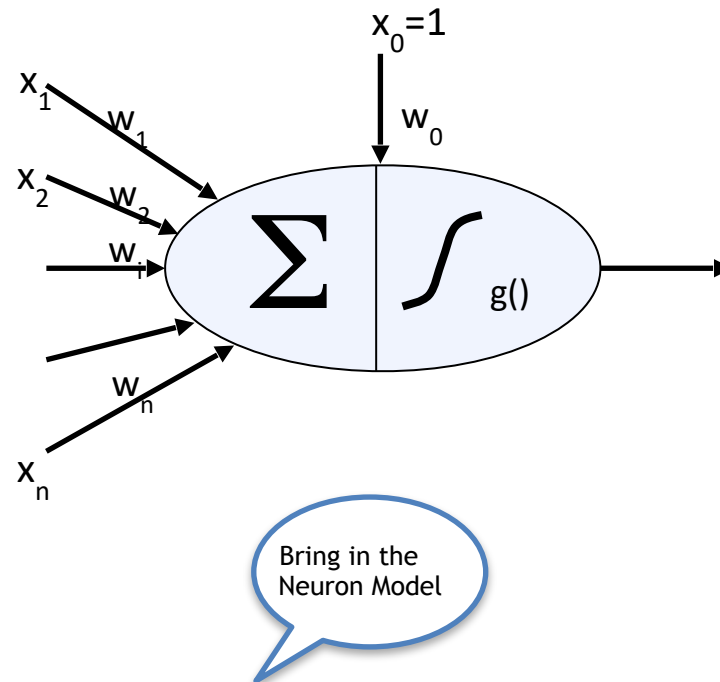
$$y^1 = 21.7$$

$$y^2 = 24.3$$

$$y^3 = 16.4$$

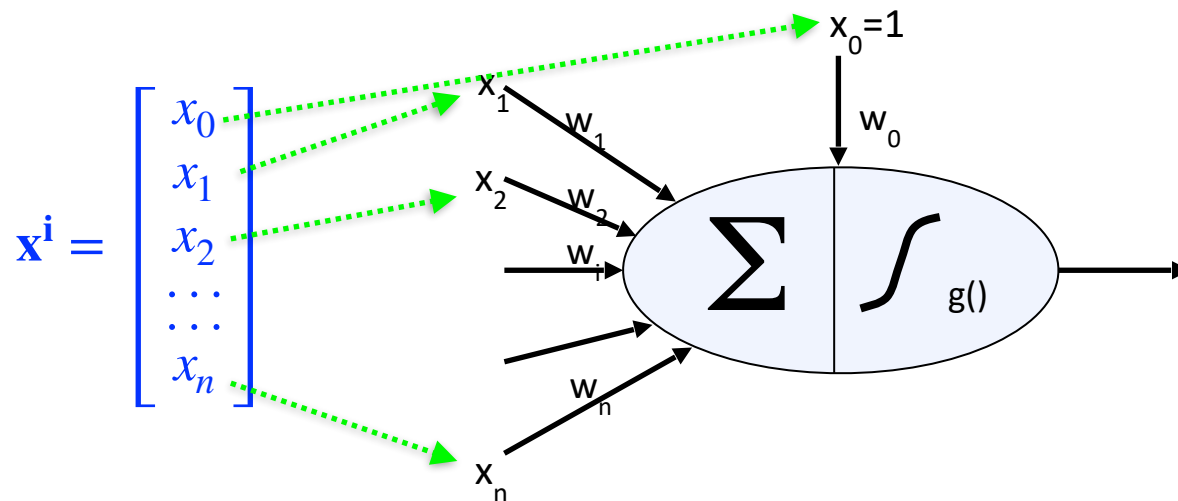
Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



