CS167: Machine Learning

Perceptron (group activity)
Discriminative vs. Generative Models
Generalizing Weight Learning Algorithm

Monday, October 28th, 2024

Drake
UNIVERSITY

Announcements

- Project #1
  - released on October 21st
  - due by November 04

- Quiz # 2
  - released today October 28th (Monday)
    - Topics: Weighted kNN, z-score normalization, PCA, linear classifier, perceptron learning algorithm, entropy calculation, information gain
  - due on November 6th (Wednesday)

Today's Agenda

- Perceptron Learning Algorithm Group Activity

Your Turn: Iteration 3: Apply Forward Step

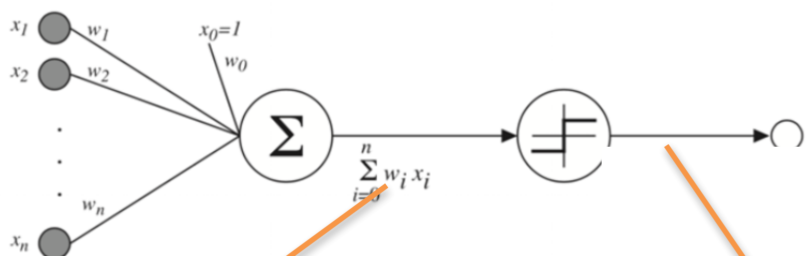
w_0^{old}= 0.0

w_1^{old}= 0.14

w_2^{old}= -0.3

η = 0.1

Predictor₁ Body Wave Mag X_1	Predictor₂ Surface Wave Mag x_2	Classification	Target Target (-1=earthquake, +1=explosion)
6.1	5.8	Earthquake	-1



Constant	Body wave mag	Surface wave mag	Neuron's linear model output	Neuron's step function output: *o*	target: *t*
x_0	x_1	x_2	$w_0x_0 + w_1x_1 + w_2x_2$	$o = \begin{cases} +1 & : w_0x_0 + w_1x_1 + w_2x_2 > 0 \\ -1 & : w_0x_0 + w_1x_1 + w_2x_2 \leq 0 \end{cases}$	explosion vs. earthquake
1	6.1	5.8	?	?	-1

Your Turn: Iteration 3: Backward Step

w_0^{old}= 0.0

w_1^{old}= 0.14

w_2^{old}= -0.3

η = 0.1

Body Wave Mag X_1 (Predictor₁)	Surface Wave Mag x_2 (Predictor₂)	Classification	Target (-1=earthquake, +1=explosion) (Target)
6.1	5.8	Earthquake	-1

$$\Delta w_i = \eta(t - o)x_i$$

$$w_i^{new} = w_i^{old} + \Delta w_i$$

computation for weight parameter W_0

η	target: *t*	Neuron's output: *o*	x_0	$\Delta w_0 = \eta(t-o)x_0$	$w_o^{new} = w_o^{old} + \Delta w_0$
0.1	-1	?	1	?	?

computation for weight parameter W_1

η	target: *t*	Neuron's output: *o*	x_1	$\Delta w_1 = \eta(t-o)x_1$	$w_1^{new} = w_1^{old} + \Delta w_1$
0.1	-1	?	6.1	?	?

computation for weight parameter W_2

η	target: *t*	Neuron's output: *o*	x_2	$\Delta w_2 = \eta(t-o)x_2$	$w_2^{new} = w_2^{old} + \Delta w_2$
0.1	-1	?	5.8	?	?

Your Turn: Results After Iteration 3

Predictor₁ Body Wave Mag X_1	Predictor₂ Surface Wave Mag x_2	Classification	Target Target (-1=earthquake, +1=explosion)
6.1	5.8	Earthquake	-1

updated weight parameter w_0 — w_0 = ?

updated weight parameter w_1 — w_1 = ?

updated weight parameter w_2 — w_2 = ?

fixed value for learning rate parameter — $\eta = 0.1$

- Using the third training example, we applied the perceptron learning rule and found the new weight parameters

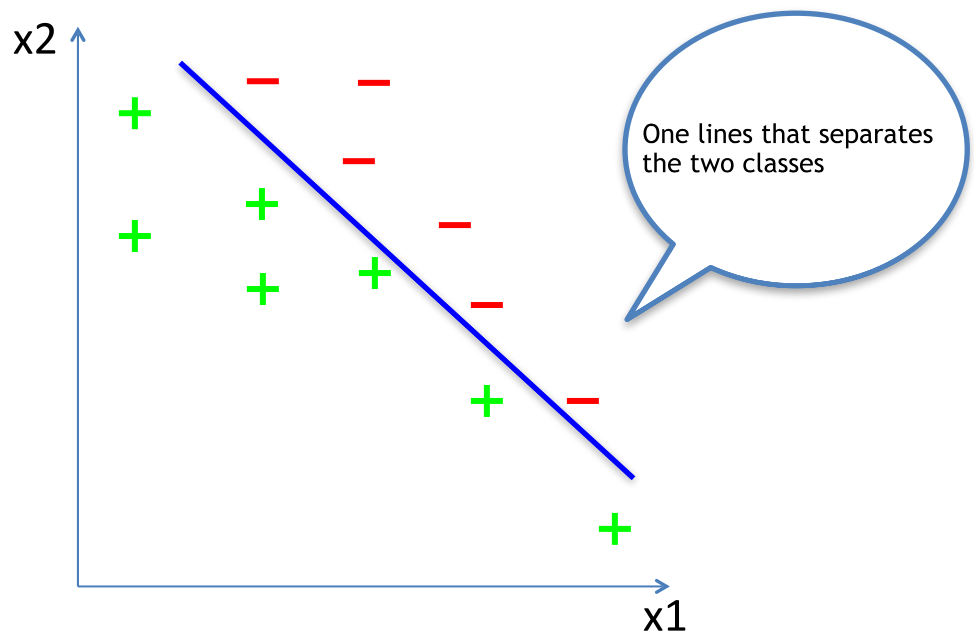
Today’s Agenda

- Perceptron Learning Algorithm Group Activity

- Generative Model vs Discriminative Model

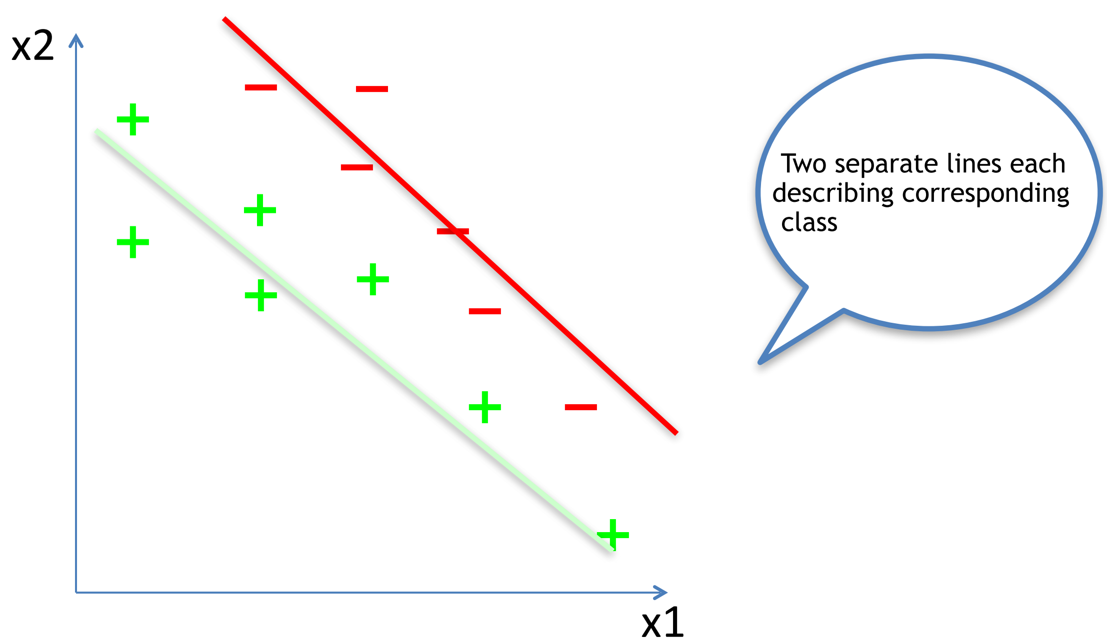
Discriminative Model for Classification

- **Discriminative Model:** Alternatively, we can build the decision functions (eg, line/plane/hyperplane) from the training samples using the ***difference* between two classes**
  - For example, if we have two classes as shown below, we model one 2D line for class-1 examples and another 2D line for class-2 examples



