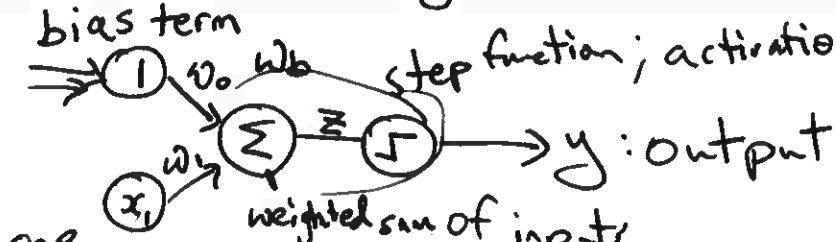
Good afternoon! Happy Thursday!

Perceptrons Part 2

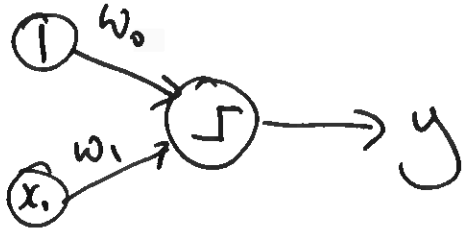
- review + terminology
- perceptron learning rule

Question of the day: Will I spill my tea again??? (Who can say?)



$$y = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i > 0 \\ 0 & \end{cases}$$



$$y = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \end{cases}$$

$$z = \sum_i w_i x_i$$

example: $w_0 = 3$ $w_1 = 2$

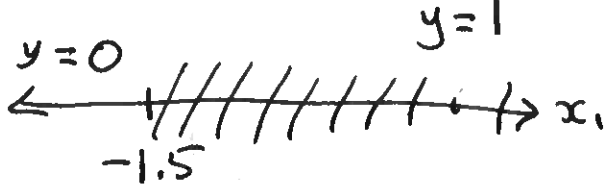
this corresponds to a specific equation & also a specific picture

$$y = \begin{cases} 1 & \text{if } 3 + 2 \cdot x_1 > 0 \\ 0 & \text{else} \end{cases}$$

$3 + 2x_1 > 0$

$2x_1 > -3$

$x_1 > -\frac{3}{2}$

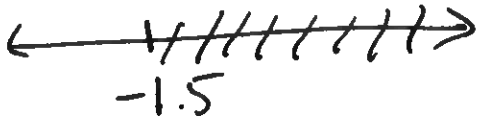


decision boundary!

binary & decision boundary in 1D

$y = 0, 1$

given a specific set of weights, a perceptron defines exactly <u>one</u> decision boundary.

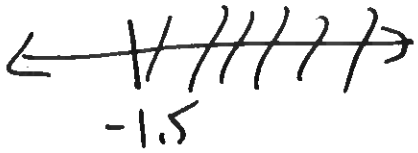


given a specific decision boundary, how many possible sets of weights could we use?

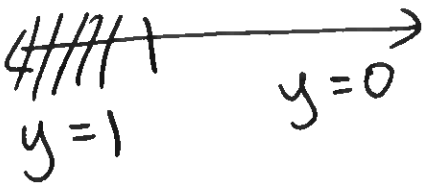
$w_0 = 3 \quad w_1 = 2$

$x_1 > -1.5$

$w_0 = 1.5 \quad w_1 = 1 \quad 1.5 + 1x_1 > 0$



$w_0 = 3 \quad w_1 = 2$



$w_0 = -3 \quad w_1 = -2$

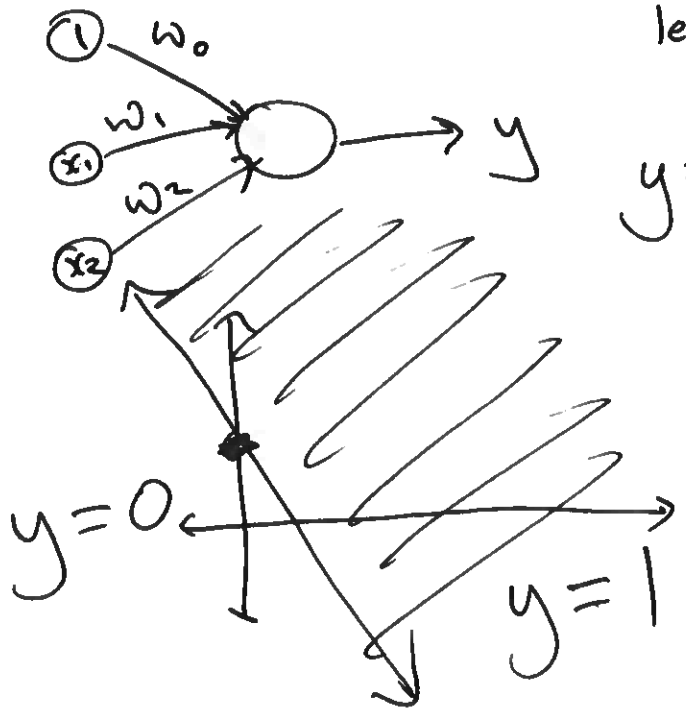
$$y = \begin{cases} 1 \text{ if } -3 - 2x_1 > 0 \\ \end{cases}$$