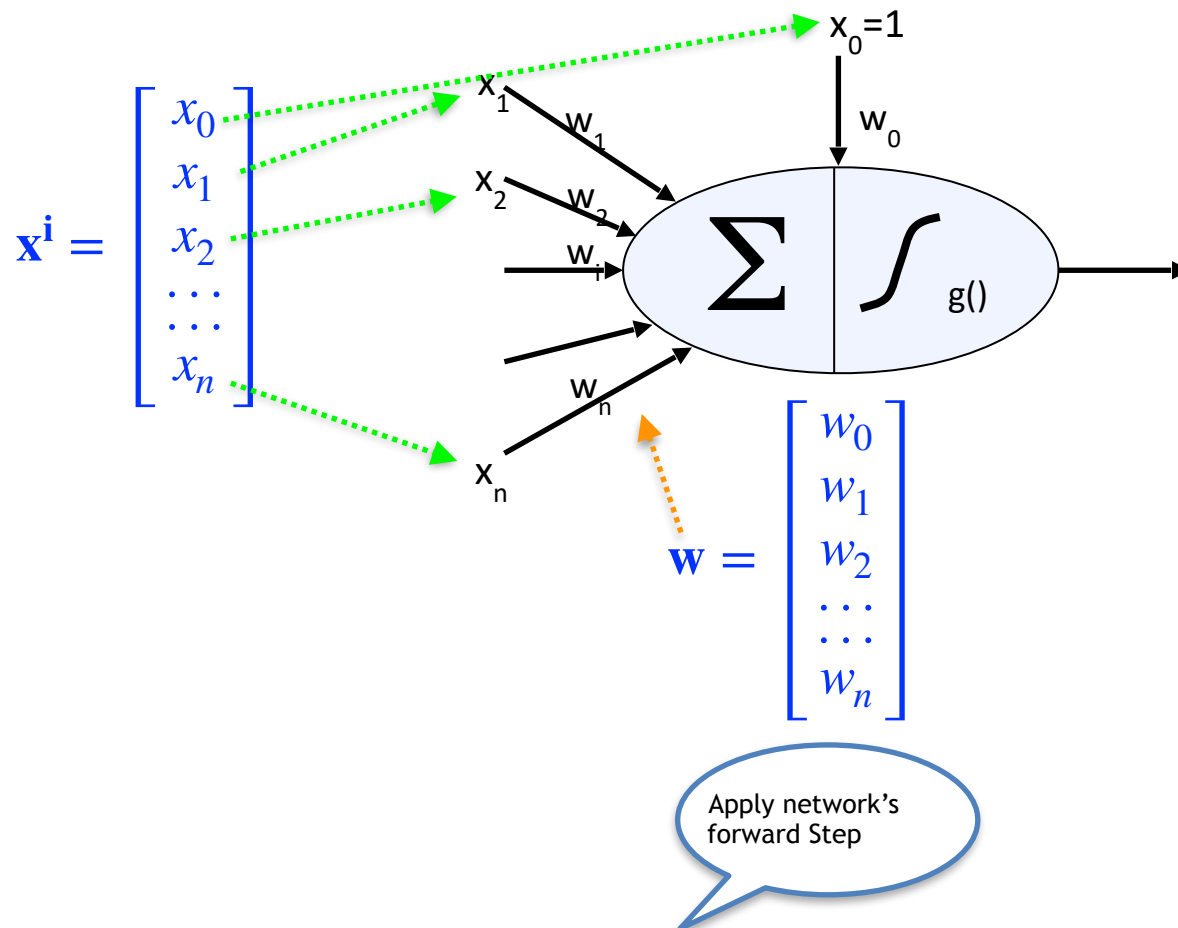
Plug-in a training example

boston-housing dataset training split

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.10793	0.0	8.56	0	0.520	6.195	54.4	2.7778	5	384	20.9	13.00
1.34284	0.0	19.58	0	0.605	6.066	100.0	1.7573	5	403	14.7	6.43
4.81213	0.0	18.10	0	0.713	6.701	90.0	2.5975	24	666	20.2	16.42