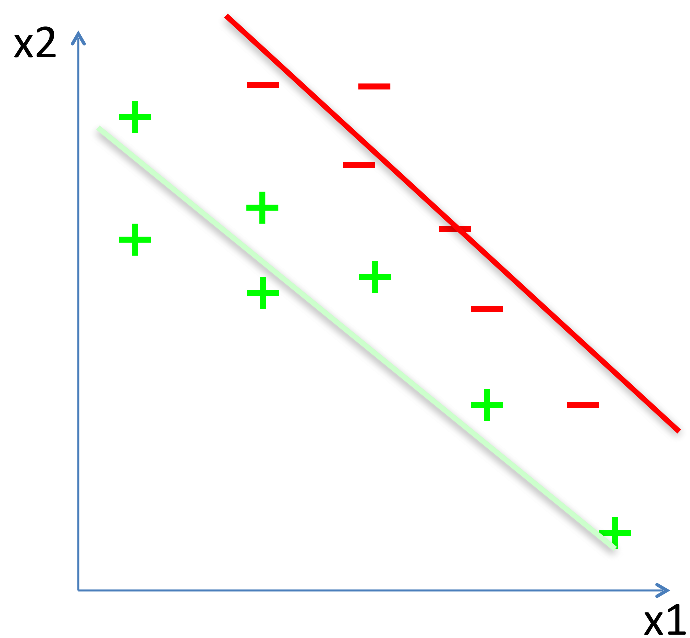
Generative Model for Classification

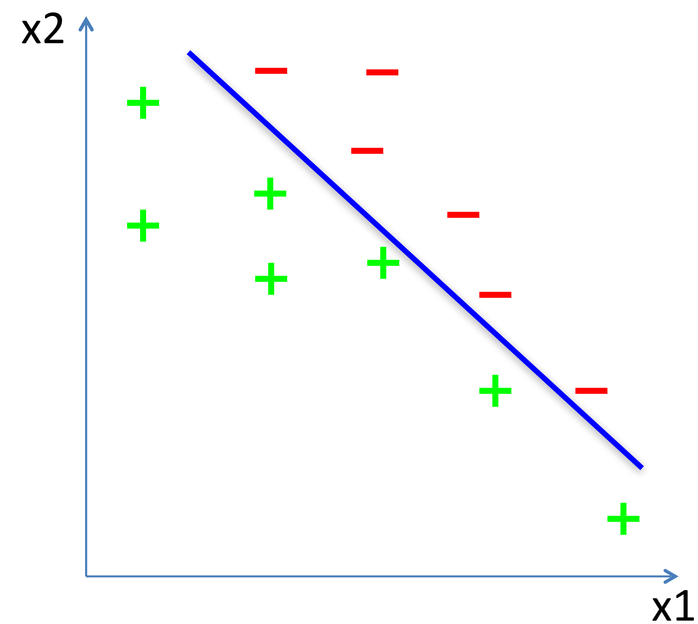
- **Generative Model:** We can build the decision functions (eg, line/plane/hyperplane) from the training samples using individual classes
  - For example, if we have two classes as shown below, we model one 2D line for class-1 examples and another 2D line for class-2 examples



Generative vs. Discriminative Model

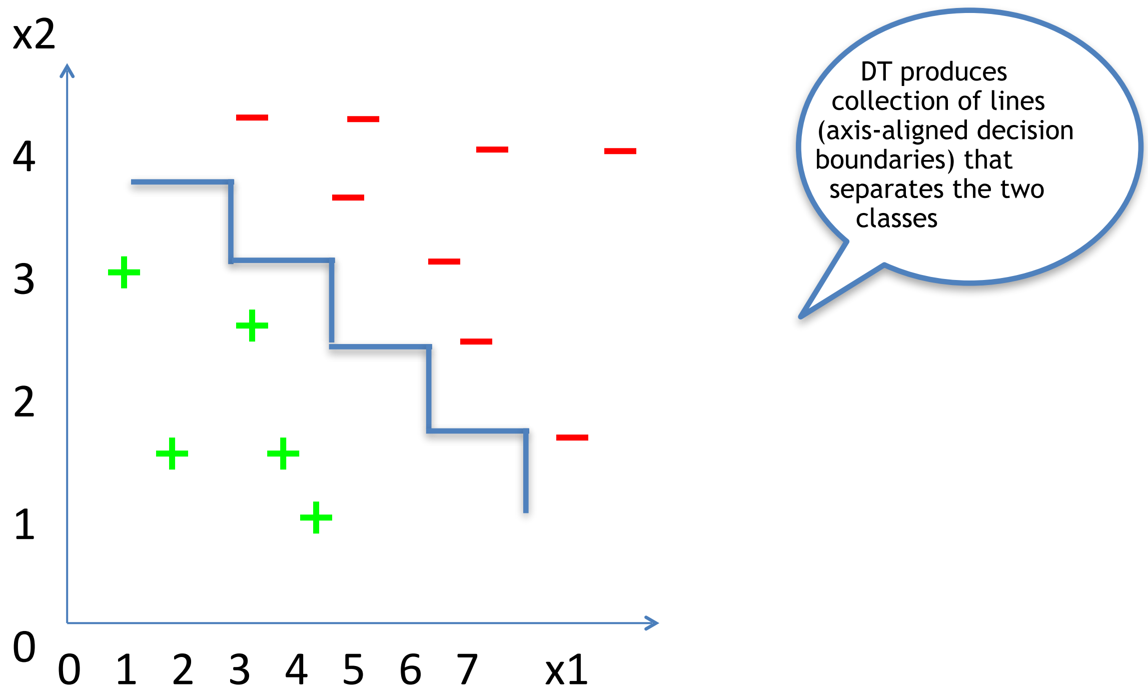
Generative vs **Discriminative**





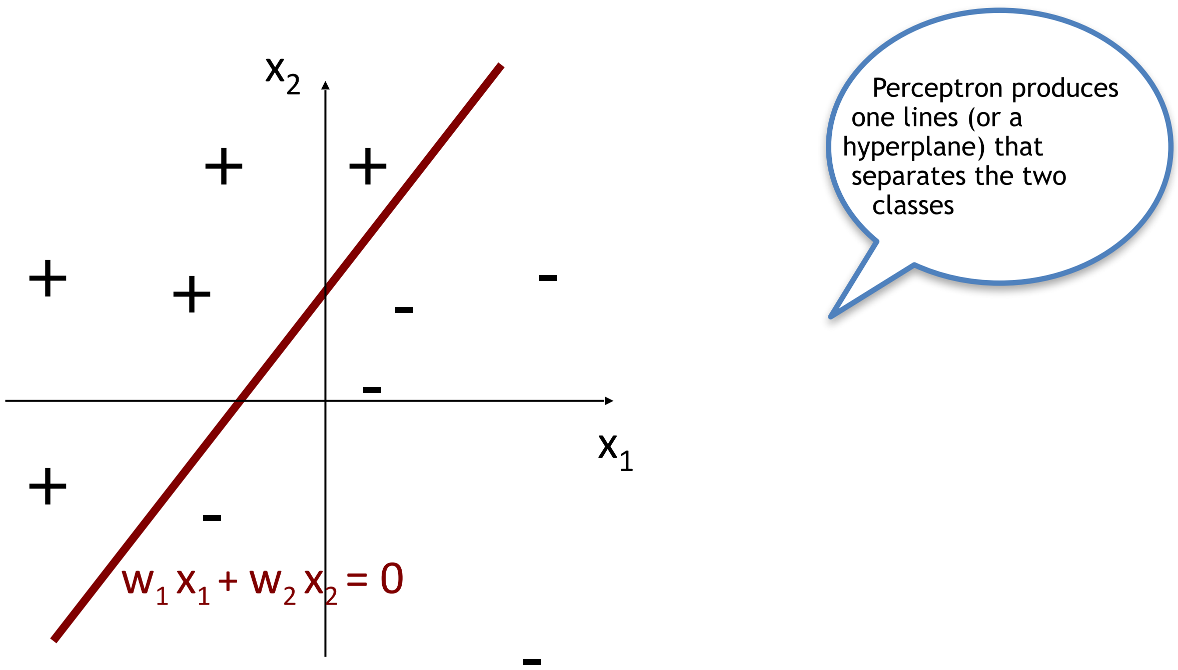
Example: Generative vs. Discriminative

- Is **decision tree (DT)** a generative model or a discriminative model?



Example: Generative vs. Discriminative

- Is **Perceptron** a generative model or a discriminative model?



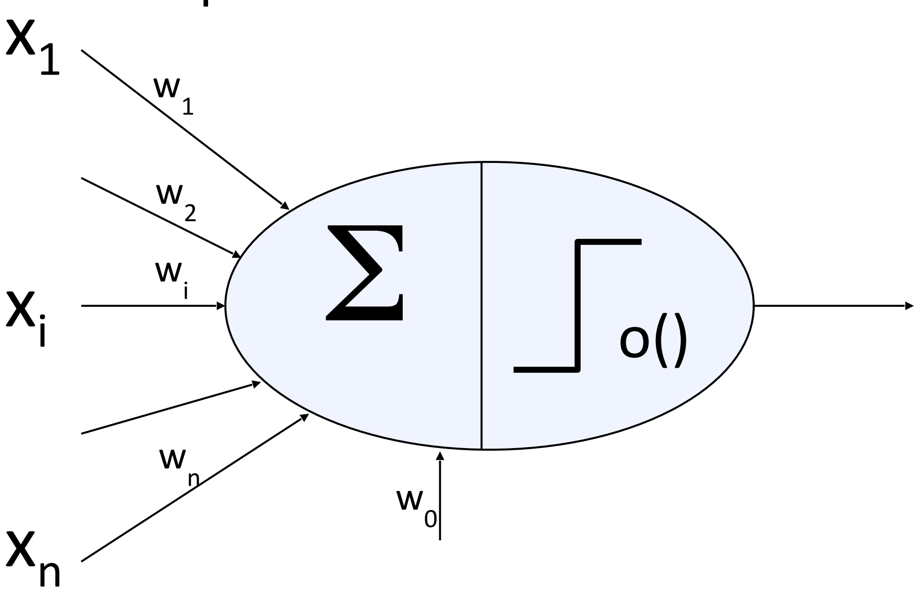
Today's Agenda

- Perceptron Learning Algorithm Group Activity
- Generative Model vs Discriminative Model
- Perceptron's limitation

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
    $$w_0+w_1*x_1+\ldots+w_n*x_n$$
  - Component 2: **a step function** which will produce 1 if the function value is positive and -1 otherwise



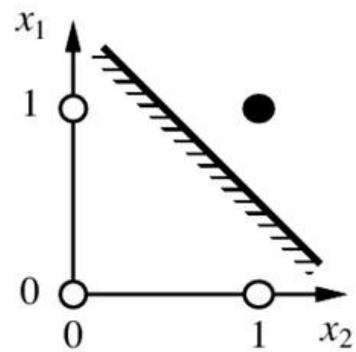
$$y = f(\mathbf{x},\mathbf{w}) = O(\Sigma_{i=1,\ldots,n} w_i x_i)$$

