CS167: Machine Learning

Perceptron (continued)
Perceptron Learning Algorithm
Expressive Power of Perceptron

Monday, March 30th, 2026

Drake
UNIVERSITY

Today's Agenda

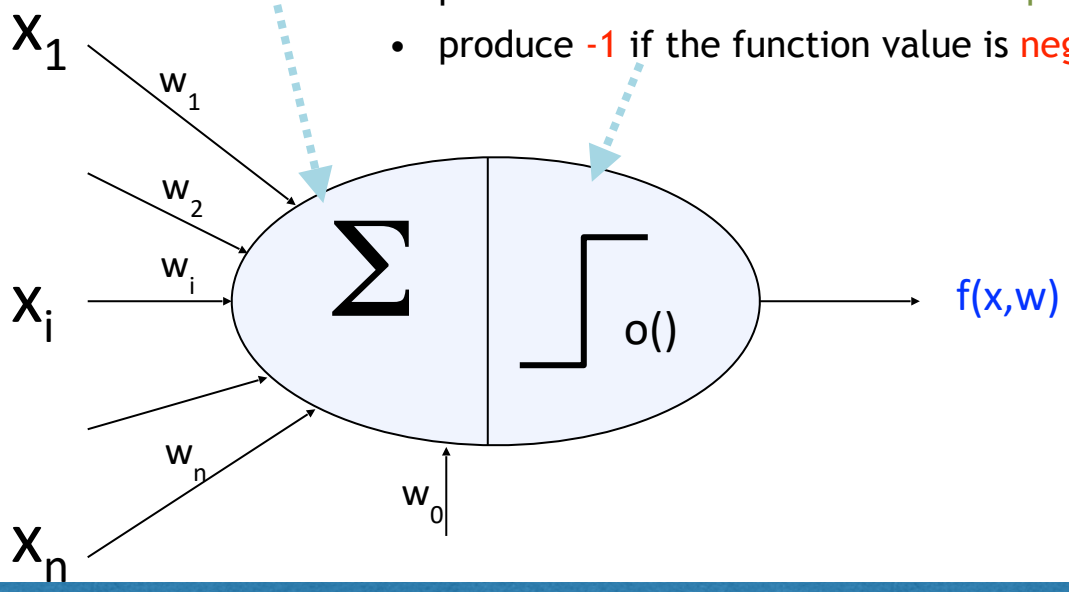
- New linear classifier: Perceptron
- Perceptron Learning Algorithm
- In-class activity: Perceptron Learning Algorithm
- Expressive Power of a Perceptron

New Linear Classifier: Perceptron

- Let's build a mathematical model consisting of a *single neuron* is called a **perceptron** expressed by a function $f(x,w)$ with two components applied in a sequential manner:
  - Component 1: a linear model of the form (it is a hyperplane which we just discussed):

$$w_0*x_0+w_1*x_1+\ldots+w_n*x_n$$

  - Component 2: **a step function o()** which will
    - produce +1 if the function value is positive or
    - produce -1 if the function value is negative

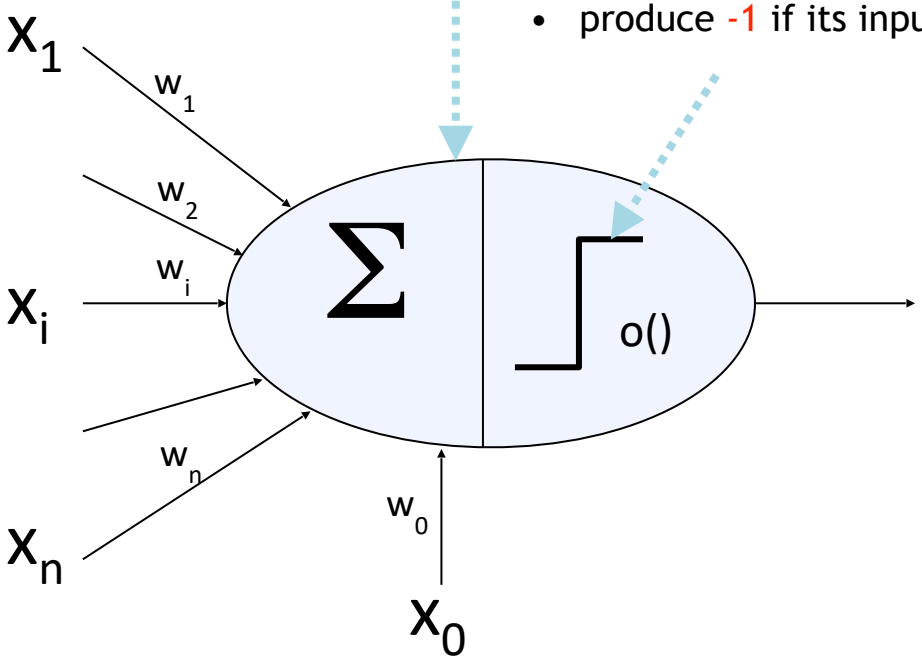


New Linear Classifier: Perceptron

- Component 1: a linear model of the form (it is a hyperplane which we just discussed):

$$w_0*x_0+w_1*x_1+\ldots+w_n*x_n$$

- Component 2: a **step function** o() which will
  - produce +1 if its input value is positive or
  - produce -1 if its input value is negative



