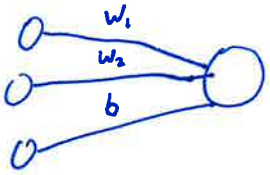
# Limitations of perceptrons

Last time:




Artificial neuron "perceptron"

Input: $x_1, x_2, \ldots, 1$ (stimulus) — extra "bias" stimulus

Output: 0 or 1 (activation), Weights: $w_1, w_2, b$

- Activate when $\sum x_i w_i + b \geq 0 \rightarrow \hat{y}$

Update: $w_i \leftarrow w_i + (y - \hat{y}) x_i$

Learn from dataset:

$(x_1, y_1), (x_2, y_2) \ldots (x_N, y_N)$

$x_{3,1}$ — example, input

Accuracy: Correct if $y = \hat{y} \Rightarrow \frac{N_{correct}}{N}$

"fraction of times we responded correctly"

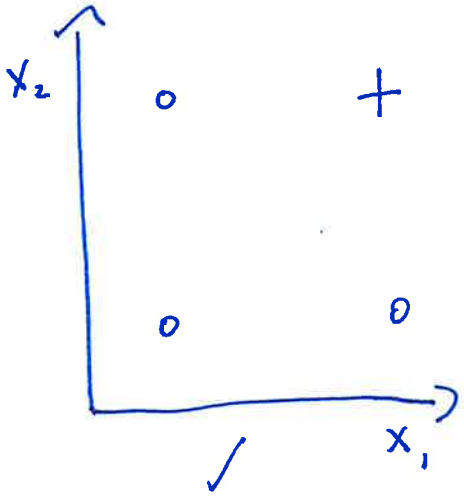
In practice: $\underbrace{(x_1, y_1)(x_2, y_2) \ldots (x_i, y_i)}$ | $\underbrace{(x_{i+1}, y_{i+1}) \ldots (x_N, y_N)}$

Update only on these — Compute accuracy on these

# What rules can we learn with a perceptron?

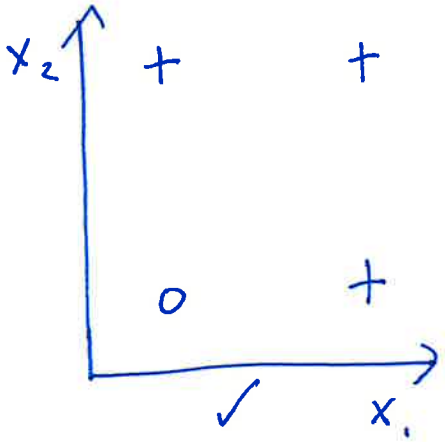
## And

| $X_1$ | $X_2$ | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



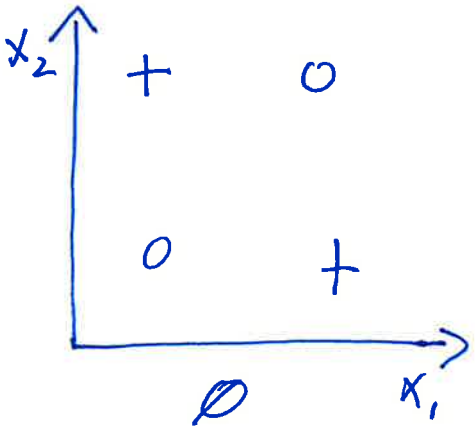

## or

| $X_1$ | $X_2$ | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |




## X or

| $X_1$ | $X_2$ | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

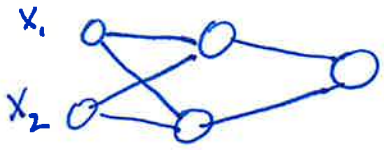



Recall: decision boundary is a line [What weights?]

Perceptrons are useless! ← Symbolic AI people in the 60's

Many problems need a non-linear decision rule

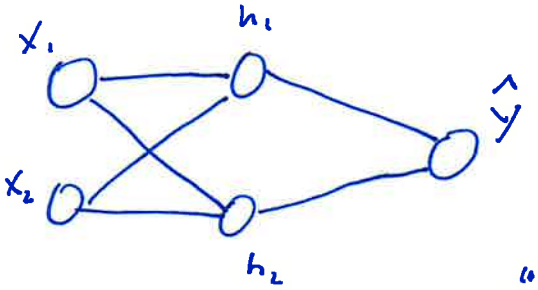
Brain Solution: Neurons that feed into other neurons!