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

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

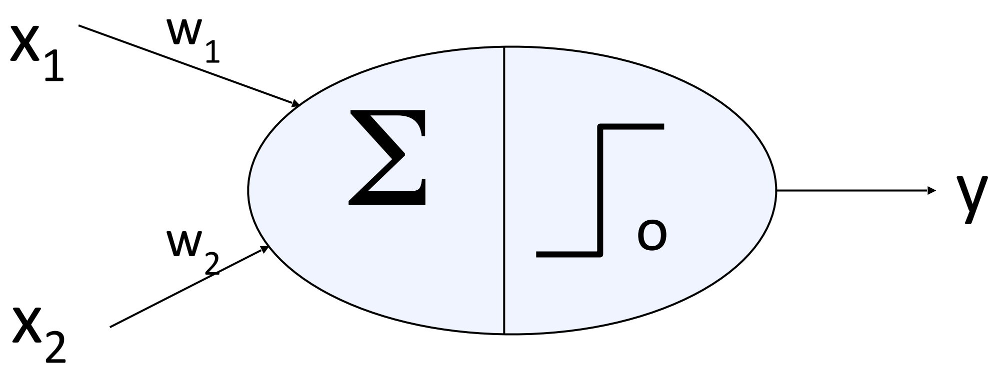


x_1 **and** x_2

YES, because we can find this 2D line

Color code:
Empty circle denotes 0
Black circle denotes 1

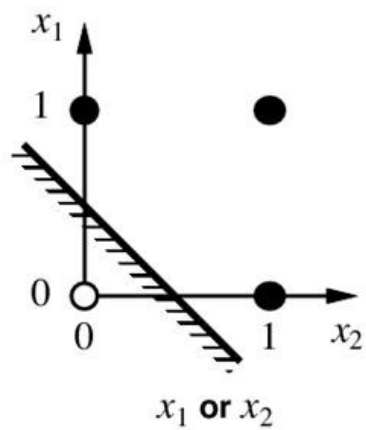
- Can a perceptron model AND function?



Perceptron can Model OR Function

- Let's consider the OR function.

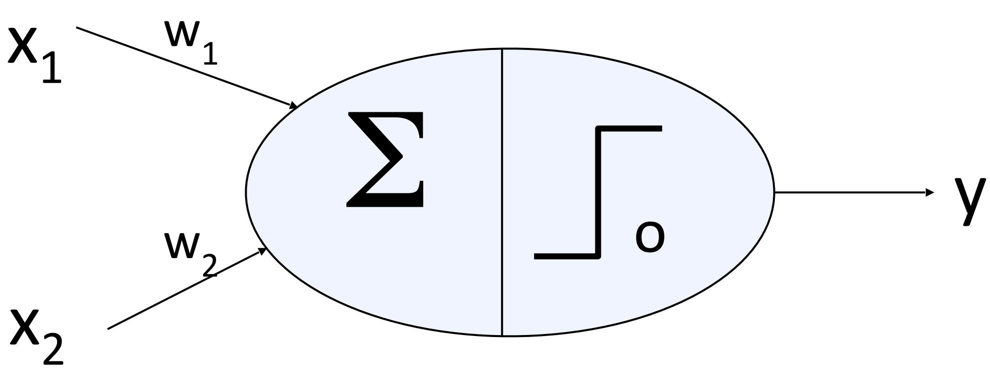
x	y	x or y
0	0	0
0	1	1
1	0	1
1	1	1



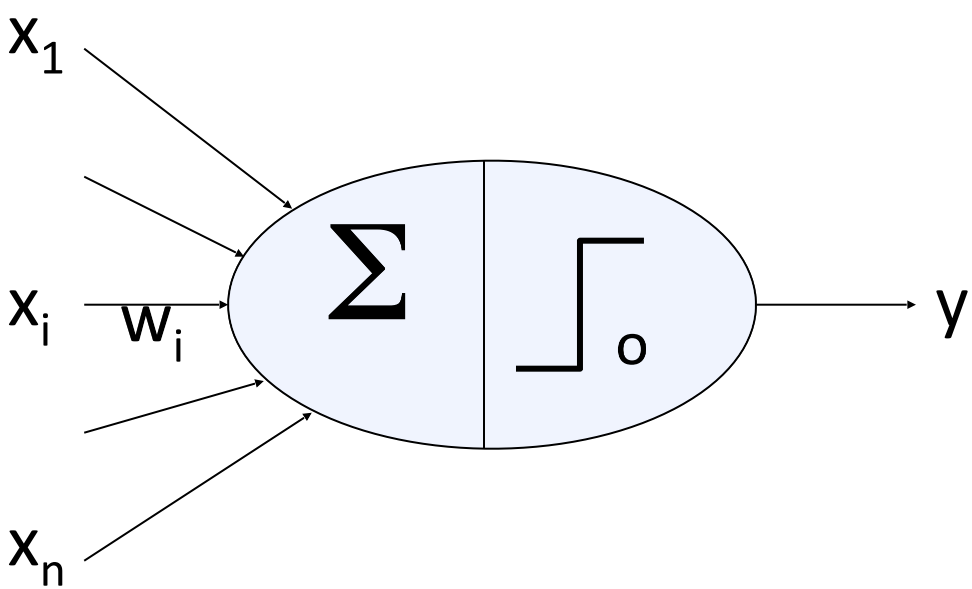
YES, because we can find this 2D line

Color code:
Empty circle denotes 0
Black circle denotes 1

- Can a perceptron model OR function?



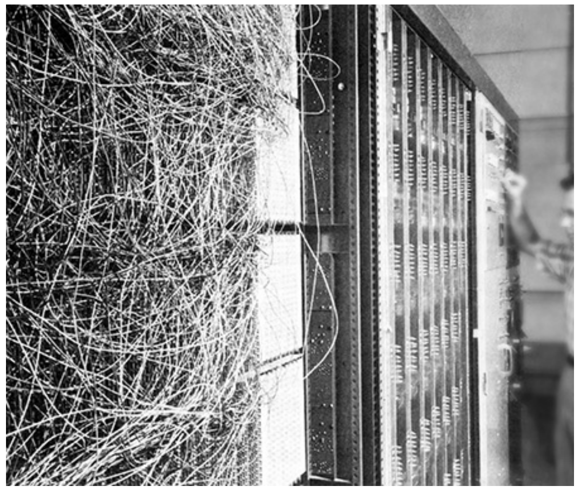
Can Perceptrons model any function?

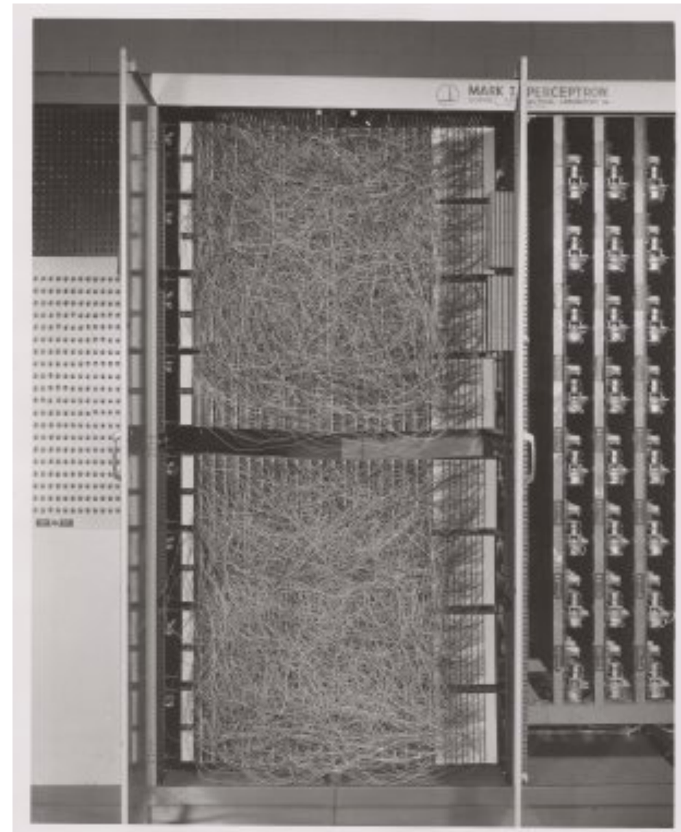


- Perceptron can also model other boolean functions such as $(x_1 \wedge x_2 \wedge \neg x_3)$. We can tabulate all combinations of the three variables and their corresponding outputs; then find weight parameters (of the plane separating it two classes (1 and 0)) using perceptron update rule we just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

- But can a perceptron model any function?

Perceptron





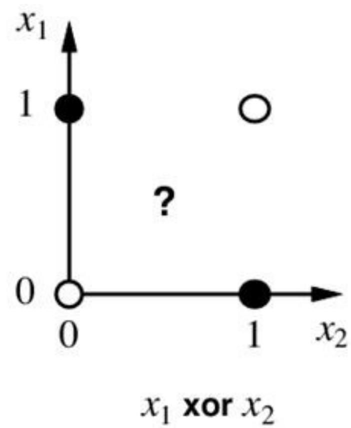
“the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”

Frank Rosenblatt, 1958

Now Let's Consider XOR Function

- Let's consider the XOR function.