$$f(x,w) = component_2(component_1)$$

$$= component_2(w_0*x_0+w_1*x_1+\ldots+w_n*x_n)$$

$$= component_2(\text{scalar value/just a number})$$

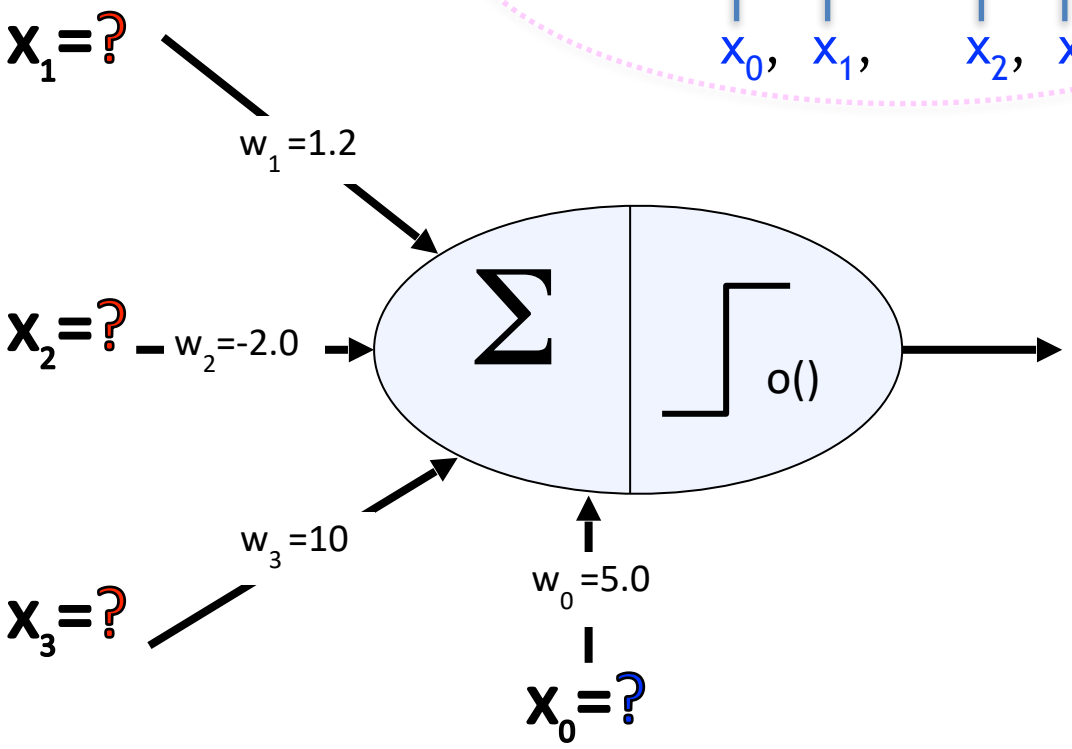
$$= o(\text{scalar value/just a number})$$

Example: Perceptron

- Consider the inputs: S_1 = [1,1,2,3], S_2 = [1, 2,4,6], … S_{20} = [1, 2,4,-6]
- The labels of the input samples are: T_1 = +1, T_2 = +1, …, T_{20} = -1

- Let consider sample#2:
- S_2 = [1, 2, 4, 6], T_2 = +1

x_0, x_1, x_2, x_3



$f(x,w)$ = component$_2$(component$_1$)

= component$_2$($w_0*x_0+w_1*x_1+w_2*x_2+w_3*x_3$)

= component$_2$(scalar value/just a number)

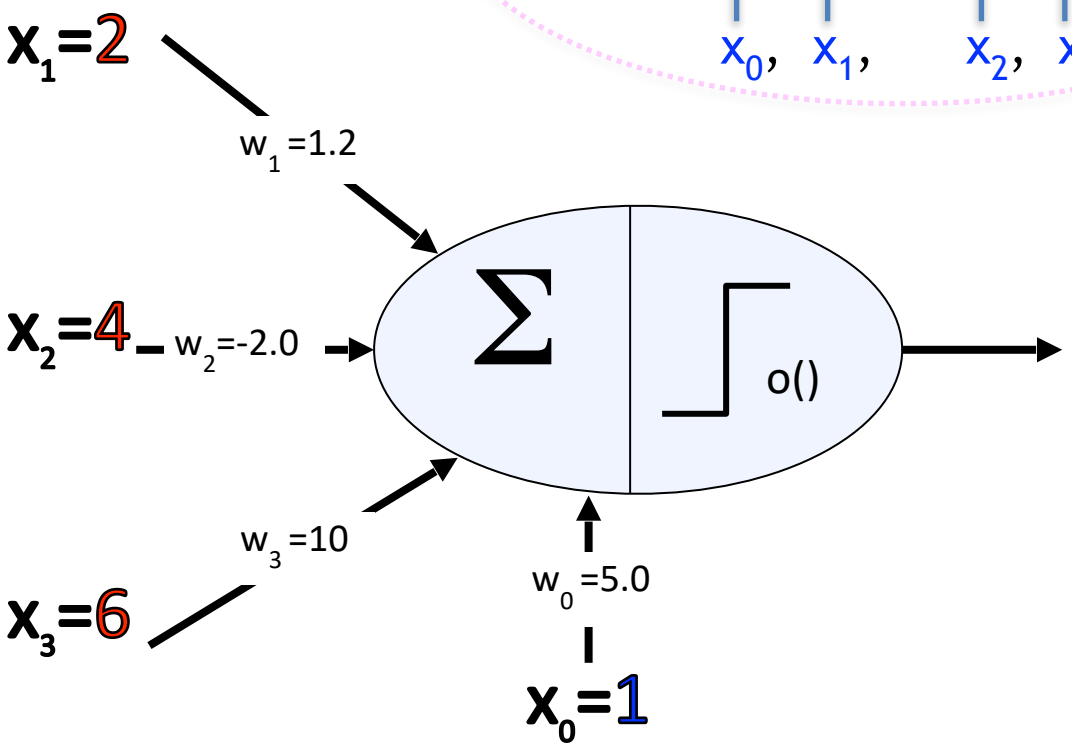
= 0(scalar value/just a number)

Example: Perceptron

- Consider the inputs: S_1 = [1,1,2,3], S_2 = [1, 2,4,6], … S_{20} = [1, 2,4,-6]
- The labels of the input samples are: T_1 = +1, T_2 = +1, …, T_{20} = -1