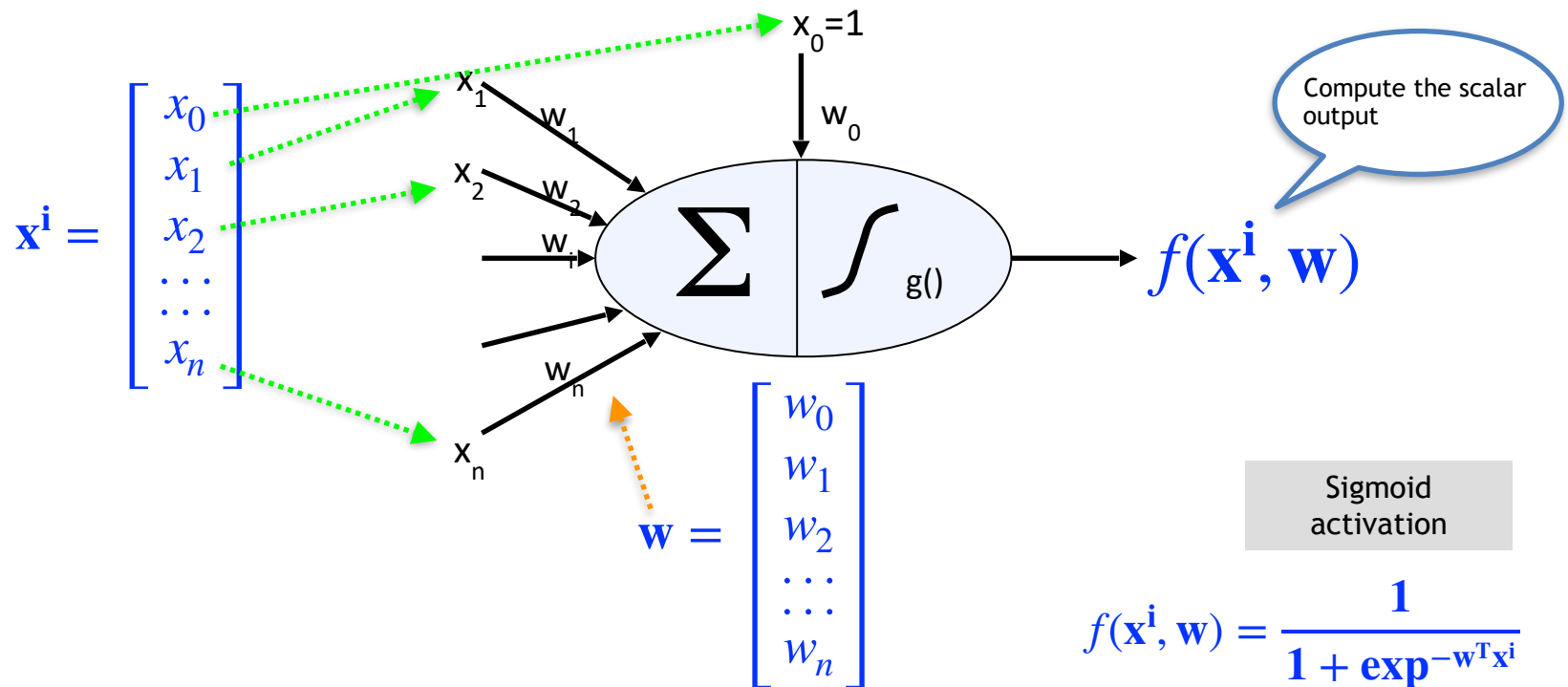
Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



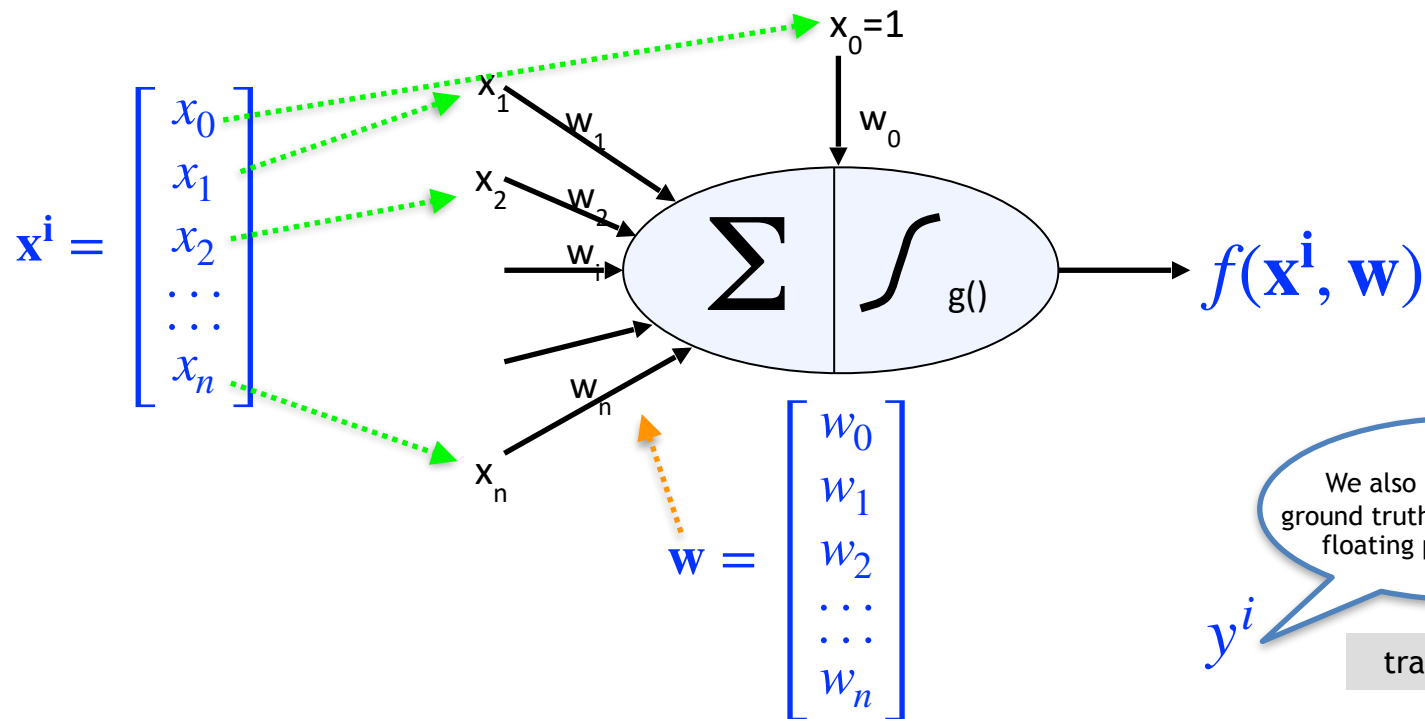
Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



