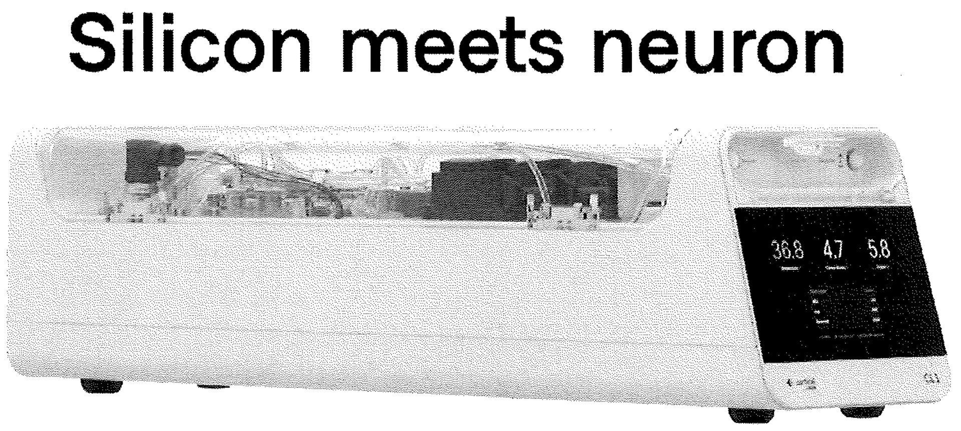
The Perceptron Learning Rule (a.k.a. The Perceptron Trick)

LIN 313 Language and Computers
UT Austin Fall 2025
Instructor: Gabriella Chronis

If ever the twain shall meet…

Introducing the world's first biological neural network: https://corticallabs.com/cl1



Real neurons are cultivated inside a nutrient rich solution, supplying them with everything they need to be healthy. They grow across a silicon chip, which sends and receives electrical impulses into the neural structure.

Admin

- HW 2 grades posted
- schedule re-arranged
  - more time on neural nets before midterm
  - word vectors after midterm
- study guide soon
- collab crib sheet

Overview 10/10

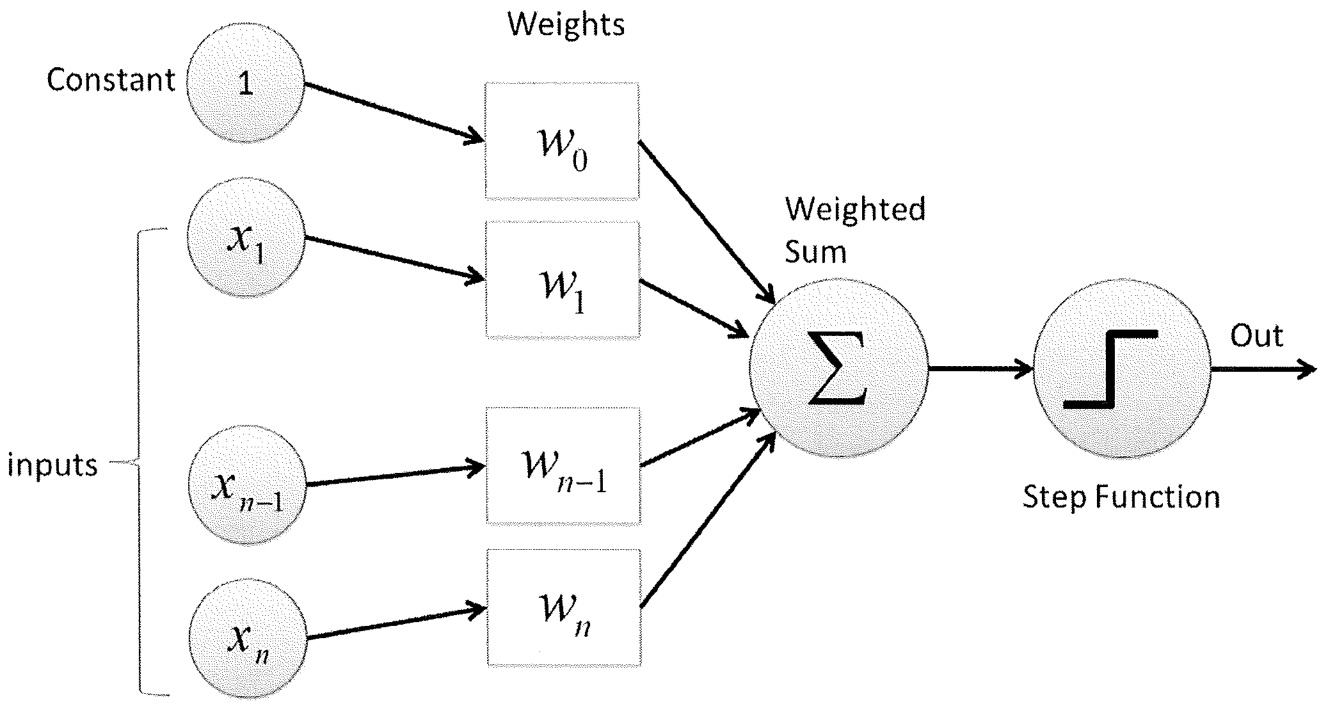
- The Perceptron
  - Applying the Perceptron algorithm (aka 'the forward pass')
  - Updating the Perceptron algorithm with the Perceptron trick (aka 'the backward pass')
- (or monday) Voting on classifiers