- Let consider sample#2:
- S_2 = [1, 2, 4, 6], T_2 = +1

x_0, x_1, x_2, x_3



$f(x,w) = component_2(component_1)$

$= component_2(w_0*x_0+w_1*x_1+w_2*x_2+w_3*x_3)$

$= component_2(\text{scalar value/just a number})$

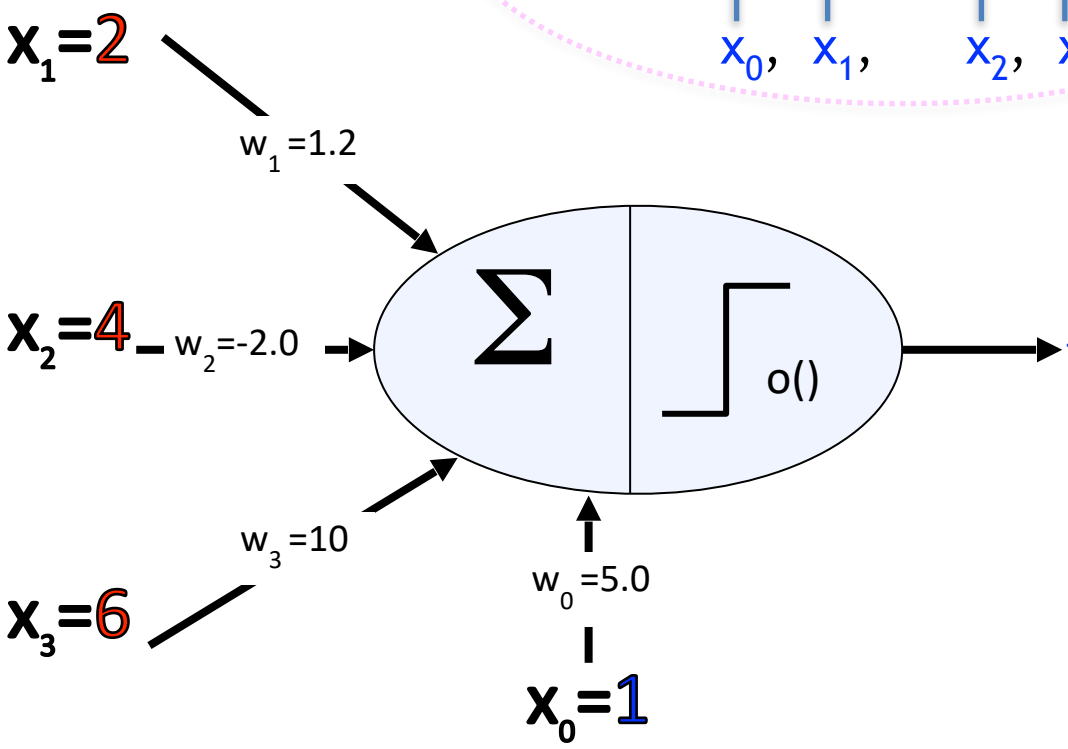
$= 0(\text{scalar value/just a number})$

Example: Perceptron

- Consider the inputs: S_1 = [1,1,2,3], S_2 = [1, 2,4,6], … S_{20} = [1, 2,4,-6]
- The labels of the input samples are: T_1 = +1, T_2 = +1, …, T_{20} = -1

- Let consider sample#2:
- S_2 = [1, 2, 4, 6], T_2 = +1

x_0, x_1, x_2, x_3



$f(x,w) = component_2(component_1)$

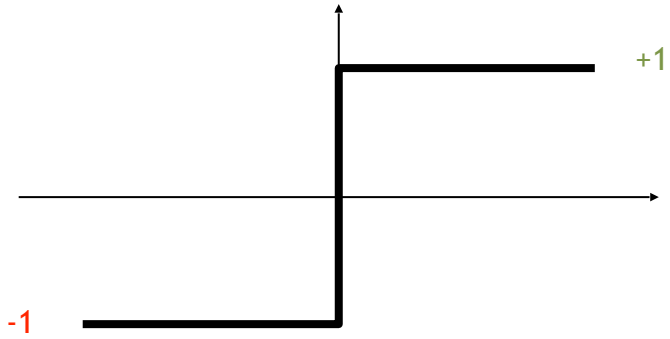
$= component_2(w_0*x_0+w_1*x_1+w_2*x_2+w_3*x_3)$

$= component_2(5.0*1+1.2*2+(-2.0)*4+10.0*6)$

$= 0(59.4)$

New Linear Classifier: 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
  - Component 2: **a step function** which will
    - produce +1 if its input value is positive or
    - produce -1 if its input value is negative



$o(w_0*x_0+w_1*x_1+\ldots+w_n*x_n)$

$f(x,w) = component_2(component_1)$

$= component_2(w_0*x_0+w_1*x_1+w_2*x_2+w_3*x_3)$

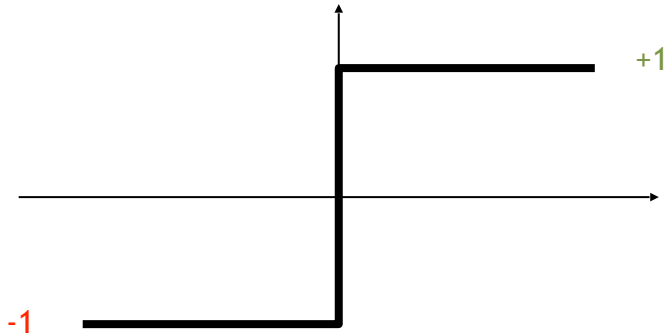
$= component_2(5.0*1+1.2*2+(-2.0)*4+10.0*6)$

$= o(59.4)$

= ?

New Linear Classifier: 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
  - Component 2: **a step function** which will
    - produce +1 if its input value is positive or
    - produce -1 if its input value is negative



$$o(w_0*x_0+w_1*x_1+\ldots+w_n*x_n)$$

$$f(x,w) = \text{component}_2(\text{component}_1)$$

$$= \text{component}_2(w_0*x_0+w_1*x_1+w_2*x_2+w_3*x_3)$$

$$= \text{component}_2(5.0*1+1.2*2+(-2.0)*4+10.0*6)$$

$$= o(59.4)$$

$$= +1$$

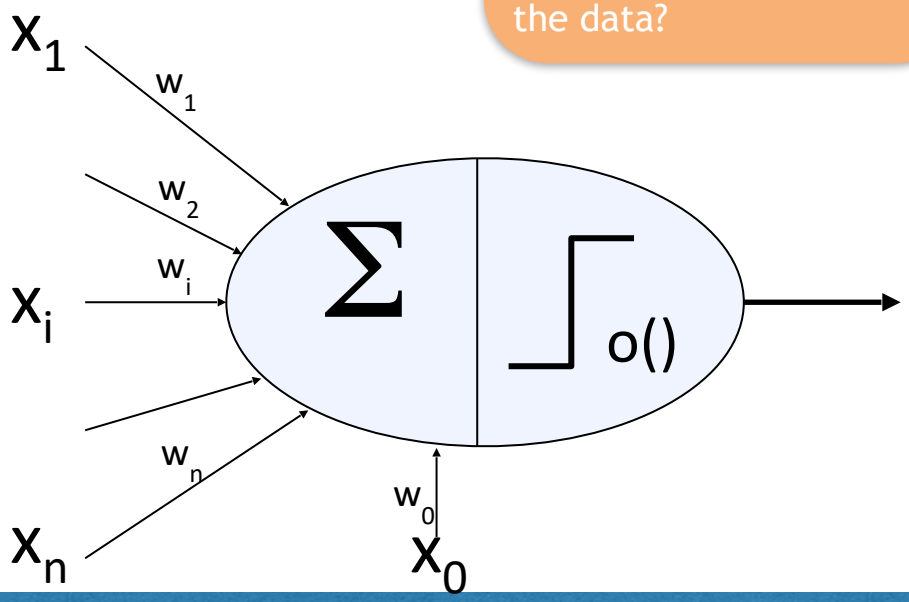
Because input value is positive so the output will be +1

Perceptron

- when you are given some training samples, you can consider
  - each x_i is one of the predictor features --i.e. `sepal length, width, etc`
  - each w_i is some weight we multiply the feature value by

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

w_1=?, w_2=?, …, w_n=?