training labels

MEDV

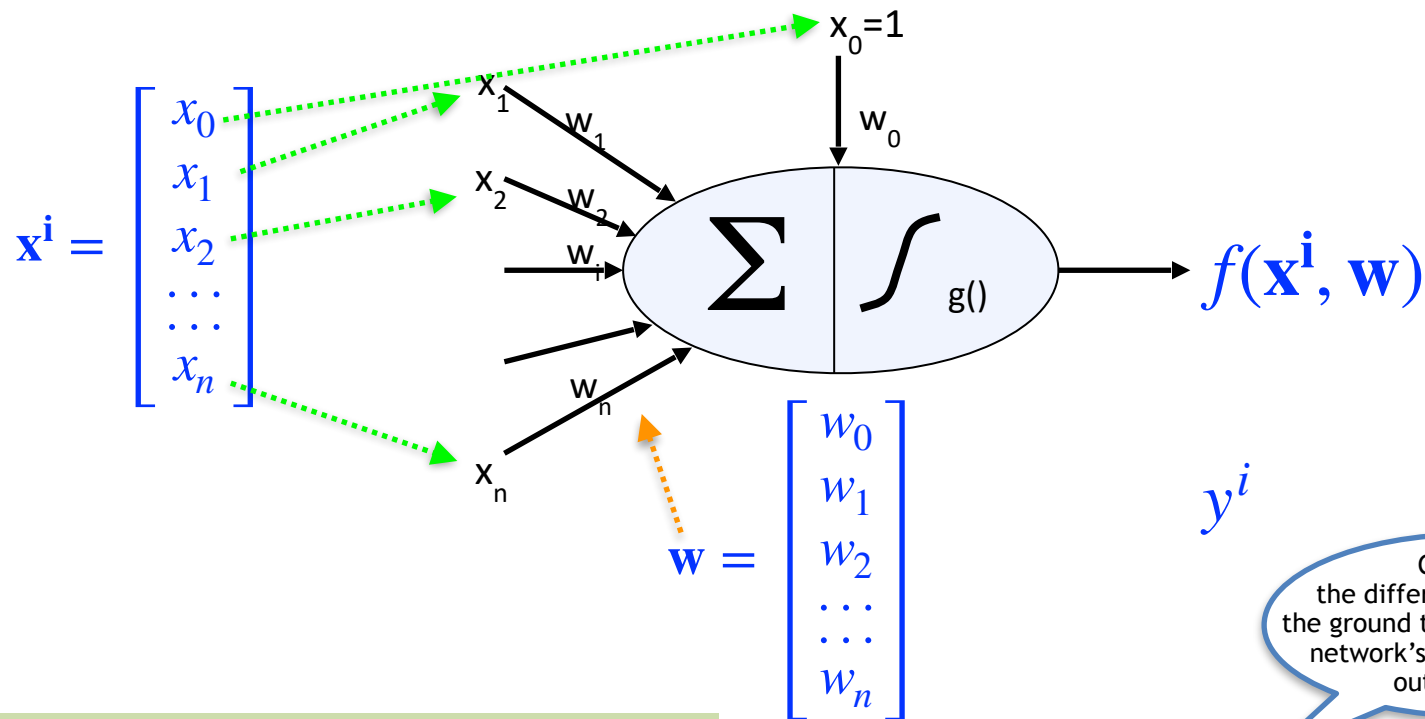
21.7

24.3

16.4

Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input

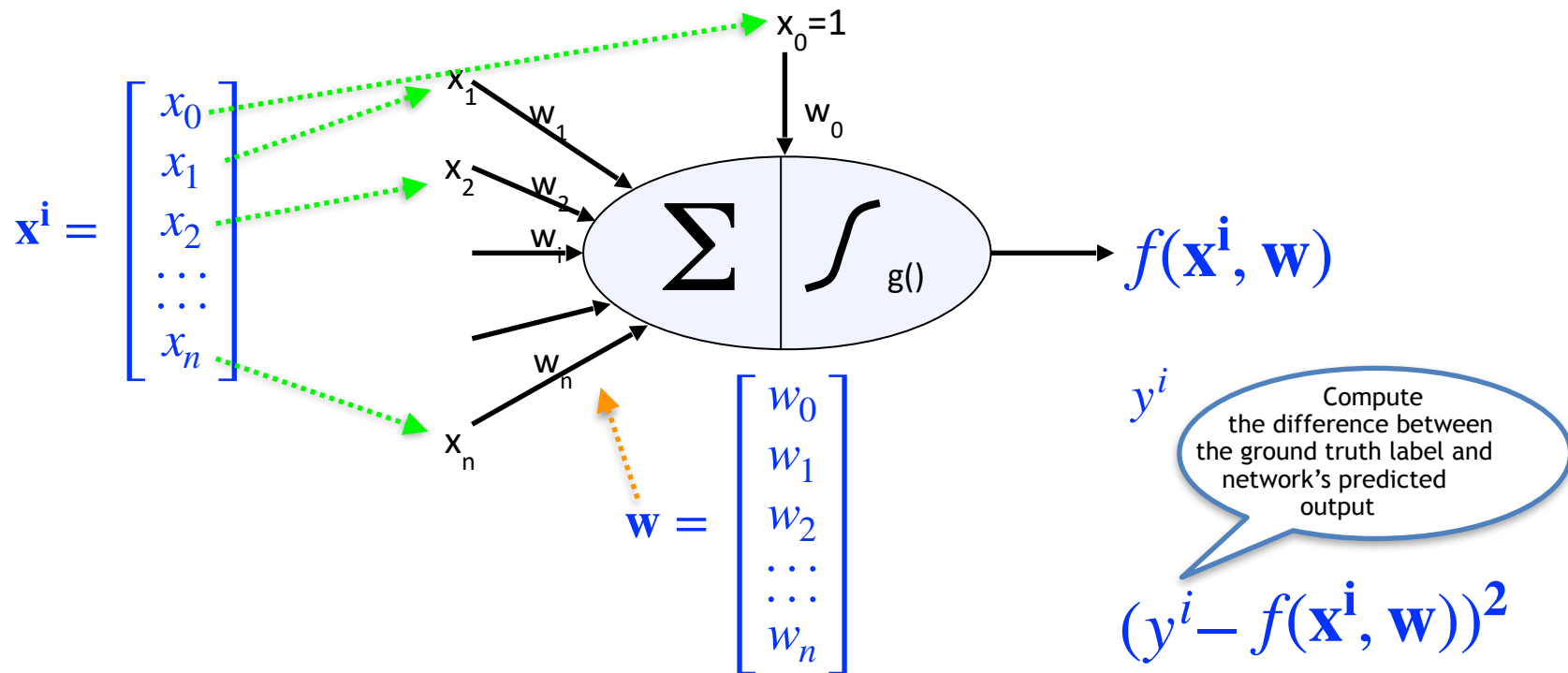


For example, here, we used **Mean Squared Error (MSE)** to measure the discrepancy. We could use any other measure to find the discrepancy between prediction and ground-truth

$$(y^i - f(\mathbf{x}^i, \mathbf{w}))^2$$

Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



$$Error(\mathbf{x}^i, y^i, \mathbf{w}) = (y^i - f(\mathbf{x}^i, \mathbf{w}))^2$$

Learning Weight Parameters with this Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input

$$Error(\mathbf{x}^i, y^i, \mathbf{w}) = (y^i - f(\mathbf{x}^i, \mathbf{w}))^2$$

- If we consider a collection of training examples and sum the above error over all examples

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x}^i, y^i, \mathbf{w}) = \sum_i (y^i - f(\mathbf{x}^i, \mathbf{w}))^2$$

Learning Weight Parameters with this Modified Neuron Model

- If we consider a collection of training examples and Sum the above error term over all examples

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x}_i, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

- Minimize errors using an optimization algorithm:
 - Gradient Descent (GD)
 - Stochastic Gradient Descent (SGD)

$E(\mathbf{w})$ term is also known as *loss function*

Today's Agenda

- Mathematical Modeling of Weight Parameters Learning
- Intuitive Understanding of Optimization (minimization/maximization)
- Finding Moving Direction in Loss Surface (Gradient Calculation)
- Gradient Descent
- Stochastic Gradient Descent (SGD)