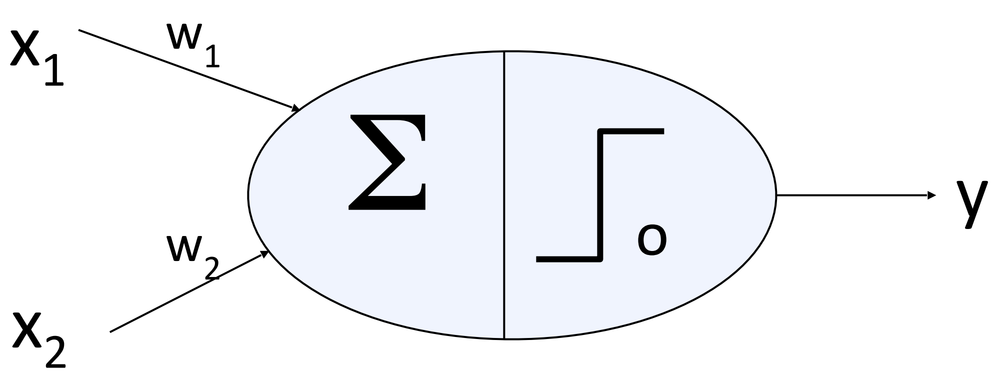
x	y	x xor y
0	0	0
0	1	1
1	0	1
1	1	0



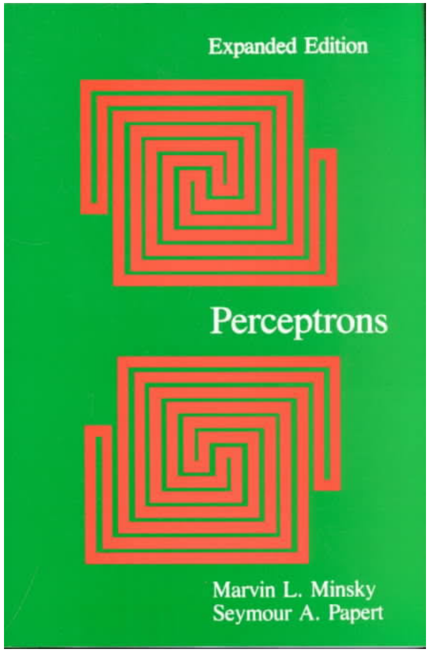
NO, because we can't find a single 2D line that separates the samples

Color code:
Empty circle denotes 0
Black circle denotes 1

- Can a perceptron model XOR function?



Perceptron

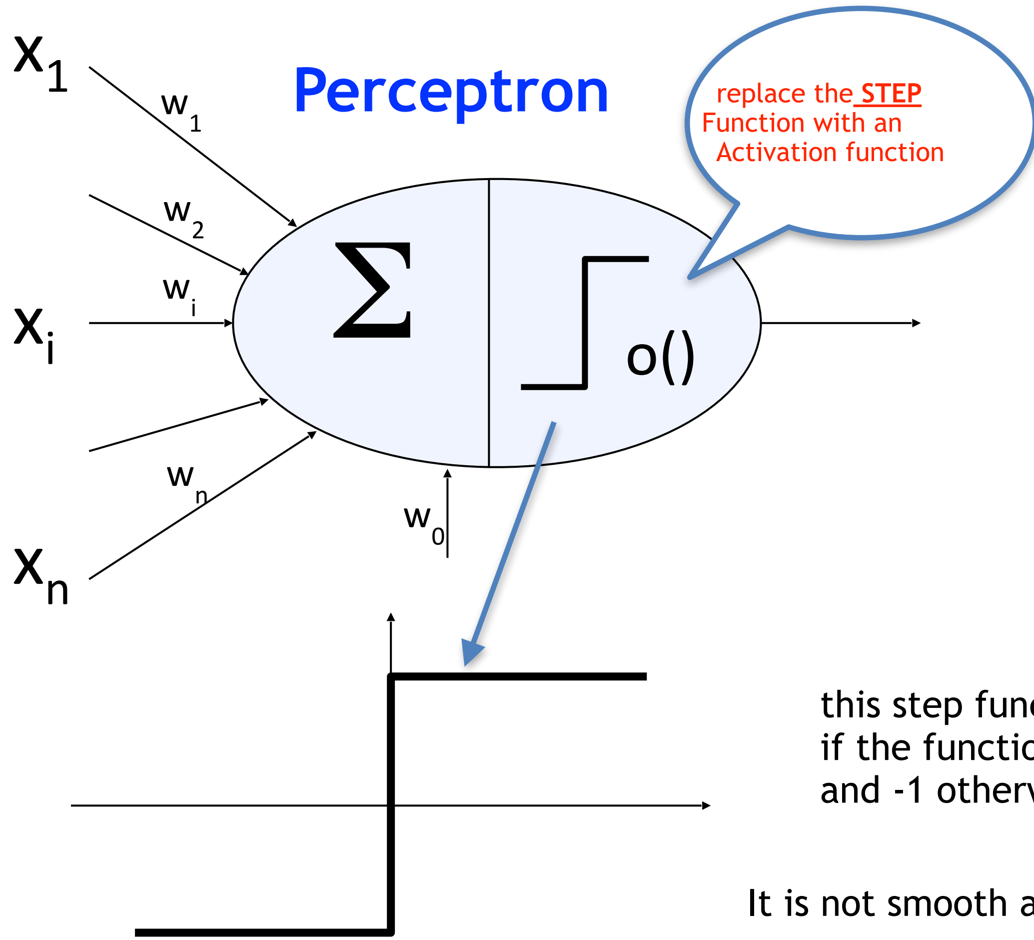


Marvin Minsky and Seymour Papert showed that they couldn't even learn XOR in 1969, which is why all the hype about the perceptron faded away

Today’s Agenda

- Perceptron Learning Algorithm Group Activity
- Generative Model vs Discriminative Model
- Perceptron’s limitation
- Neuron Model with Activation Function