Pilot Annotation Results

results:

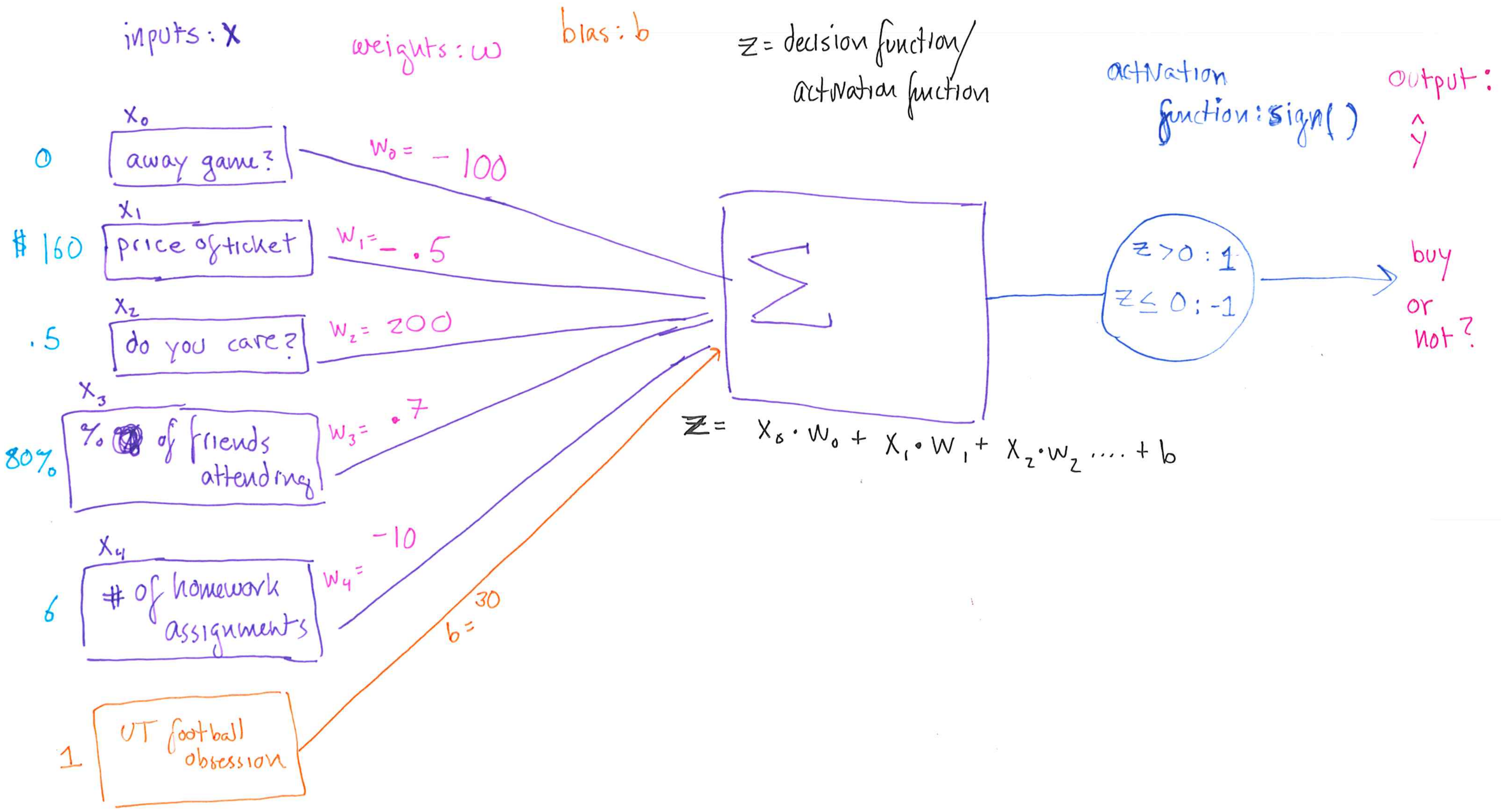
https://docs.google.com/spreadsheets/d/1c3AnPdg9rl0Q5ZDEljUy6t0FxV4g1_L8mUPW-irkZ20/