$f(\mathbf{x},\mathbf{w}) = o(w_0*x_0+w_1*x_1+\ldots+w_n*x_n)$

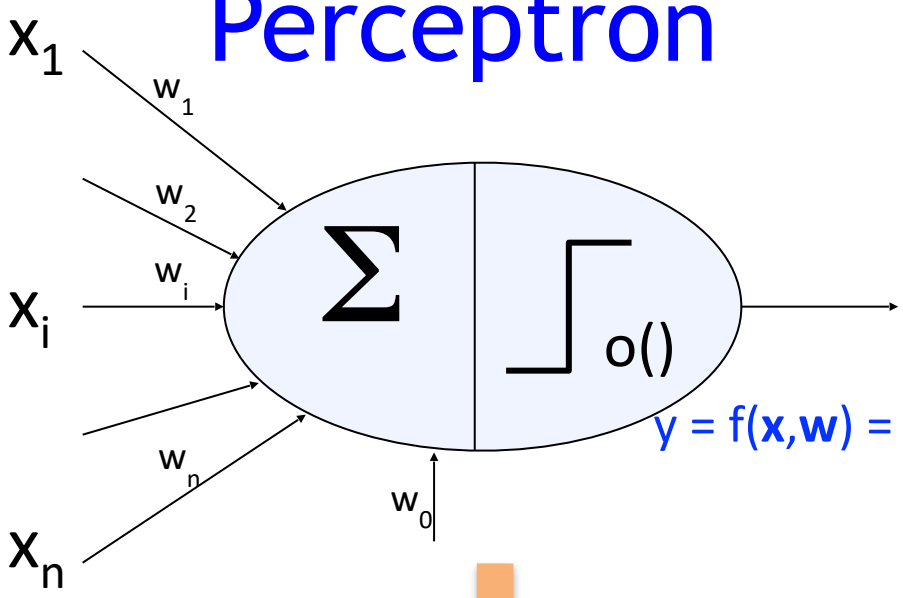
Today's Agenda

- New linear classifier: Perceptron
- Perceptron Learning Algorithm
- In-class activity: Perceptron Learning Algorithm
- Expressive Power of a Perceptron

How to train your Perceptron?

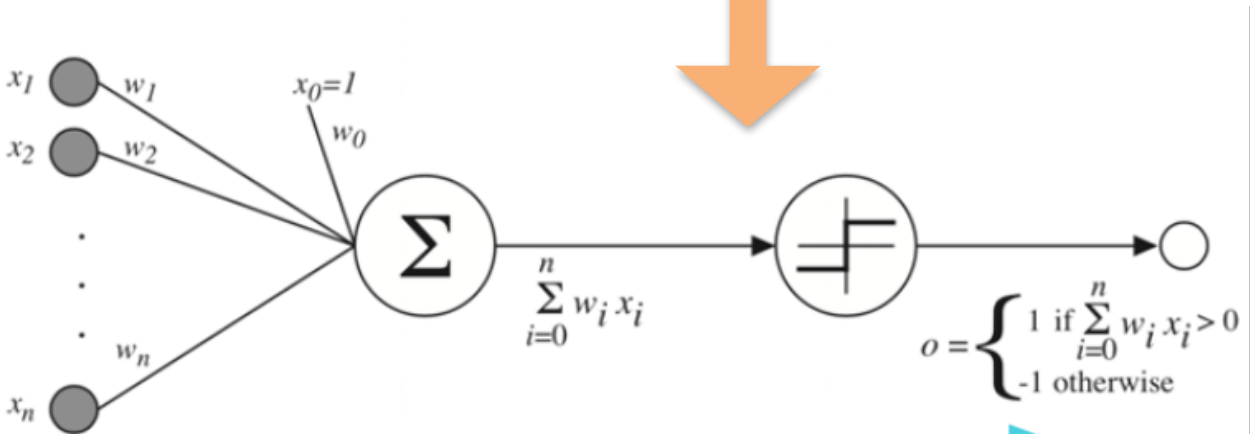
- How do we **determine the weights that best classify** the data?
- One strategy could be as follows:
  - Start with 0 values for all weights: $(w_0, w_1, w_2, \ldots, w_n)$
  - Feed in *one* *training example* at a time
    - If *training example* is properly classified
      - great, move on
    - Else
      - update weights according to the *training example*
    - Repeat

Perceptron



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

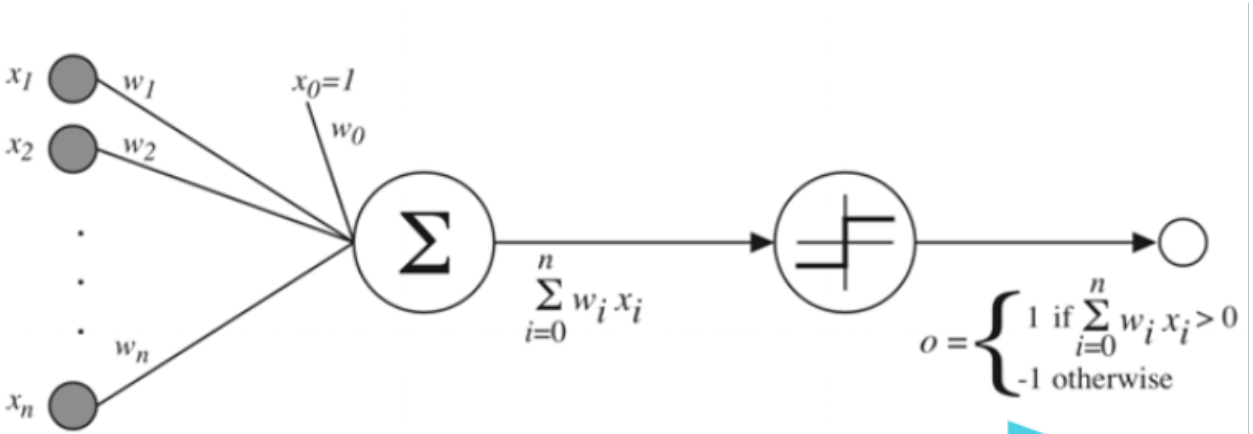
Expansion of the two components inside the perceptron