Modification of Perceptron

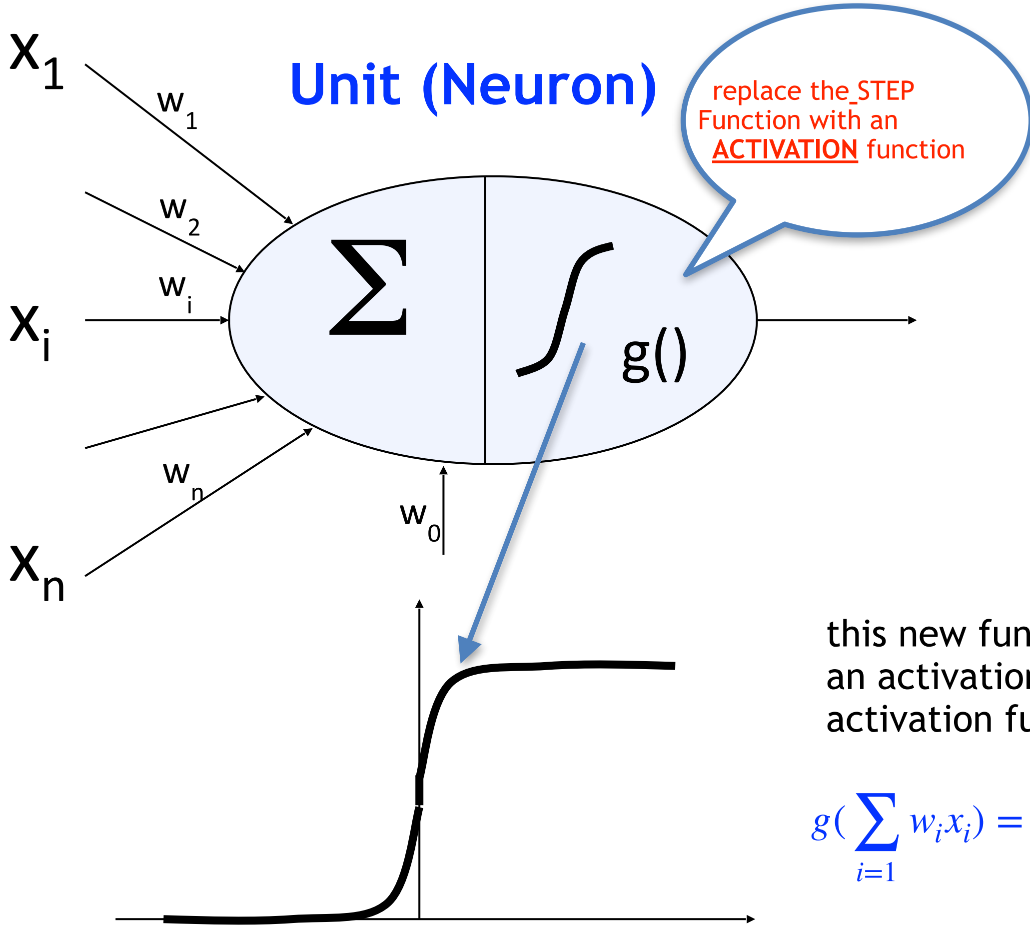


this step function outputs +1 if the function value is positive and -1 otherwise

It is not smooth and not differentiable

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

Make a Neuron with a Differentiable Function



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

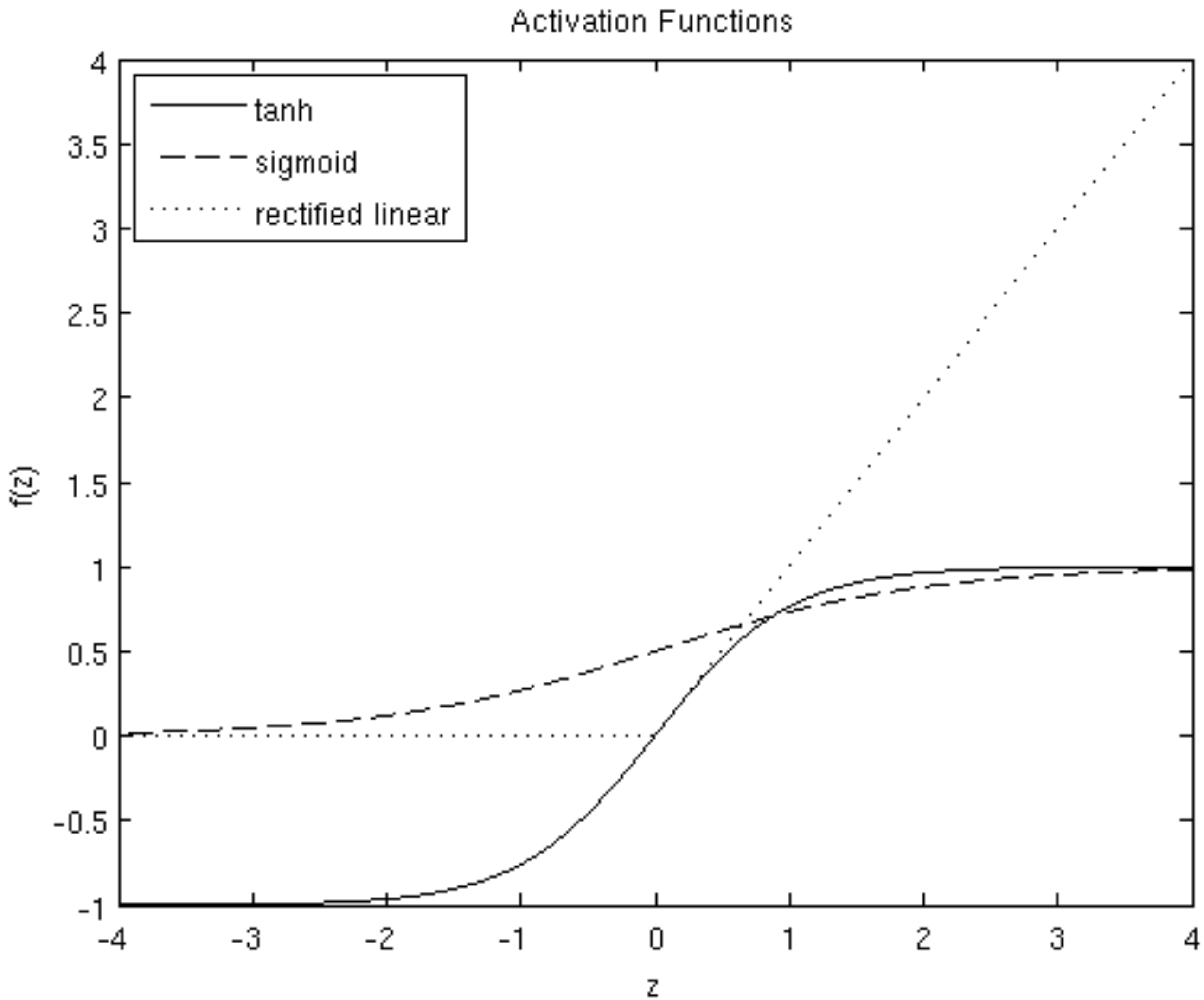
$$g(\sum_{i=1} w_i x_i) = \frac{1}{1 + exp(-\sum_{i=1} w_i x_i)}$$

It is smooth and differentiable

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

Common Activation Functions

Each activation function is smooth and differentiable

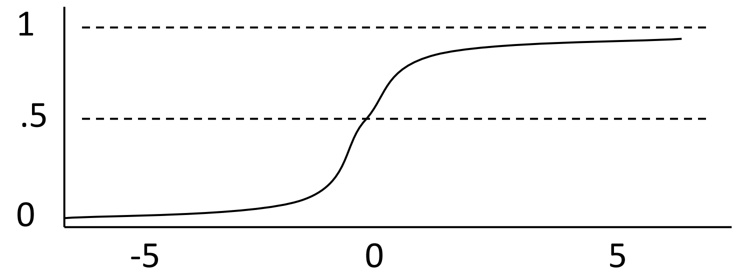


Sigmoid (a.k.a. Logistic Function) Activation

Sigmoid:

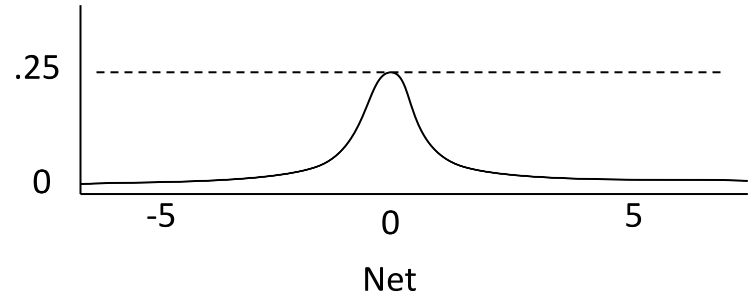
$$g(u) = \frac{1}{1 + exp^{-u}}$$

It is smooth and differentiable



Differentiation of sigmoid:

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



Today's Agenda

- Perceptron Learning Algorithm Group Activity
- Generative Model vs Discriminative Model
- Perceptron's limitation
- Neuron Model with Activation Function
- Mathematical Modeling of Weight Parameters Learning

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 ($w_0, w_1, w_2 \ldots, w_n$)

- But what is this new **weight parameter learning algorithm**?

Learning Weight Parameters with this Modified Neuron Model

Earthquake dataset training features

Seismometer Data	
Predictor₁	Predictor₂
Body Wave Magnitude	**Surface Wave Magnitude**
5.2	3.4
5.8	3.5
5.9	4.4
6.1	4.1
5.2	5
4.5	4.9
5.3	4.2
5.5	5.5
6.1	5.8

training labels

	Target
Classification	
underground explosion	`target: +1`
underground explosion	`target: +1`
underground explosion	`target: +1`
underground explosion	`target: +1`
earthquake	`target: -1`
earthquake	`target: -1`
earthquake	`target: -1`
earthquake	`target: -1`
earthquake	`target: -1`

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

$\mathbf{x^i}$ is a column vector denoting each training example

y^i

y^i is a scalar value denoting label of each training example

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

$$\mathbf{x}^1 = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.4 \end{bmatrix} \qquad \mathbf{x}^2 = \begin{bmatrix} 1.0 \\ ? \\ ? \end{bmatrix} \qquad \mathbf{x}^3 = \begin{bmatrix} 1.0 \\ ? \\ ? \end{bmatrix}$$

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

take a training example

Earthquake dataset training split

Seismometer Data	
Predictor₁	Predictor₂
Body Wave Magnitude	Surface Wave Magnitude
5.2	3.4
5.8	3.5
5.9	4.4
6.1	4.1
5.2	5
4.5	4.9
5.3	4.2
5.5	5.5
6.1	5.8

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

$$\mathbf{x^i} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{x^1} = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.4 \end{bmatrix} \qquad \mathbf{x^2} = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.5 \end{bmatrix} \qquad \mathbf{x^3} = \begin{bmatrix} 1.0 \\ ? \\ ? \end{bmatrix}$$

take a training example

Earthquake dataset training split

Seismometer Data	
Predictor$_1$	Predictor$_2$
Body Wave Magnitude	**Surface Wave Magnitude**
5.2	3.4
5.8	3.5
5.9	4.4
6.1	4.1
5.2	5
4.5	4.9
5.3	4.2
5.5	5.5
6.1	5.8

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

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

$$\mathbf{x^1} = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.4 \end{bmatrix} \qquad \mathbf{x^2} = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.5 \end{bmatrix} \qquad \mathbf{x^3} = \begin{bmatrix} 1.0 \\ 5.9 \\ 5.4 \end{bmatrix}$$

take a training example

Earthquake dataset training split

Seismometer Data	
$Predictor_1$	$Predictor_2$
Body Wave Magnitude	**Surface Wave Magnitude**
5.2	3.4
5.8	3.5
5.9	4.4
6.1	4.1
5.2	5
4.5	4.9
5.3	4.2
5.5	5.5
6.1	5.8

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

$$\mathbf{x^i} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{x^1} = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.4 \end{bmatrix} \qquad \mathbf{x^2} = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.5 \end{bmatrix} \qquad \mathbf{x^3} = \begin{bmatrix} 1.0 \\ 5.9 \\ 5.4 \end{bmatrix}$$

training labels

Classification	Target
underground explosion	target: +1
underground explosion	target: +1
underground explosion	target: +1
underground explosion	target: +1
earthquake	target: −1
earthquake	target: −1
earthquake	target: −1
earthquake	target: −1
earthquake	target: −1

$$y^1 = ? \qquad y^2 = ? \qquad y^3 = ?$$

y^i

y^i is a scalar value denoting label of each training example

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

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

$$\mathbf{x^1} = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.4 \end{bmatrix} \qquad \mathbf{x^2} = \begin{bmatrix} 1.0 \\ 5.2 \\ 3.5 \end{bmatrix} \qquad \mathbf{x^3} = \begin{bmatrix} 1.0 \\ 5.9 \\ 5.4 \end{bmatrix}$$

training labels

Classification	Target
underground explosion	target: +1
underground explosion	target: +1
underground explosion	target: +1
underground explosion	target: +1
earthquake	target: -1
earthquake	target: -1
earthquake	target: -1
earthquake	target: -1
earthquake	target: -1

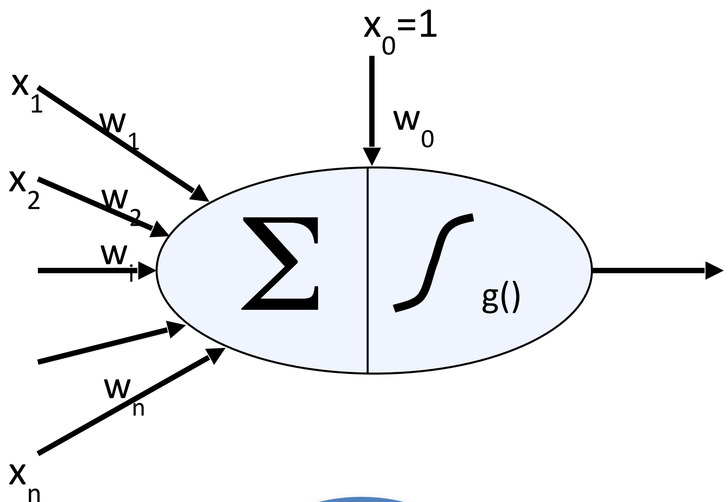
$$y^1 = +1 \qquad y^2 = +1 \qquad y^3 = +1$$