Multi-Layer perceptron!
(MLP)

- +

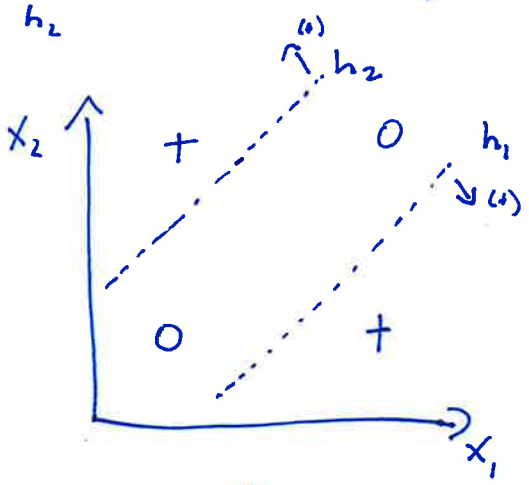
# Can it solve XOR?




$h_1$ $\quad x_1 w_{11} + x_2 w_{12} + b_1 > 0$

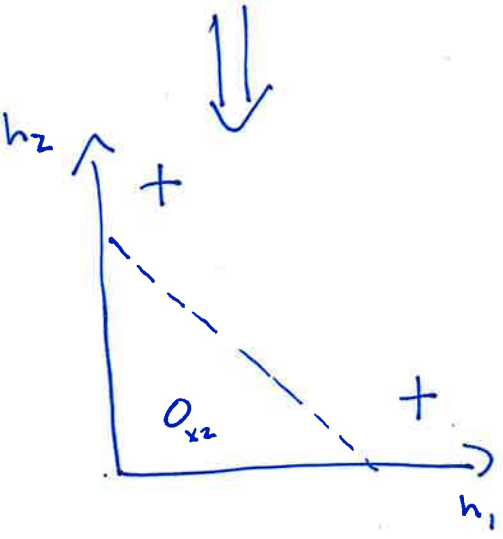
$h_2$ $\quad x_1 w_{21} + x_2 w_{22} + b_2 > 0$

$\hat{y}$ $\quad h_1 w_1 + h_2 w_2 + b > 0$

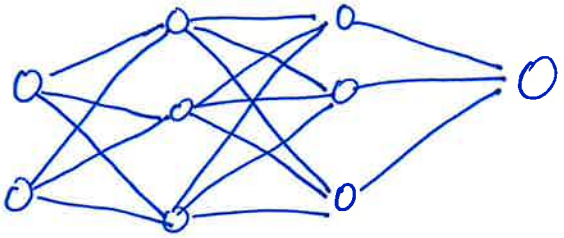



| $h_1$ | $h_2$ | $y$ | |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 1 | |
| 1 | 1 | 1 | ← Not possible |

↑ or table!

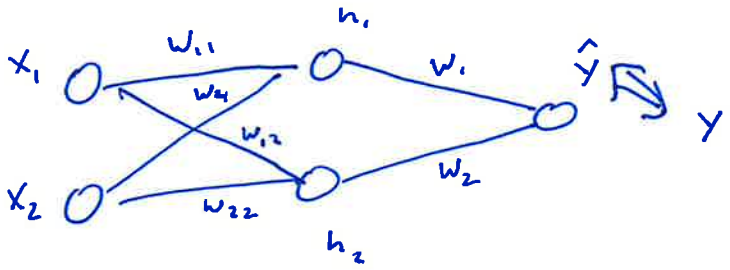



# In general:



Can stack many neurons!

But how do we learn?