The Perceptron Algorithm



Perceptron Intuition

Should you buy a ticket to the next UT Game?



$$z = x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 \dots + b$$

The Perceptron Algorithm (again)

$$\boxed{\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)}$$

w is the weight vector $(w_1 w_2 \ldots w_i)$

x is the input vector $(x_1 x_2 \ldots x_i)$

b is the bias, still a regular number

sign( ) is the activation function

w · x is the dot product of **w** and **x**

the dot product is just shorthand for the weighted sum!

The Dot Product

Definition

$\mathbf{a} = \langle a_1, a_2, a_3 \rangle \quad \mathbf{b} = \langle b_1, b_2, b_3 \rangle$

$$\boxed{\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3}$$

Vectors

The equation gets simpler if we think of the input as vectors.

column vector

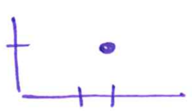
row vector

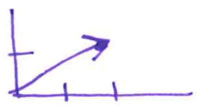
A vector is an ordered list of numbers. $[2, -1, 3]$ $\langle 2, -1, 3 \rangle$ $\begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}$

An ordered pair is a vector. $\mathbf{a}$ $\vec{a}$

An ordered pair has a **geometric interpretation**.

$\vec{x} = [2, 1]$





A vector is an ordered pair with more numbers

A vector also has a geometric interpretation

Vector Dot Products

The Dot Product

Definition

$\mathbf{a} = \langle a_1, a_2, a_3 \rangle \quad \mathbf{b} = \langle b_1, b_2, b_3 \rangle$

$$\boxed{\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3}$$

To calculate the dot product, multiply each cell in vector **a** with the corresponding cell in vector **b**, then add them together.

The dot product of two **vectors** is always a **scalar.**
A scalar is just a plain old number.

$$\vec{a} \cdot \vec{b} = \begin{array}{c|c} i= & \\ 1 & 2 \\ 2 & -1 \\ 3 & 3 \end{array} \cdot \begin{array}{ccc} i= 1 & 2 & 3 \\ 1 & 2 & 3 \end{array} = a_1 b_1 + \ldots\ldots$$

$$= 2\cdot 1 + (-1)(2) + 3\cdot 3$$

$$= 2 + (-2) + 9$$

$$= 9$$

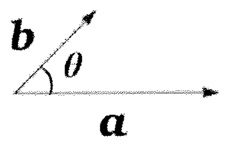
Meaning of the Dot Product

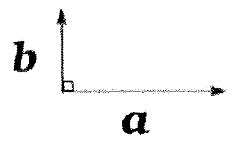
The dot product of two vectors tells you about what direction they are pointing.

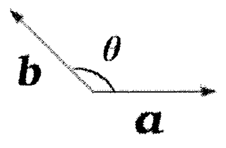
If the dot product is positive, the vectors are pointing in the same direction.

If the dot product is negative, the vectors are pointing in the opposite direction.

If the dot product is 0, the vectors are at right angles to each other.