$$-2x_1 > 3$$

$$x_1 < -\frac{3}{2}$$

---

also do this in 2D



let: $w_0 = -1 \quad w_1 = 2 \quad w_2 = 1$

$$y = \begin{cases} 1 \text{ if } -1 + 2x_1 + 1x_2 > 0 \\ \end{cases}$$

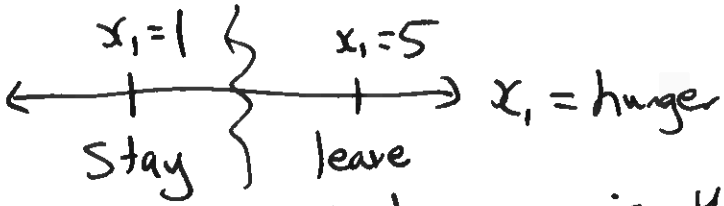
$$1x_2 > -2x_1 + 1$$

slope-intercept form of a line!

Def: a perceptron with n inputs and n+1 weights defines a decision boundary in an n-dimensional space. This d.b. is linear

if 1D: d.b. is a point
2D: d.b. is line
3D: d.b. is plane
higher D: hyperplane

Learning: What if we have some observations of x and y ("data") and we want to find a good set of weights?



two data points!

is there an automated procedure for finding a good set of weights?

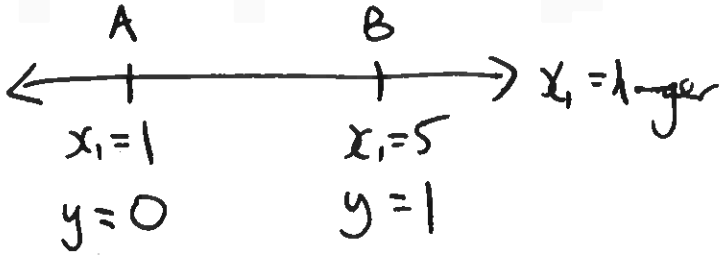
YES. iterative procedure: change weights step by step until the perceptron does what we want.

weight update rule: $\bar{w}_{t+1} = \bar{w}_t + \Delta\bar{w}$

time t+1 (steps in the iterative process)

$$\bar{w}_{new} = \bar{w}_{old} + \Delta\bar{w}$$

☆ What should Δw be???



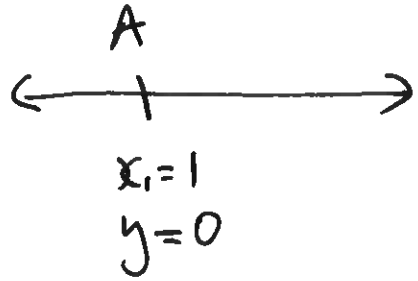
Step 1: Start with a random perceptron.

$w_0 = 3 \quad w_1 = 2$ is this correct, given our data?

data	x_1	y_{true}	perceptron w_0	w_1	$w_0 + w_1 x_1$ z	$y_{predicted}$	correct?	$y_{true} - y_{pred}$ error
A	1	0	3	2	$3 + 2 \cdot 1 = 5$	1	X	−1
B	5	1	3	2	$3 + 2 \cdot 5 = 13$	1	✓	0

Step 2: how can we tweak the weights to make this perceptron less wrong?

idea: use error as a clue!



$$\underbrace{w_0 + w_1 x_1}_{\text{want smaller!}} > 0$$

if error is positive, want to move threshold to left
negative, right

$\boxed{\Delta w_i = \text{error}}$ idea # 1

step 1: $w_0 = 3 \quad w_1 = 2 \quad 3 + 2 \cdot 1 > 0$

2: 2 1 $2 + 1 \cdot 1 \quad 3 > 0$

3: 1 0 $1 + 0 \cdot 1 \quad 1 > 0$

4: 0 −1 $0 + -1 \cdot 1 : -1 < 0$

$w_0 + w_1 x_1 > 0$

make this smaller

this only worked b/c x_1 was postive

if x_1 was negative, then need to increase w_1 to make the whole term smaller

$$\Delta w_i = (\text{error})(x_i)$$ idea #2

- takes care of the sign of the error
- takes care of the sign of the input
- takes care of the magnitude of the input

$w_0 + w_1 x_1 > 0$

idea #3: add "learning rate" η

constant parameter to scale the size of the weight changes

some

$$\Delta w_i = \eta (\text{error}) x_i$$

$$\Delta w_i = \eta (y_{true} - y_{predicted}) x_i$$

$$w_{new} = w_{old} + \Delta w$$

perceptron learning rule

(perceptron update rule)

- perceptron update rule is guaranteed to converge to a correct solution for <u>any</u> n-dimensional dataset that is linearly separable!!!

things you should know how to do:

- calculate output of a perceptron
- go back & forth between eq., perceptron, drawing view in 1D or 2D
- write down the P. update rule
- calculate the P. update rule