Optimization

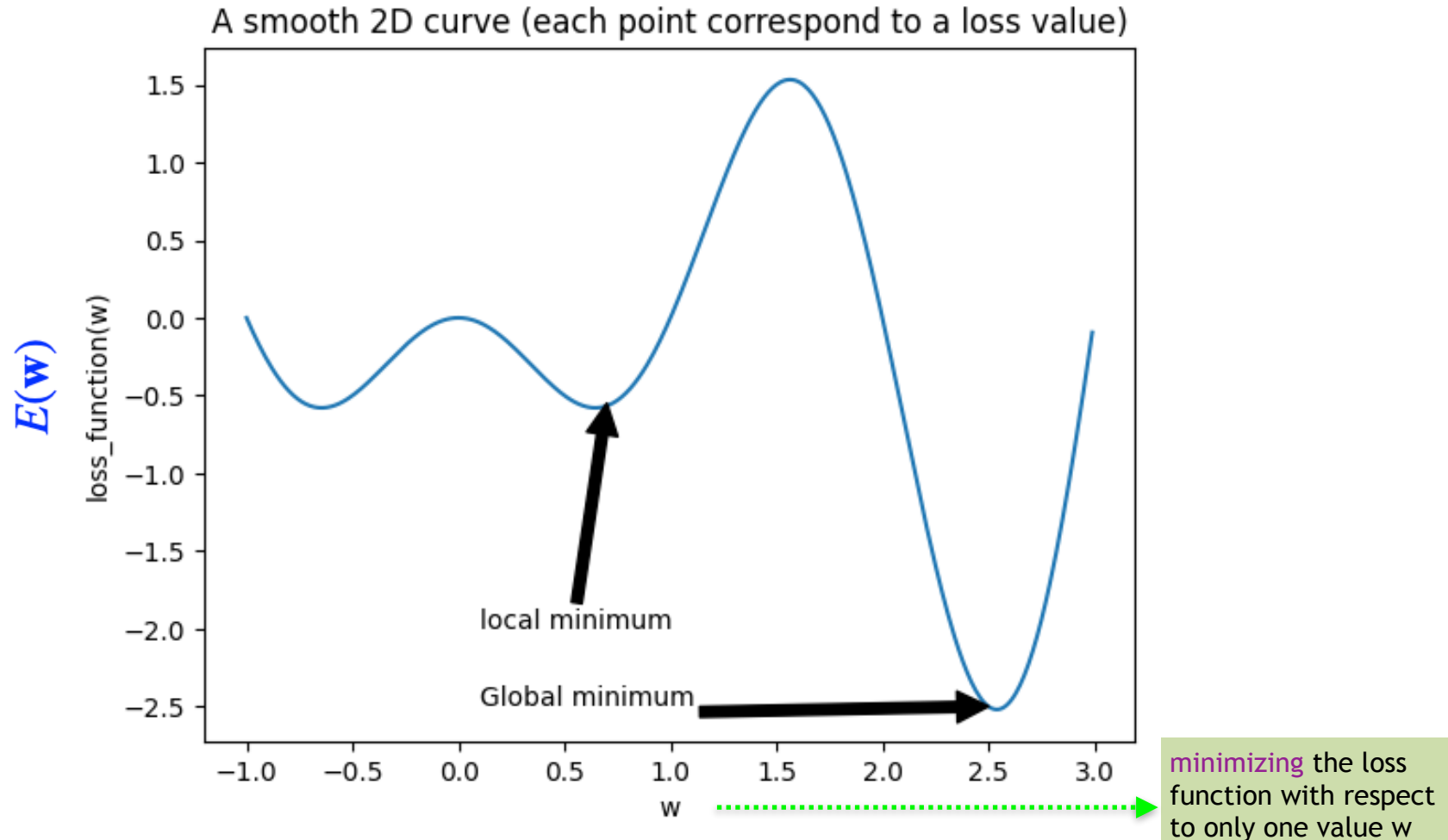
- Mathematical **optimization** is the process of selecting the "*best element*" with regard to some criterion from some set of available alternatives
- Optimization problems come in two flavors:
 - **minimization**: trying to find the subset of values for attributes that gives you the **minimum** value in the objective function
 - **maximization**: trying to find the subset of values for attributes that gives you the **maximum** value in the objective function

Optimization

- **minimization**: trying to find the subset of values for attributes that gives you the minimum value in the objective function
- The term objective function is generalized term which leaves room for the function to be something that we want to either **minimize** or **maximize**. The other terms used for the minimizing setting are as follows:
 - loss function
 - error function
 - cost function