Perceptron Learning Algorithm: forward step

- The **forward step** in a perceptron is the step of receiving input, and getting an output based on the input. In other words, making a prediction from an input. This step is also known as the *inference step*

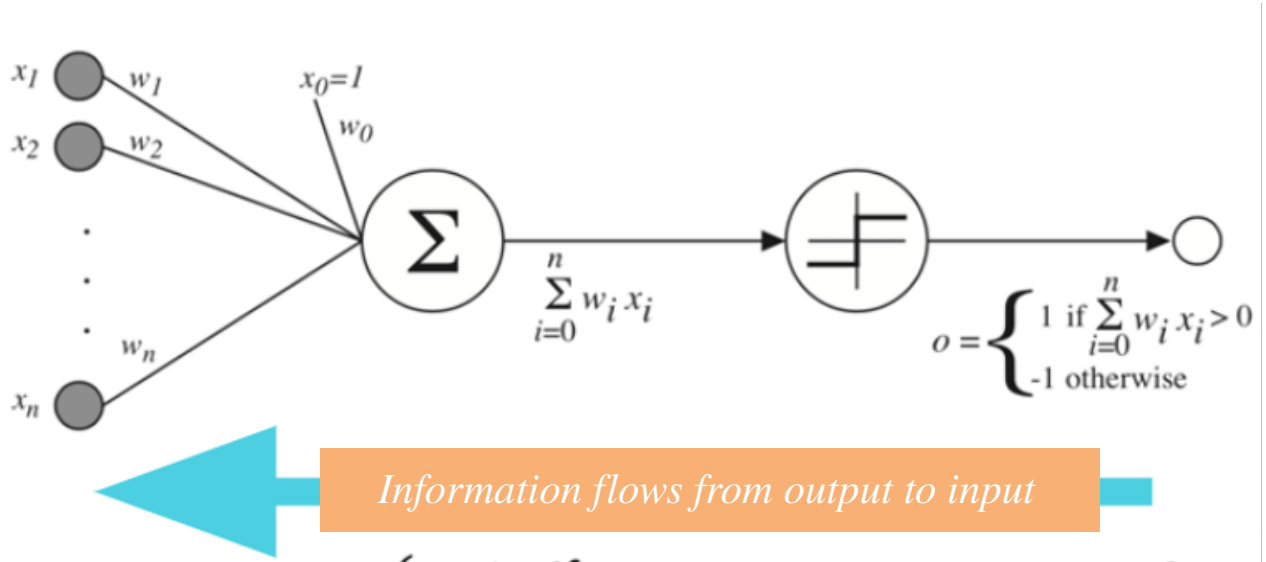
- Feed a training example $(x_0, x_2, \ldots, x_n)$ into the perceptron, then get its output (which is a scalar value): o



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Perceptron Learning Algorithm: backward step

- The **backward step** in a perceptron is the step of checking to see if the model got the prediction correct, and if not, adjusting the weights to make a better prediction next time. We also call this step *updating the weights*
- Feed a training example $(x_0, x_2, \ldots, x_n)$ into the perceptron, then for each weight $(w_0, w_1, \ldots, w_n)$ update its value



Perceptron Learning Algorithm: backward step

- Feed a training example $(x_0, x_2, \ldots, x_n)$ into the perceptron, then for each weight $(w_0, w_1, \ldots, w_n)$ update its value as follows:

Frank Rosenblatt proposed this process in 1958 which is known as **Perceptron Learning Algorithm**

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

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

Value in the **target column** in your csv file/dataframe

t = target

o = output

- **target** is the thing in the 'target column' for this example - what you're trying to predict
- **output** is the perceptron output - what the perceptron is currently set to predict for this example
- η**(eta)** is a small constant (e.g. 0.1) called the **learning rate** (controls how quickly the model adapts to the training data)

Perceptron Learning Algorithm: backward step

- Feed a training example $(x_0, x_2, \ldots, x_n)$ into the perceptron, then for each weight $(w_0, w_1, \ldots, w_n)$ update its value as follows:

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

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

Value in the **target column** in your csv file/dataframe

t = target

o = output

- Note what happens when:
  - target is +1, output is +1 — Agreement
  - target is -1, output is -1 — Agreement
  - target is +1, output is -1 — Disagreement
  - target is -1, output is +1 — Disagreement

Group Activity on Perceptron Learning Algorithm

- Here is a sample dataset of binary classification problem:
  - `underground explosion (target: +1) vs. earthquake (target: -1)`

Seismometer Data			
Predictor$_1$	Predictor$_2$		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`
5.6	3.9	?	
5.7	4.6	?	
4.7	4.7	?	
6.4	4.5	?	

Group Activity

Predictor$_1$ | Predictor$_2$ | Target

X_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	5.2	3.4	Explosion	1
1	4.5	4.9	Earthquake	-1
1	6.1	5.8	Earthquake	-1
1				
1	5.6	3.9	?	?

Append this column with a constant value of '**1**' in each row. It stands for variable ***x_0***

Group Activity

Predictor$_1$ | Predictor$_2$ | Target

X_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	5.2	3.4	Explosion	1
1	4.5	4.9	Earthquake	-1
1	6.1	5.8	Earthquake	-1
1				
1	5.6	3.9	?	?

initial value for weight parameter W_0 { $w_0 = 0$

initial value for weight parameter W_1 { $w_1 = 0$

initial value for weight parameter W_2 { $w_2 = 0$

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

- Use the first training example and apply the perceptron learning rule to find the weight parameter values again

$$\Delta w_i = \eta(t - o)x_i$$
$$w_i^{new} = w_i^{old} + \Delta w_i$$

Iteration 1 (Step#1): Forward Pass

$w_0^{old} = 0$

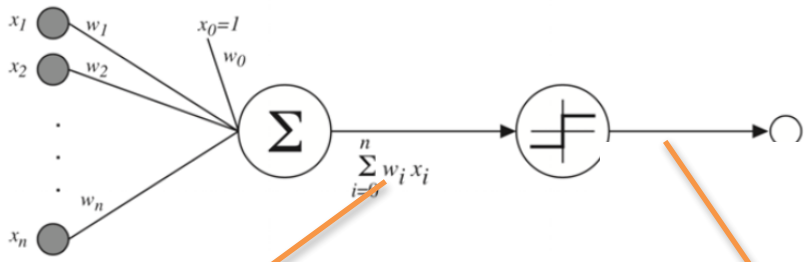
$w_1^{old} = 0$

$w_2^{old} = 0$

$\eta = 0.1$

Predictor$_1$ | Predictor$_2$ | Target

X_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	5.2	3.4	Explosion	1



x_0 is always 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	5.2	3.4	0*1+0*5.2+0*3.4 = 0	-1	1