y^i is a scalar value denoting label of each training example

y^i

Learning Weight Parameters with this Modified Neuron Model

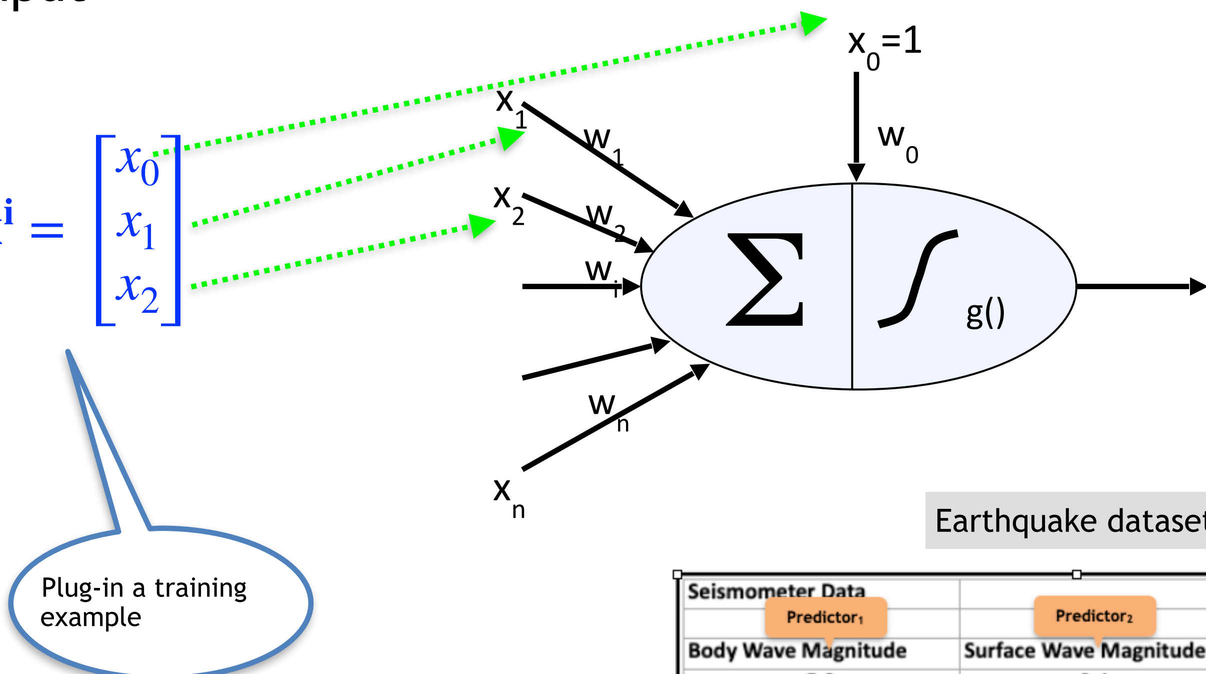
- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



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

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

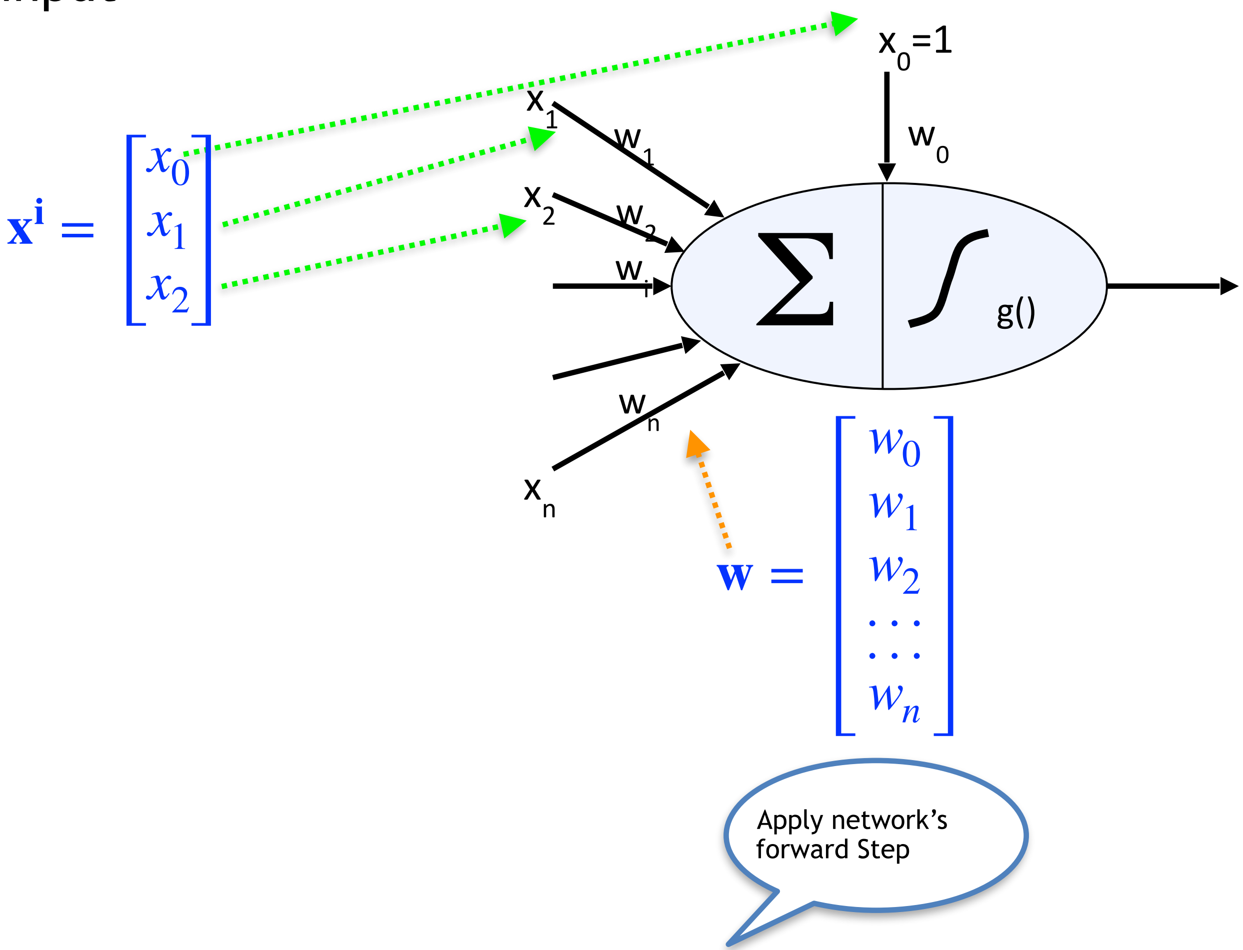


Earthquake dataset training split

Seismometer Data			
Predictor₁	Predictor₂		Target
Body Wave Magnitude	Surface Wave Magnitude	Classification	
5.2	3.4	underground explosion	target: +1
5.8	3.5	underground explosion	target: +1
5.9	4.4	underground explosion	target: +1
6.1	4.1	underground explosion	target: +1
5.2	5	earthquake	target: -1
4.5	4.9	earthquake	target: -1
5.3	4.2	earthquake	target: -1
5.5	5.5	earthquake	target: -1
6.1	5.8	earthquake	target: -1

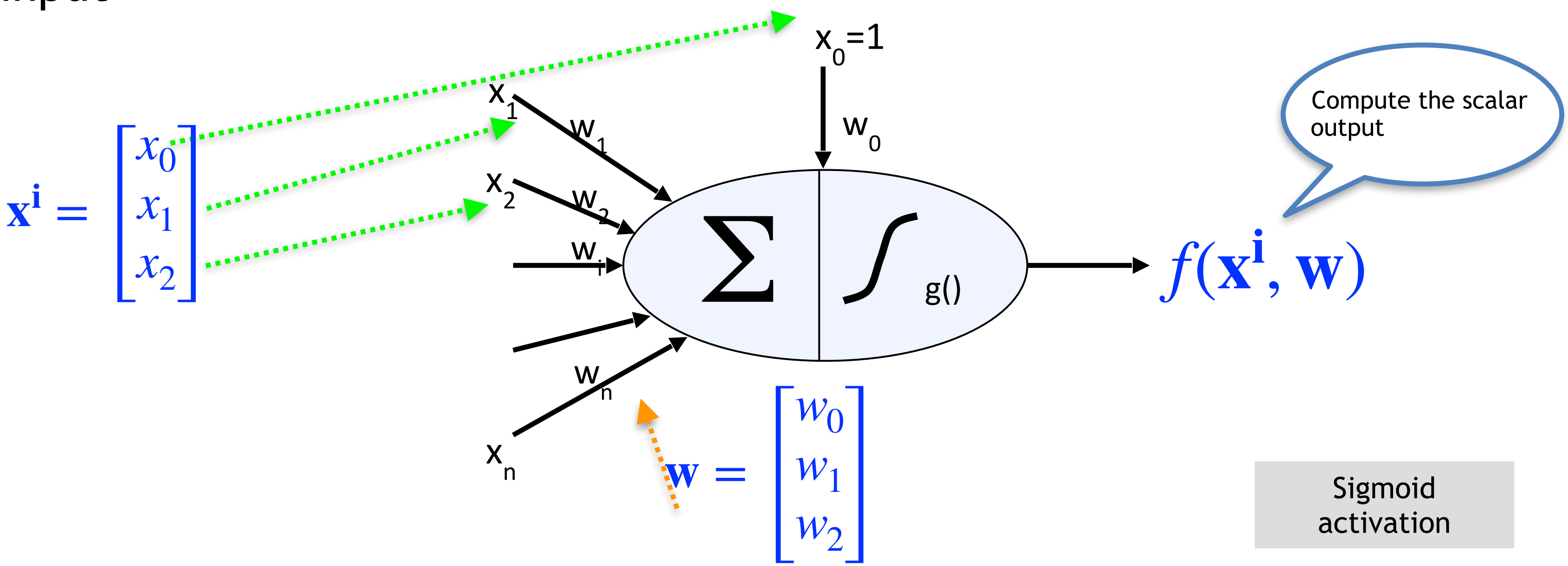
Learning Weight Parameters with this Modified Neuron Model