Optimization Intuition

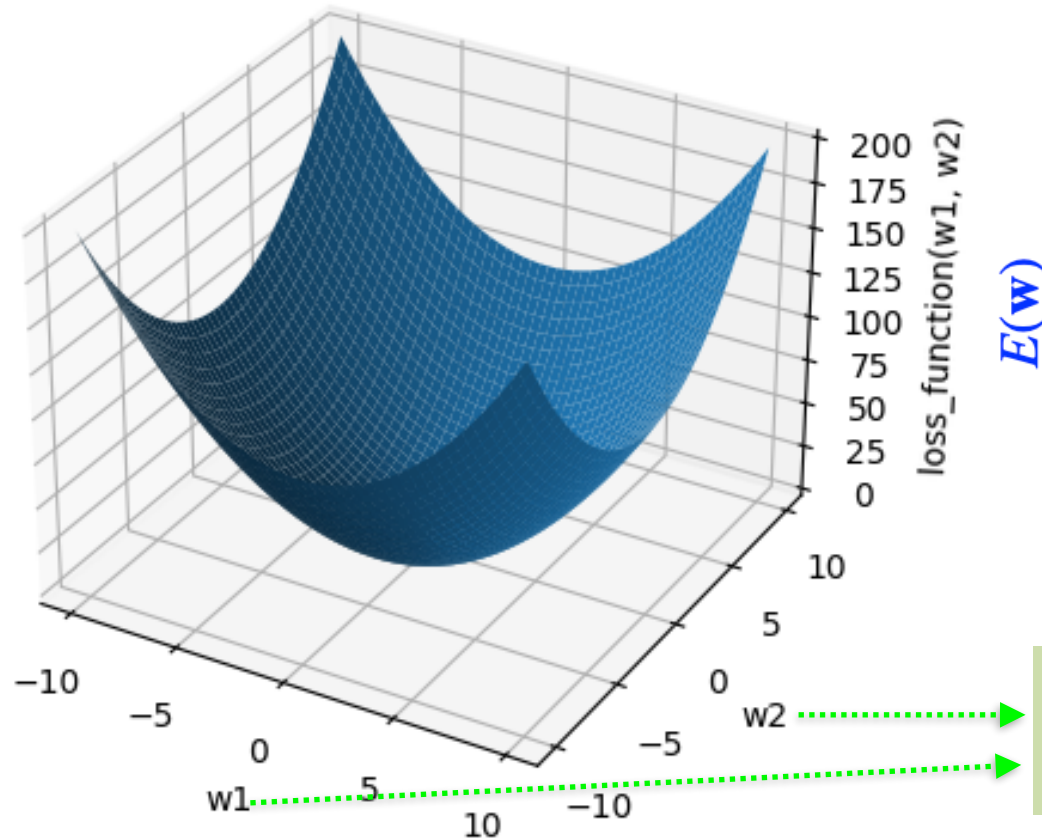
- **minimization**: trying to find the subset of values for attributes that gives you the minimum value in the objective function



Optimization Intuition

- **minimization**: trying to find the subset of values for attributes that gives you the minimum value in the objective function

A smooth 3D surface (each point correspond to a loss value)

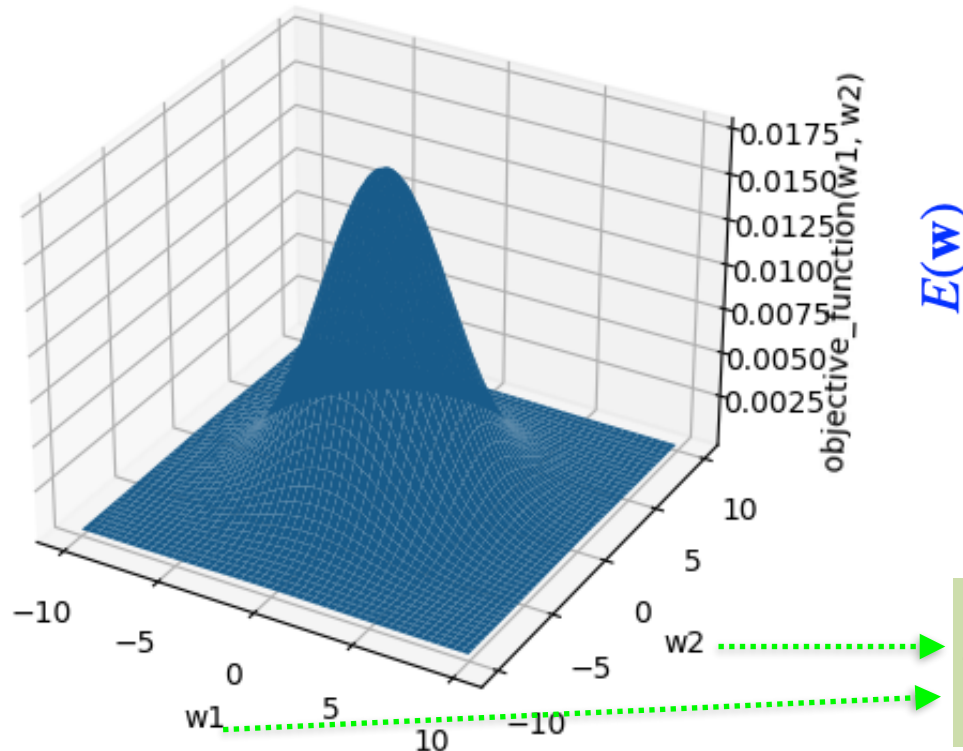


Play with the code I uploaded on Blackboard to generates different surfaces

Optimization Intuition

- **maximizations**: trying to find the subset of values for attributes that gives you the maximum value in the objective function