Perceptron rule: Compare target to prediction, update if wrong

$$w_1 \leftarrow w_1 + \gamma(y-\hat{y})\, h_1$$

($h_1$ — effect on $w$ on $\hat{y}$; $(y-\hat{y})$ — "Error" — How we should change $\hat{y}$)

Works for $w_1/w_2$, but what about the first layer?

$$w_{11} \leftarrow w_{11} + \gamma(\quad - h_1)\, x_1$$

What's the target? $\Rightarrow$ How do we want to change $h_1$

Insight: we need $h_1$ to be something that's going to push $\hat{y}$ closer to $y$. $w_{11} \leftarrow w_{11} + \gamma(y-\hat{y})$ [effect of $w_{11}$ on $\hat{y}$]

Problem: Right now activations are binary $h_1 \in \{0,1\}$, so best we can do is set $w_{11}$ to swap it

Change: Let's relax our activations

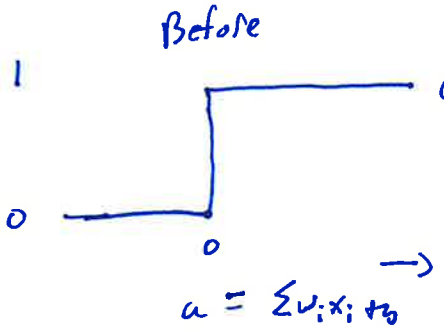
Before: Neuron fires or doesn't

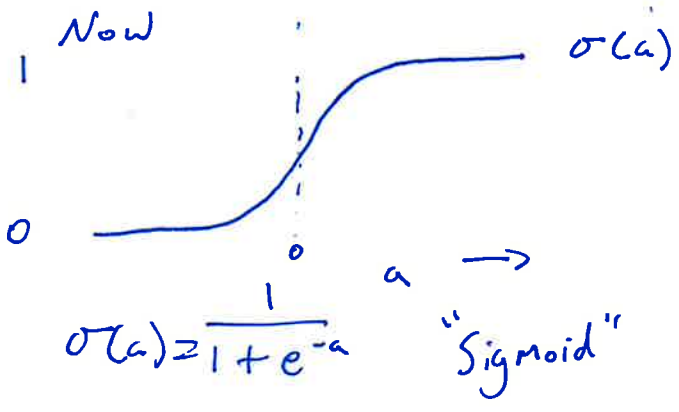
Now: Neuron fires some amount $\geq 0$

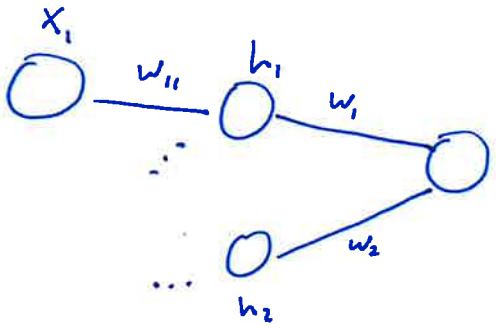
$\hat{y}$

before: $\hat{y} \in \{0,1\}$

Now: $\hat{y} \in [0,1]$






Output:

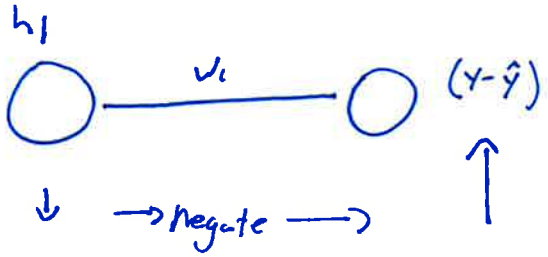
$\hat{y} = \sigma(a)$

$a = w_1 h_1 + w_2 h_2 + b$

Update for $w_1$:

$$w_1 \leftarrow w_1 + \gamma (y - \hat{y}) \sigma(a) \left[ \begin{array}{l} \text{Effect of } w_1 \text{ on } \sigma(a), \\ \text{depends on } h_1 \end{array} \right]$$

Push toward 0 → (+): Increase $\sigma(a)$; (−): decrease $\sigma(a)$




What is effect of $w_1$ on $\sigma(a) = \hat{y}$ → derivative! $\frac{\partial \hat{y}}{\partial w_1}$

Calculus rules

$\frac{d}{dx} C = 0$

$\frac{d}{dx} nx = n$

$\frac{d}{dx}(g(x) + h(x)) = \frac{dg}{dx} + \frac{dh}{dx}$

$\frac{d}{dx} g(x)^n = n g(x)^{n-1} \frac{dg}{dx}$

$\frac{d}{dx} e^u = e^u \frac{du}{dx}$, $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$ ← new

$$\frac{d}{dw_1}\left[\sigma(\underbrace{w_1 h_1 + w_2 h_2 + b}_{a})\right]$$

$$= \sigma(a)(1-\sigma(a))[h_1]$$

$$w_1 \leftarrow w_1 + \gamma (y - \hat{y}) \frac{\partial \hat{y}}{\partial w_1}$$