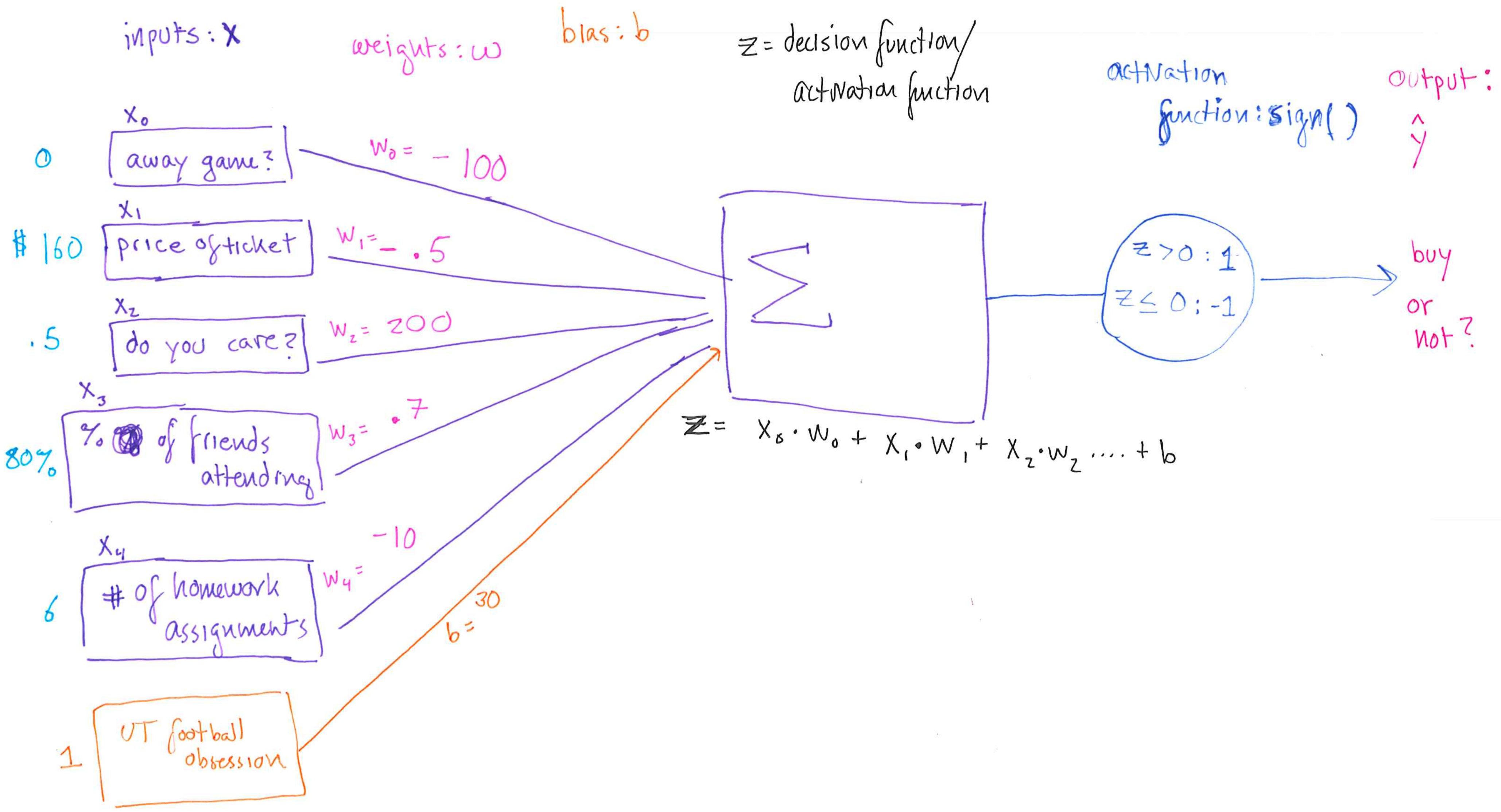




$a \cdot b > 0$ $a \cdot b = 0$ $a \cdot b < 0$

Perceptron Intuition

Should you buy a ticket to the next UT Game?



Putting it all together

What does a **forward pass** through the Perceptron look like for our UT example?

$$\boxed{\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)}$$

It's an away game. two of my friends are actually going, but I'm broke and I have two write an essay and finish a lab write up this weekend.

$$\mathbf{w} = \begin{array}{ccccc} [-100, & -0.5, & 200, & 0.7, & -10] \\ 1 & 2 & 3 & 4 & 5 \end{array}$$

$$\mathbf{x} = \begin{array}{ccccc} [1, & 160, & 0.8, & 60, & 6 \quad] \\ 1 & 2 & 3 & 4 & 5 \end{array}$$

$$b = 30$$

$$\vec{w} \cdot \vec{x} = (-100)\cdot 1 + (-0.5)\cdot 160 + 200\cdot(0.8) + (0.7)\cdot 60 + (-10)\,6$$

$$= -100 + (-80) + 160 + 42 + (-60)$$

$$= -38$$

$$\hat{y} = \text{sign}(-38 + 30)$$

$$= \text{sign}(-8)$$

$$= -1$$

Putting it all together

It's a home game, the weather is balmy. I have a ton of homework but all of my friends are going.