Iteration 1 (Step#2): Backward Pass

$w_0^{old} = 0$

$w_1^{old} = 0$

$w_2^{old} = 0$

	Predictor$_1$	Predictor$_2$		Target
x_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	5.2	3.4	Explosion	1

$\eta = 0.1$

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

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

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	1	0.1x(1-(-1))x1= 0.1x2x1= 0.2	0+0.2= 0.2

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	-1	5.2	0.1x(1-(-1))x5.2= 0.1x2x5.2= 1.04	0+1.04= 1.04

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	-1	3.4	0.1x(1-(-1))x3.4= 0.1x2x3.4= 0.68	0+0.68= 0.68

Results After Iteration 1

	Predictor$_1$	Predictor$_2$		Target
X_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	4.5	4.9	Earthquake	-1

updated weight parameter W_0 { $w_0 = 0.2$

updated weight parameter W_1 { $w_1 = 1.04$

updated weight parameter W_2 { $w_2 = 0.68$

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

- Now use the second training example and apply the perceptron learning rule to find the weight parameter values again

$$\Delta w_i = \eta(t - o)x_i$$
$$w_i^{new} = w_i^{old} + \Delta w_i$$

Iteration 2 (Step#1): Forward Pass

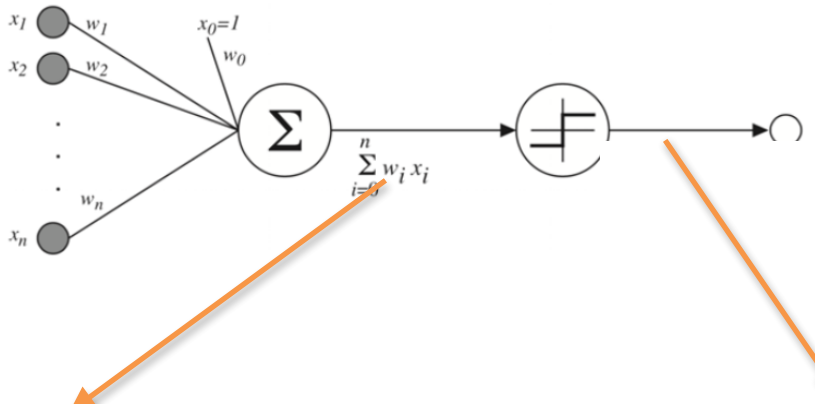
$w_0^{old} = 0.2$

$w_1^{old} = 1.04$

$w_2^{old} = 0.68$

$\eta = 0.1$

	Predictor$_1$	Predictor$_2$		Target
X_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	4.5	4.9	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	4.5	4.9	0.2*1+1.04*4.5+0.68*4.9 = 8.2	1	-1

Iteration 2 (Step#2): Backward Pass

$w_0^{old} = 0.2$

$w_1^{old} = 1.04$

$w_2^{old} = 0.68$

$\eta = 0.1$

	Predictor$_1$	Predictor$_2$		Target
X_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	4.5	4.9	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	1	0.1x(-1-1)x1= 0.1x(-2)x1= -0.2	0.2+(-0.2)= 0

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	1	4.5	0.1x(-1-1)x4.5= 0.1x(-2)x4.5= -0.9	1.04+(-0.9)= 0.14

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	1	4.9	0.1x(1-(-1))x4.9= 0.1x(-2)x4.9= -0.98	0.68+(-0.98)= -0.30

Results After Iteration 2

	Predictor$_1$	Predictor$_2$		Target
X_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	6.1	5.8	Earthquake	-1

Your turn to update weight parameters using the last example

updated weight parameter W_0 → $w_0 = 0.0$

updated weight parameter W_1 → $w_1 = 0.14$

updated weight parameter W_2 → $w_2 = -0.30$

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

- Now use the third training example and apply the perceptron learning rule to find the weight parameter values again

$$\Delta w_i = \eta(t - o)x_i$$
$$w_i^{new} = w_i^{old} + \Delta w_i$$

Your Turn for Iteration 3 (Step#1): Forward Pass

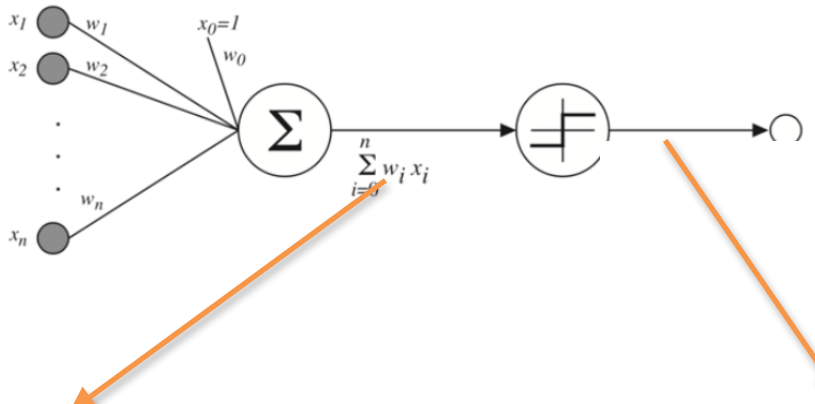
$w_0^{old} = 0.0$

$w_1^{old} = 0.14$

$w_2^{old} = -0.3$

$\eta = 0.1$

	Predictor$_1$	Predictor$_2$		Target
X_0	Body Wave Mag X_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	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 for Iteration 3 (Step#1): Backward Pass

$w_0^{old} = 0.0$

$w_1^{old} = 0.14$

$w_2^{old} = -0.3$

$\eta = 0.1$

	Predictor$_1$ Body Wave Mag X_1	Predictor$_2$ Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
X_0				
1	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

	Predictor$_1$	Predictor$_2$		Target
x_0	Body Wave Mag x_1	Surface Wave Mag x_2	Classification	Target (-1=earthquake, +1=explosion)
1	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

- In-class activity: Perceptron Learning Algorithm
- Expressive Power of a Perceptron