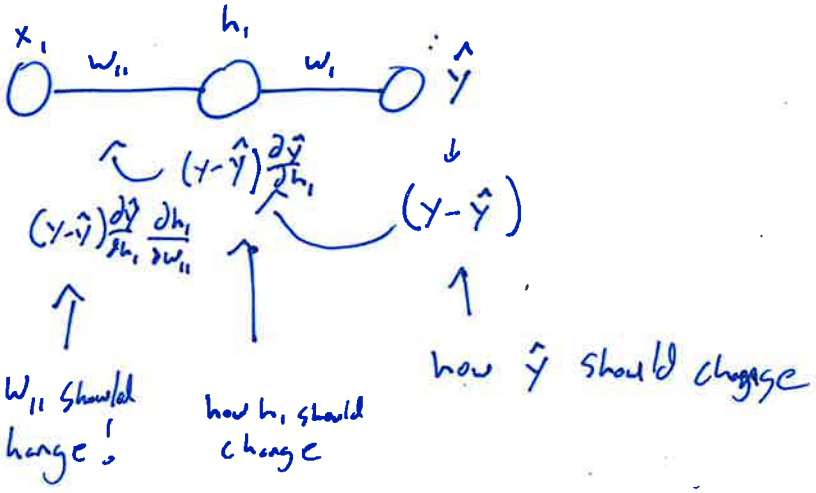
How to change $w_1$

Now for $h_1$: What is our "error", how do we want $h_1$ to change?

$h_1$: $(y - \sigma(a))\Big[\underbrace{\text{effect of } h_1 \text{ on } \sigma(a)}\Big]$

$$\frac{\partial \sigma(a)}{\partial h_1} = \sigma(a)(1-\sigma(a))\, w_1$$

Update for $w_{11}$:




Notes: This is Backpropagation!

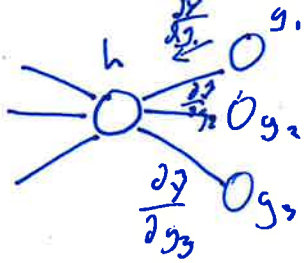
In general $(y-\hat{y})\frac{\partial \hat{y}}{\partial h}$ → How to change neuron h (how it affects error)




Recursion: Have $(y-\hat{y})\frac{\partial \hat{y}}{\partial g}$

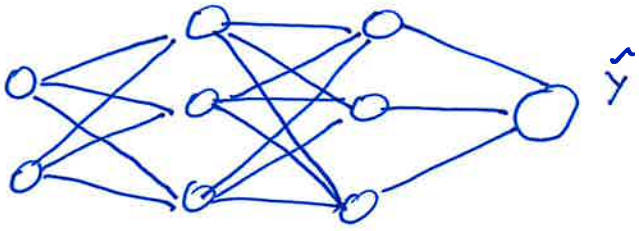
$$(y-\hat{y})\frac{\partial \hat{y}}{\partial h} = (y-\hat{y})\frac{\partial \hat{y}}{\partial g} \times \frac{\partial g}{\partial h}$$

General neuron: Multiple inputs and outputs




$$\frac{\partial \hat{y}}{\partial h} = \frac{\partial \hat{y}}{\partial g_1}\frac{\partial g_1}{\partial h} + \frac{\partial \hat{y}}{\partial g_2}\frac{\partial g_2}{\partial h} + \frac{\partial \hat{y}}{\partial g_3}\frac{\partial g_3}{\partial h}$$

↳ Same calculation!




Update

1) Compute $\hat{y}$, store $\sigma(a)$ at every neuron

2) Compute $(y-\hat{y})$, $\frac{\partial \hat{y}}{\partial \hat{y}} = 1$

3) For each layer (let $g$ be output $h$ be input)

Input: $(y-\hat{y})\frac{\partial \hat{y}}{\partial g}$ for each $g$

update $w_i$:

$$w_i \longleftarrow w_i + \gamma(y-\hat{y})\frac{\partial \hat{y}}{\partial g}\frac{\partial g}{\partial w}$$

update $(y-\hat{y})\frac{\partial \hat{y}}{\partial h}$