What does a **forward pass** through the Perceptron look like for our UT example?

$$\boxed{\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)}$$

$$\mathbf{w} = [\underset{1}{-100}, \underset{2}{-0.5}, \underset{3}{200}, \underset{4}{0.7}, \underset{5}{-10}]$$

$$\mathbf{x} = [\underset{1}{0}, \underset{2}{90}, \underset{3}{0.7}, \underset{4}{100}, \underset{5}{10}]$$

$$b = 30$$

$$\vec{W} \cdot \vec{X} = \begin{bmatrix} 0 \\ 90 \\ 0.7 \\ 100 \\ 10 \end{bmatrix} \cdot [-100, -0.5, 200, 0.7, -10]$$

$$= 0 \cdot (-100) + 90(-0.5) + 0.7(200) + 100(0.7) + 10(-10)$$

$$= 65$$

$$\hat{y} = \text{sign}(65 + 30)$$

$$= \text{sign}(90)$$

$$= 1$$

Observation

There is a relationship between the *signs* of the weights and inputs and the *sign* of the dot product.

w_i	x_i	$x_i w_i$ is ____. Dot product is pushed towards ____
+	+	+
-	-	+
-	+	-
+	-	-

When the signs at $x_i w_i$ match, that connection pushes the output towards YES

Perceptron Learning

Perceptron learning rule
If $\hat{y} = y$, do nothing.
Otherwise, for each w_i in $\mathbf{w}$:
$w_i' = w_i + \alpha * y * x_i$

$\mathbf{w}_i'$:	the new weight at index i (pronounced 'w eye prime')
$\mathbf{w}_i$:	the current weight at index i
α :	the learning rate (alpha) usually small
$\mathbf{y}$:	the ~~incorrect prediction~~ actual value
$\mathbf{x}_i$:	the input at index i

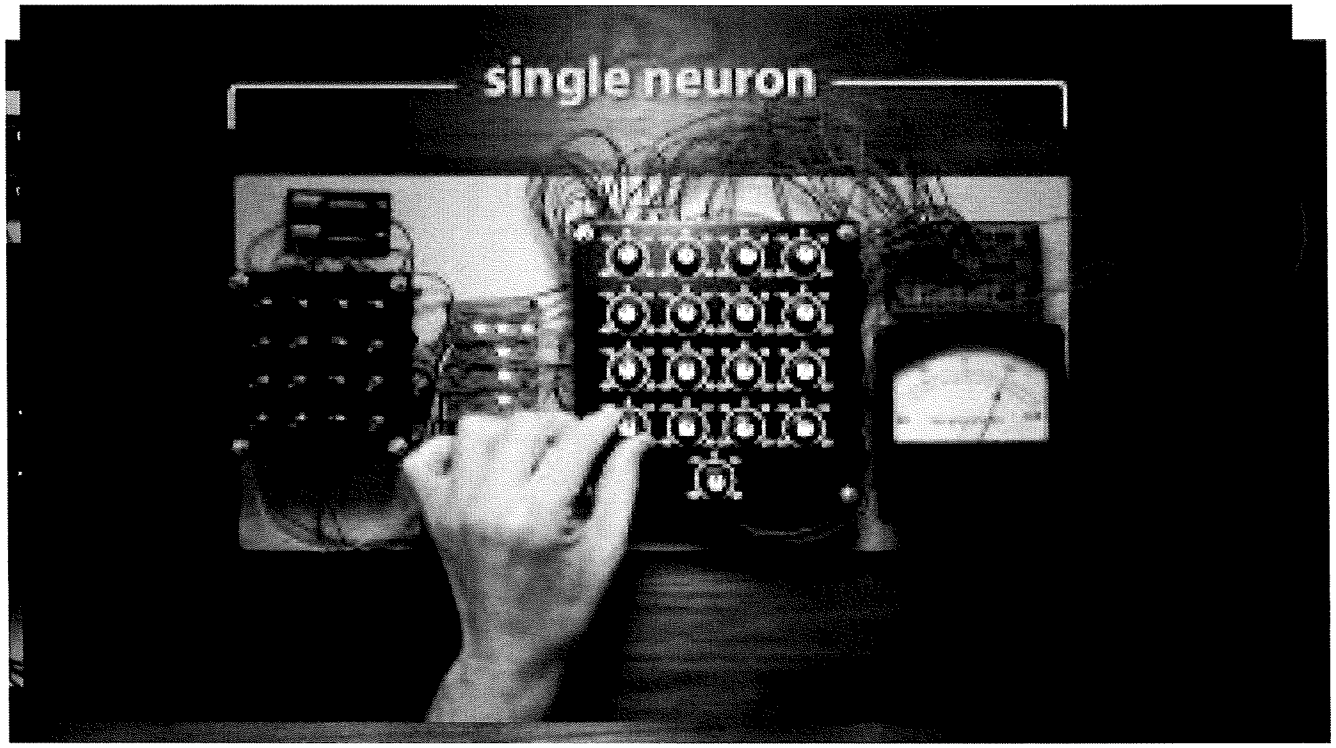
In English:

If we guessed the right answer, leave everything as is.

If we guessed the wrong answer, then update all of the weights in the network a to look a little bit more like the input at that weight. That is, we move the weights a little bit towards a correct prediction for that input.

Visualizing the Perceptron rule with dials as weights

When the perceptron incorrectly outputs POS for a J shape, turn all of the dials for ON inputs a little bit towards NEG,and turn all of the dials for OFF inputs towards positive



Learning AND with the Perceptron Trick

We have a Perceptron with two weights. We want it to learn the logical AND function. (That is, we want to output 1 if and ONLY if w_1 =1 AND w_2 = 1.)

$$\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

Perceptron learning rule
If $\hat{y} = y$, do nothing. Otherwise, for each w_i in $\mathbf{w}$: $w_i' = w_i + \alpha * y * x_i$

initial settings:

- learning rate α = 1
- **w** = [0, 0]
- b = 0

x_1	x_2	$\hat{y}$

Training on real data

One run through training data through the update rule is called an "epoch."

We typically train neural nets with multiple epochs, i.e. the algorithm sees each training pair several times.

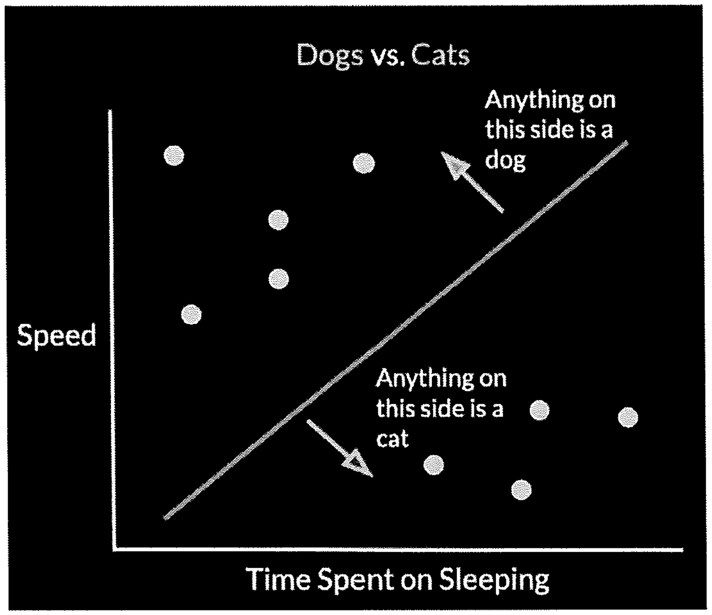
Geometric Interpretation

we can make decision function (the weighted sum) into an equation for a line:

$$w_1 x_1 + w_2 x_2 + b = 0$$

Rewrite that line in standard y=mx+b form

$$x_2 = -(w_1/w_2)x_1 - (b/w_2)$$


Dogs vs. Cats
Anything on this side is a dog
Speed
Anything on this side is a cat
Time Spent on Sleeping

https://karthikvedula.com/2024/01/05/visualizing-the-perceptron-learning-algorithm/