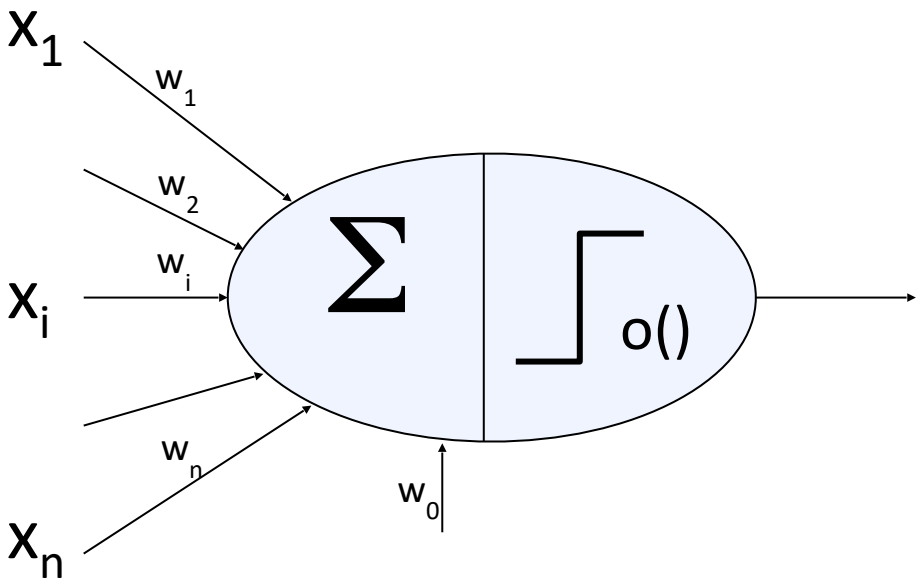
Perceptron

- 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



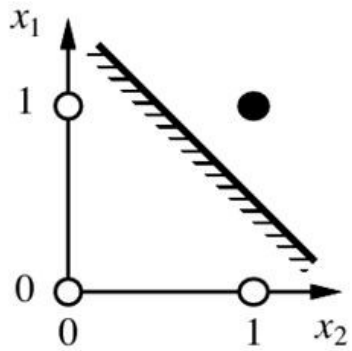
Big Question?
How do we determine the weights that best classify the data?

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

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

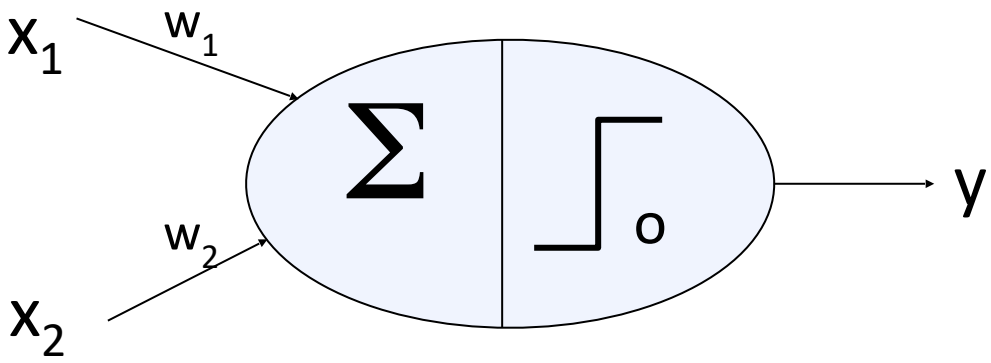


x_1 **and** x_2

YES, because we can find this 2D line

Color code:
Empty circle denotes 0
Black circle denotes 1

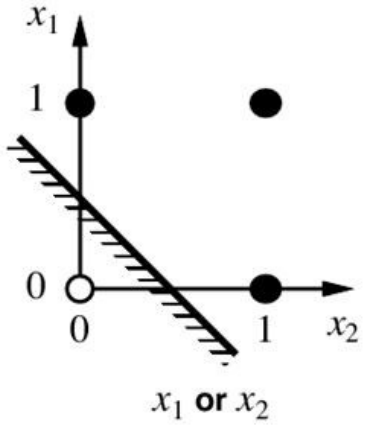
- Can a perceptron model AND function?



Perceptron can Model OR Function

- Let's consider the OR function.

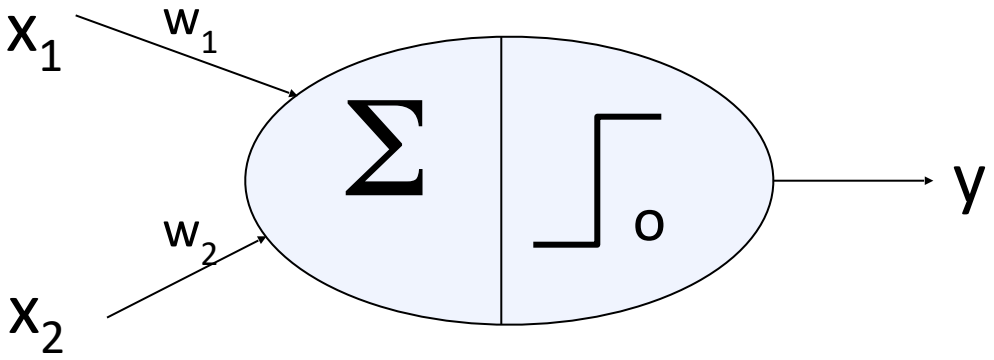
x	y	x or y
0	0	0
0	1	1
1	0	1
1	1	1



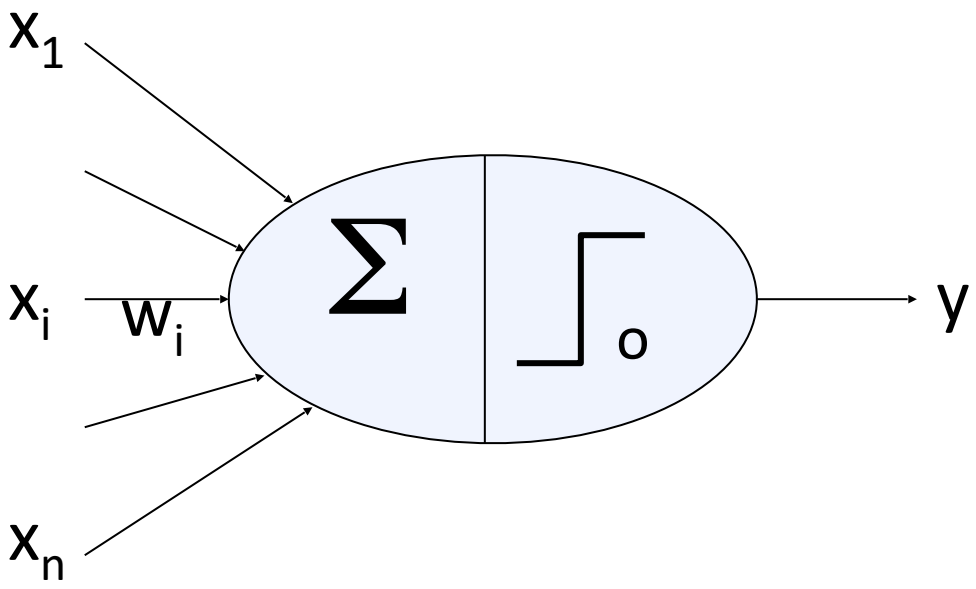
YES, because we can find this 2D line

Color code:
Empty circle denotes 0
Black circle denotes 1

- Can a perceptron model OR function?