- 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

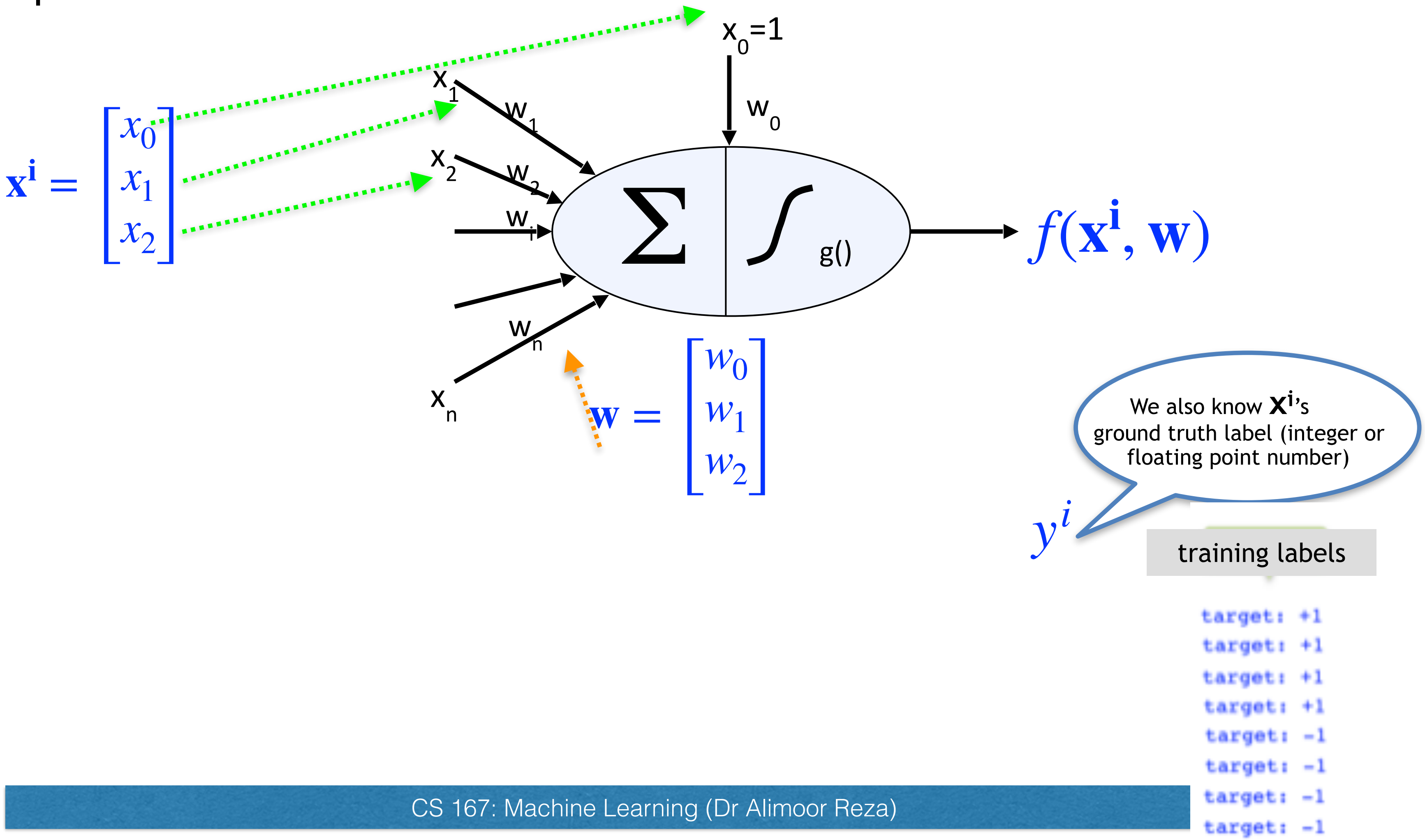


Sigmoid activation

$$f(\mathbf{x^i}, \mathbf{w}) = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{exp}^{-\mathbf{w^T x^i}}}$$

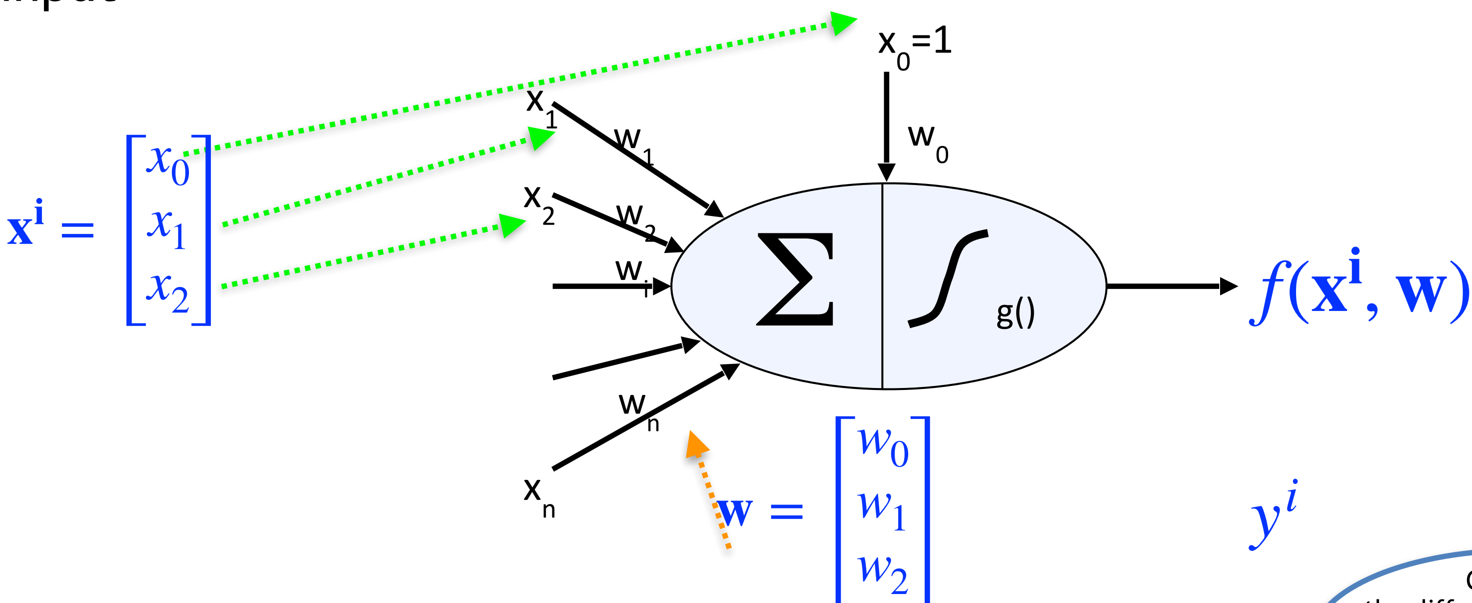
Learning Weight Parameters with this Modified Neuron Model

- 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



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



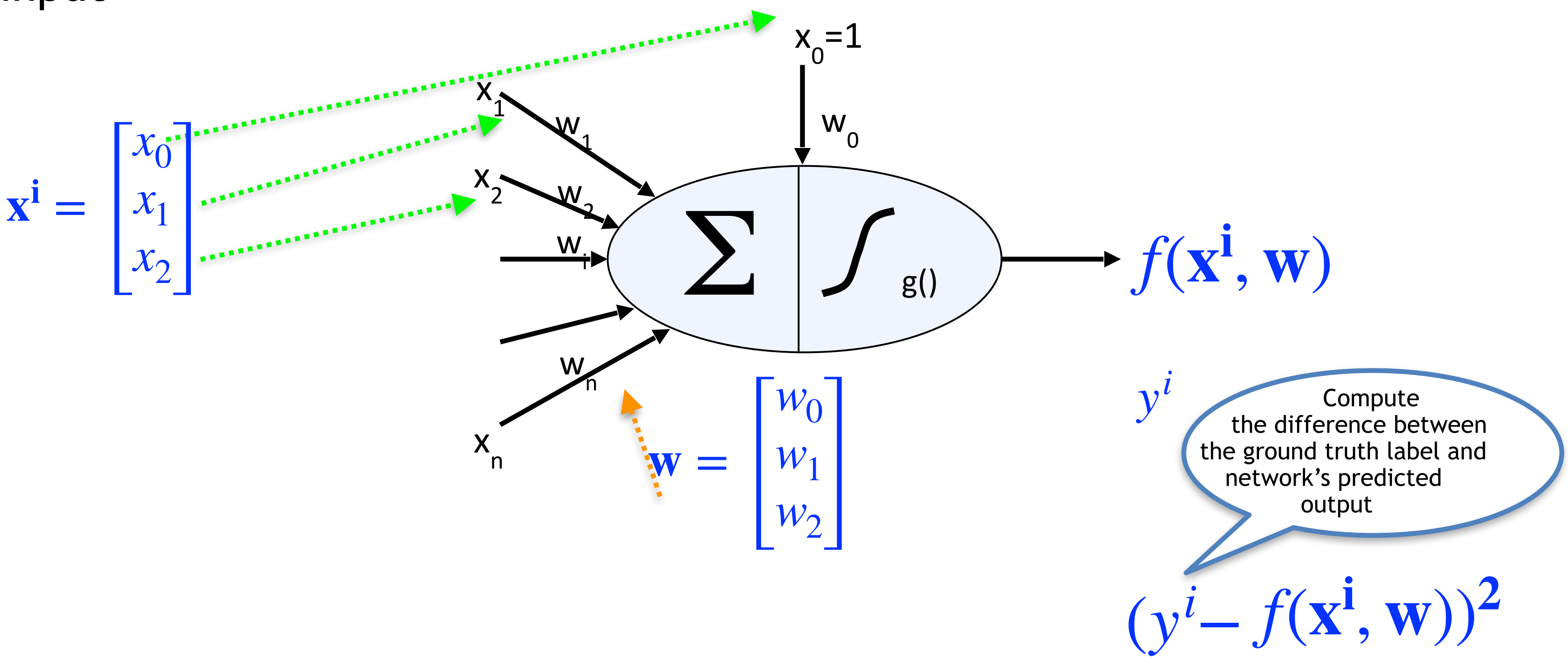
Compute the difference between the ground truth label and network's predicted output