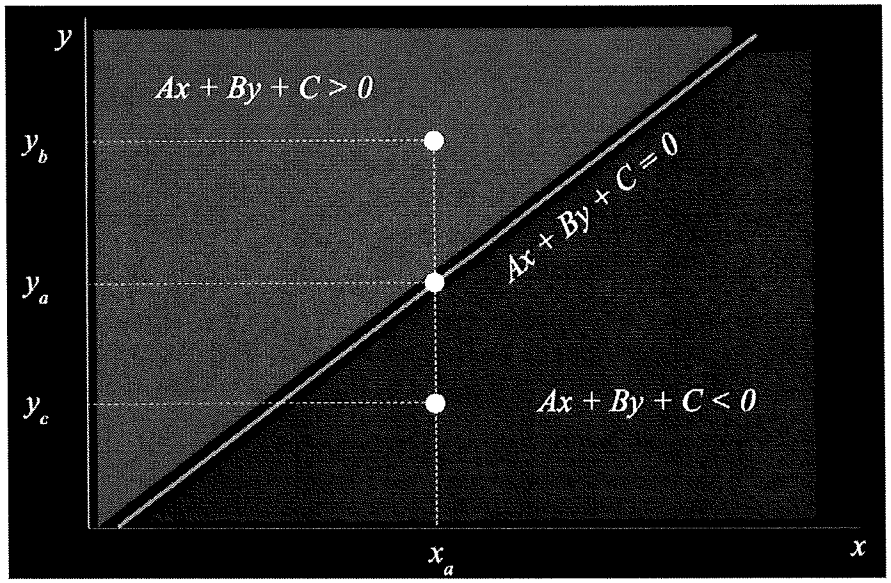
Geometric Interpretation

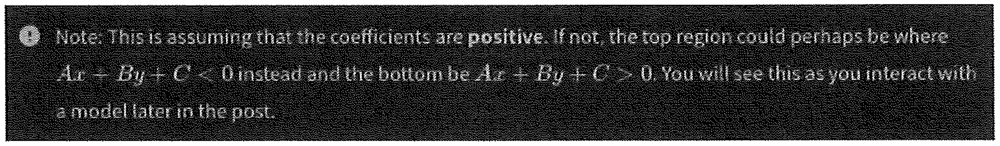
we can make decision function (the weighted sum) into an equation for a line:

$$w_1x_1 + w_2x_2 + b = 0$$

Rewrite that line in standard y=mx+b form

$$x_2 = -(w_1/w_2)x_1 - (b/w_2)$$




Note: This is assuming that the coefficients are **positive**. If not, the top region could perhaps be where $Ax + By + C < 0$ instead and the bottom be $Ax + By + C > 0$. You will see this as you interact with a model later in the post.

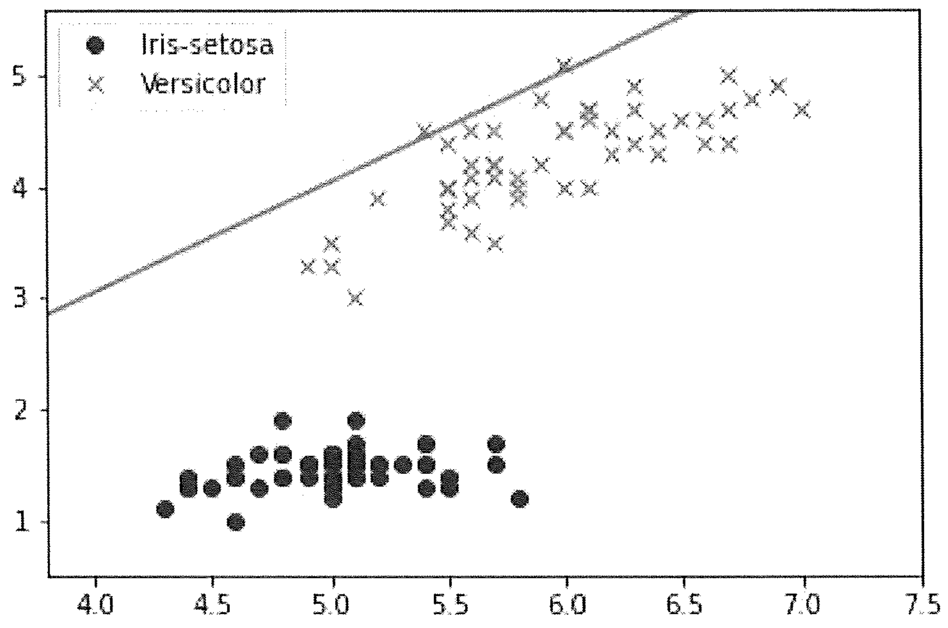
https://karthikvedula.com/2024/01/05/visualizing-the-perceptron-learning-algorithm/

Perceptron Learning

The model converges when it stops updating.