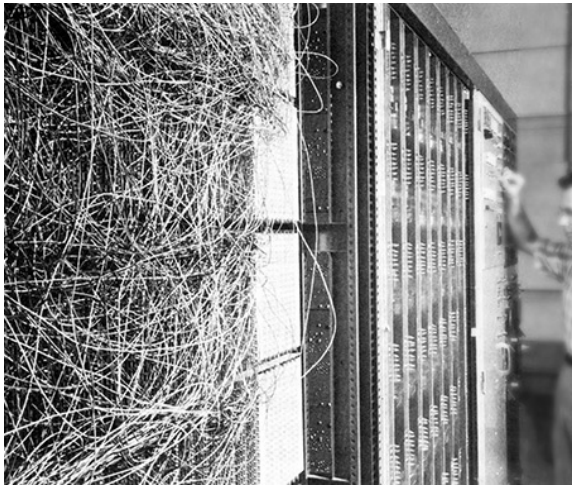
Can Perceptrons model any function?

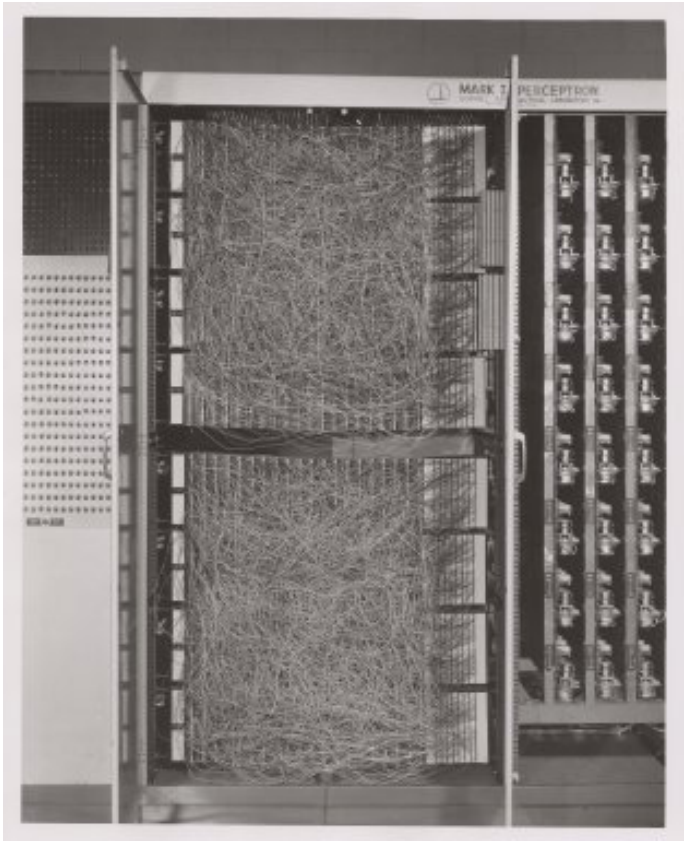


- Perceptron can also model other boolean functions such as ($x_1 \wedge x_2 \wedge \neg x_3$). We can tabulate all combinations of the three variables and their corresponding outputs; then find weight parameters (of the plane separating it two classes (1 and 0)) using perceptron update rule we just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

- But can a perceptron model any function?

Perceptron





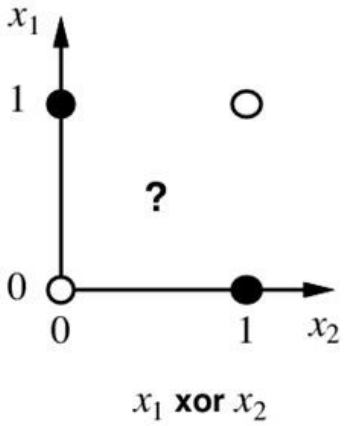
"the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

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

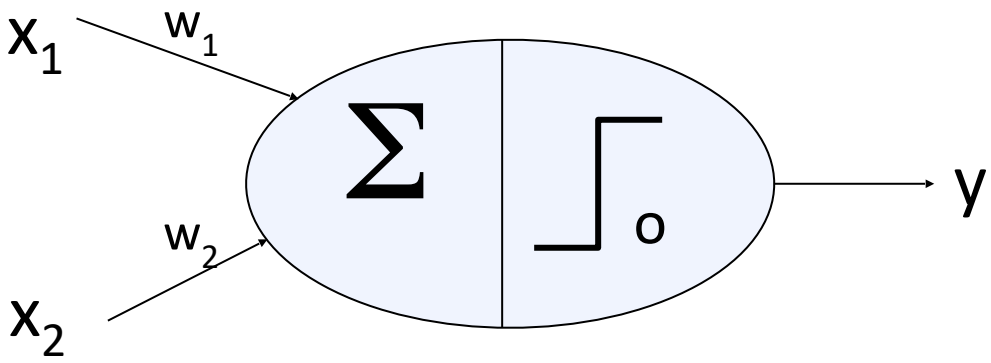


x_1 **xor** x_2

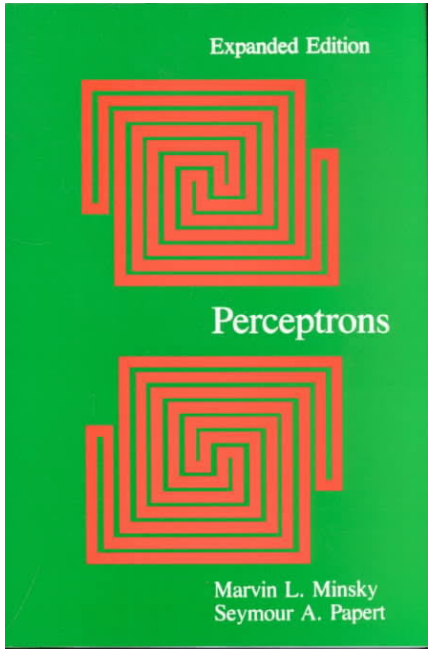
NO, because we can't find a single 2D line that separates the samples

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