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

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

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

- Perceptron Learning Algorithm Group Activity
- Generative Model vs Discriminative Model
- Perceptron’s limitation
- Neuron Model with Activation Function
- Mathematical Modeling of Weight Parameters Learning
- Intuitive Understanding of Optimization (minimization/maximization)

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 <u>objective function</u>

  - **maximization**: trying to find the subset of values for attributes that gives you the maximum value in the <u>objective function</u>

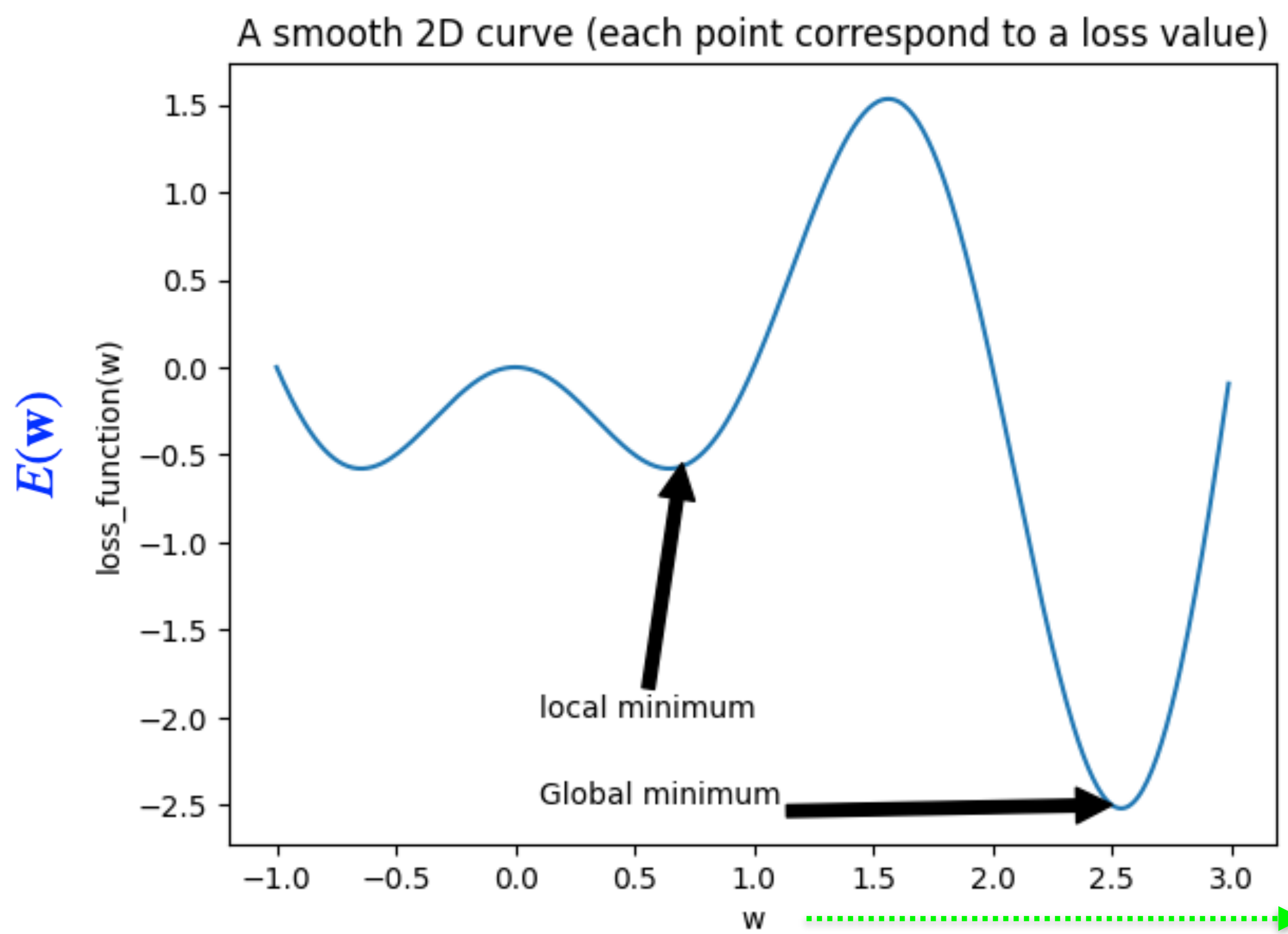
Optimization

- **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

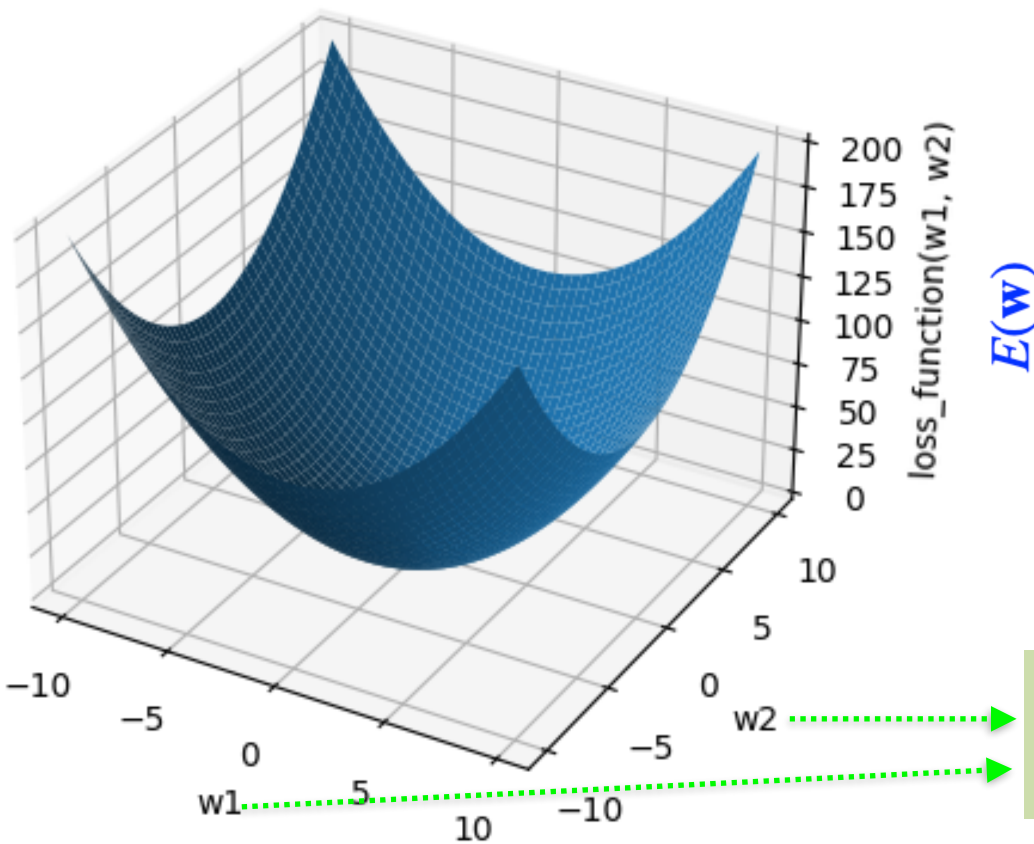


minimizing the loss function with respect to only one value w

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)



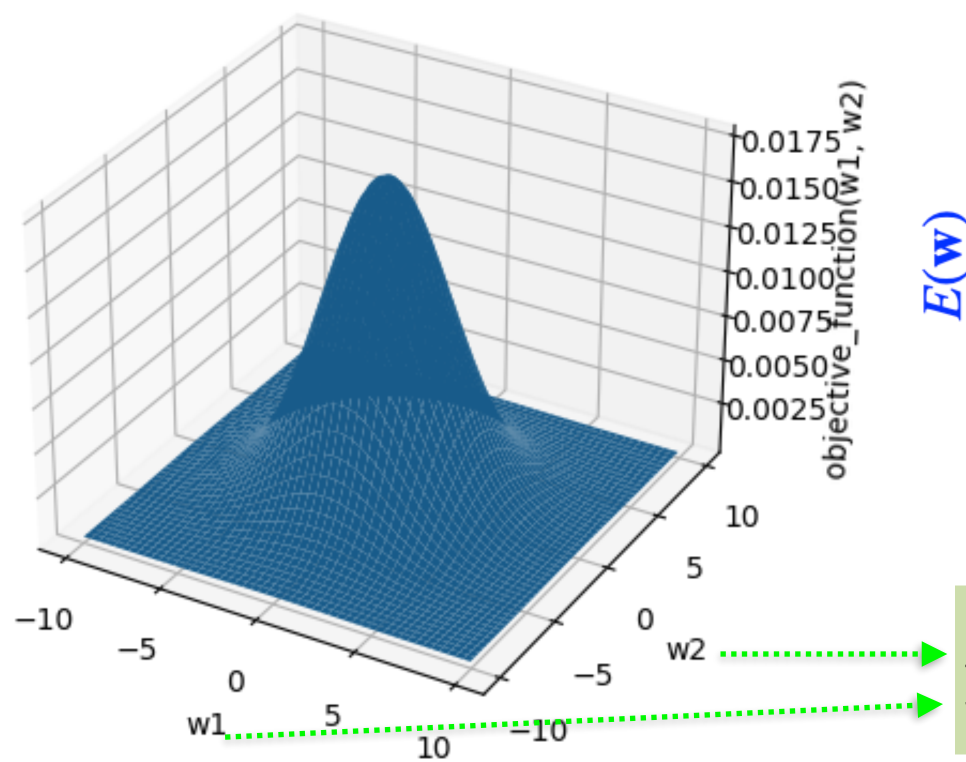
Play with the code I uploaded on Blackboard to generates different surfaces

minimizing the loss function with respect to two values w_1 and w_2

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)



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

maximizing the loss function with respect to two values w_1 and w_2

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)