The perceptron learning rule is guaranteed to **converge** if the data are linearly separable

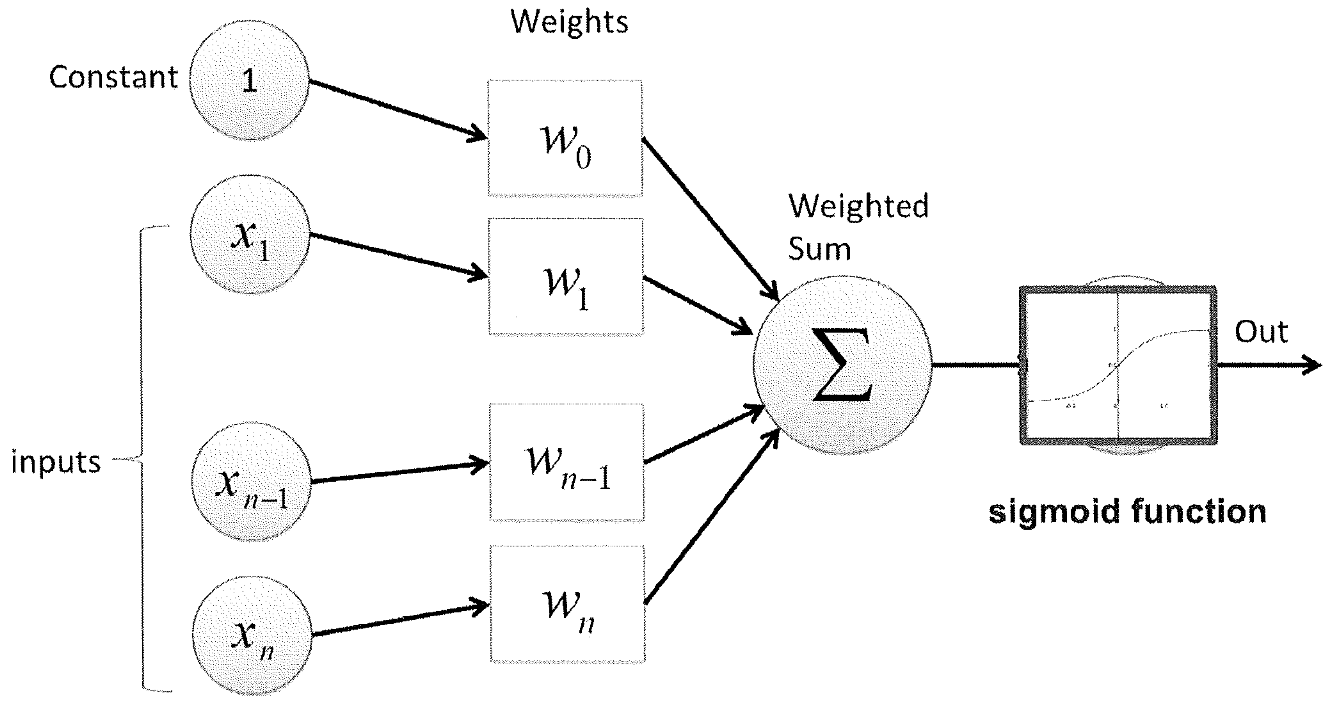
linear seperability



Linear Regression

The only difference is the activation function that we choose!

Linear regression uses the **sigmoid** activation function



Interactive Demos

- perceptron: https://karthikvedula.com/2024/01/05/visualizing-the-perceptron-learning-algorithm/
- perceptron: https://perceptron.streamlit.app/
- dot product: https://maththebeautiful.com/dot-product/

Pilot Annotation Results

results:

https://docs.google.com/spreadsheets/d/1c3AnPdg9rl0Q5ZDEljUy6t0FxV4g1_L8mUPW-irkZ20/

Vote:

https://strawpoll.com/e7ZJabzWdg3