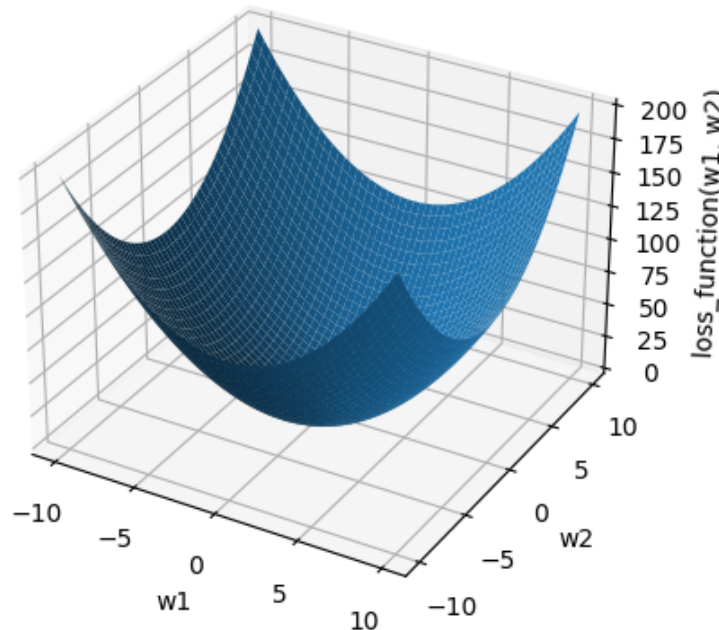
A smooth 3D surface (each point correspond to a value of the objective function)



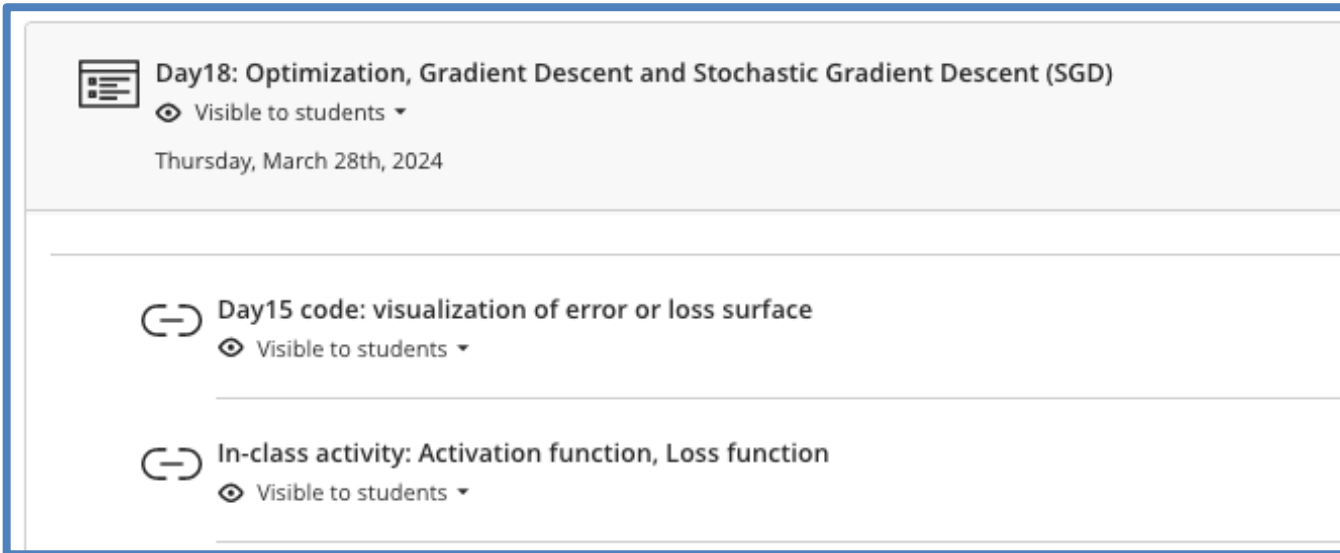
Optimization Intuition

- **minimization**: trying to find the subset of values for attributes that gives you the minimum value in the objective function
- **How to reach to the minimum?**
 - we can start at an arbitrary point on the surface and gradually explore the surface until we reach the minimum value

A smooth 3D surface (each point correspond to a loss value)



In-class Activity



The screenshot shows a course page with the following content:

- Day18: Optimization, Gradient Descent and Stochastic Gradient Descent (SGD)**
 - Visible to students
 - Thursday, March 28th, 2024
- Day15 code: visualization of error or loss surface**
 - Visible to students
- In-class activity: Activation function, Loss function**
 - Visible to students

- Explore the code first
- Participate in this poll

<https://forms.gle/U9Wa8smjrTsC5JMf6>