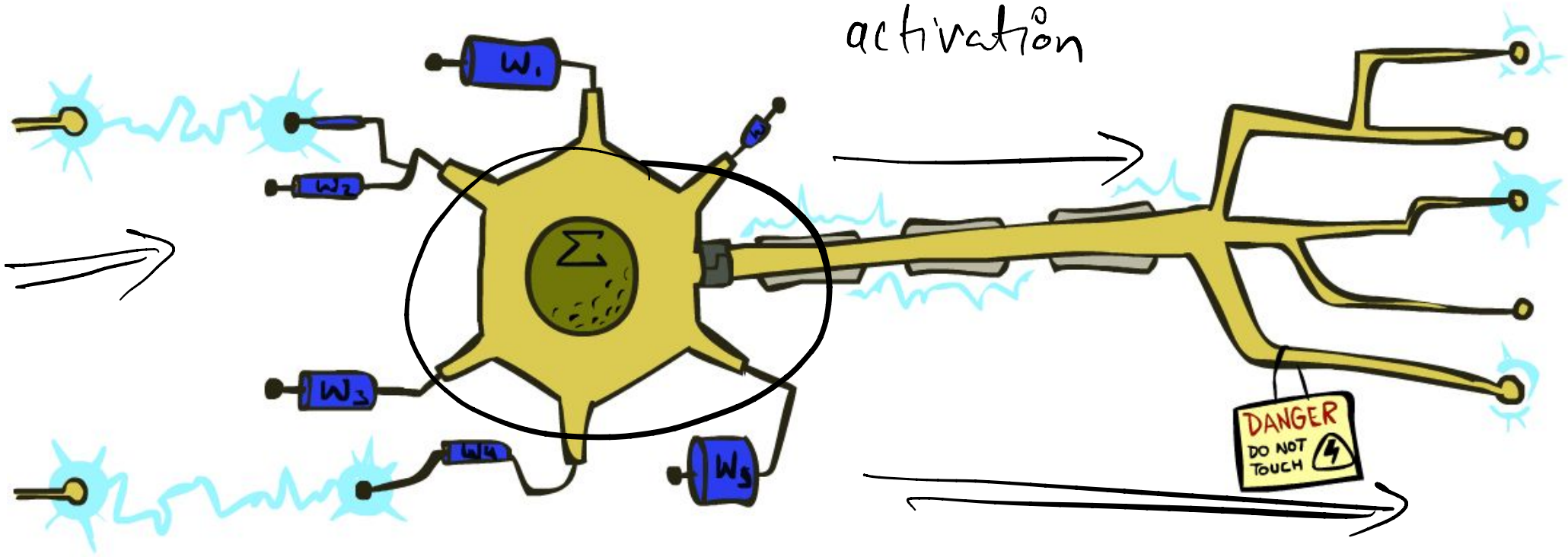
# Perceptrons and Logistic Regression

neural network

deep learning

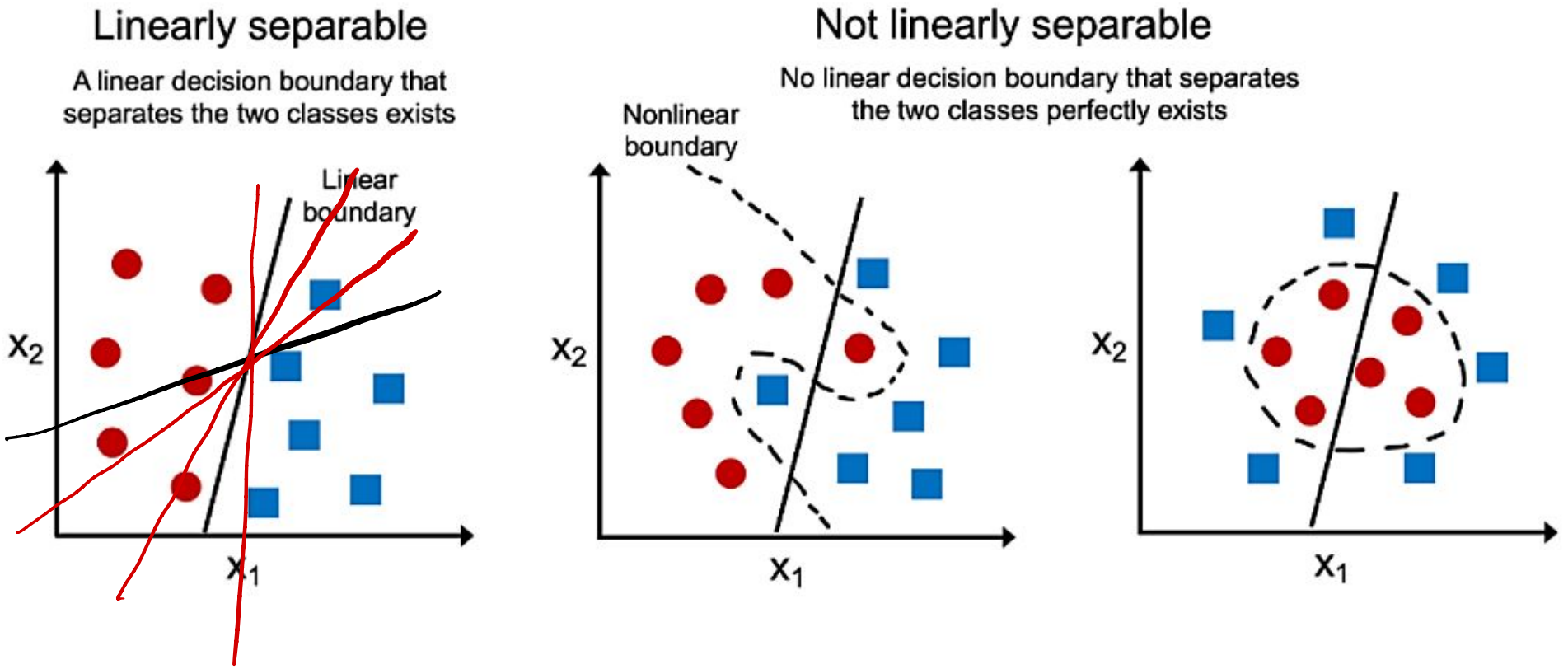
Machine learning

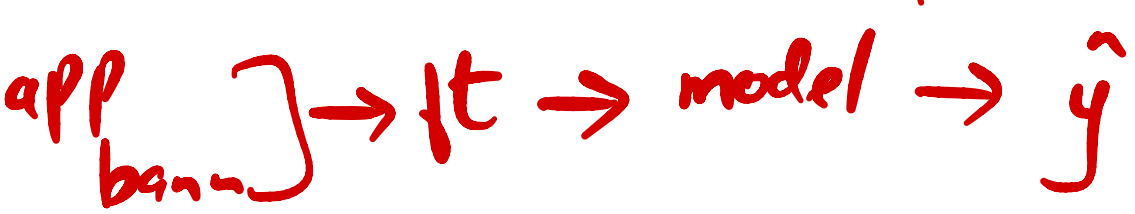
## Perceptrons and Logistic Regression



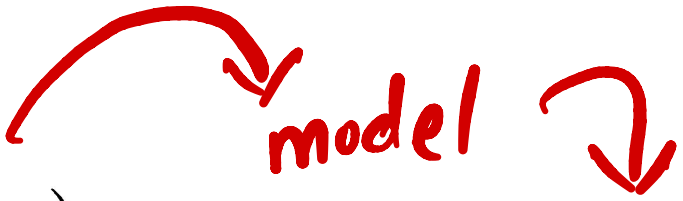
## Classifiers

equation ⇒ parameters
↓
weights and bias






## Feature Vectors




$x$ $f(x)$ $y$

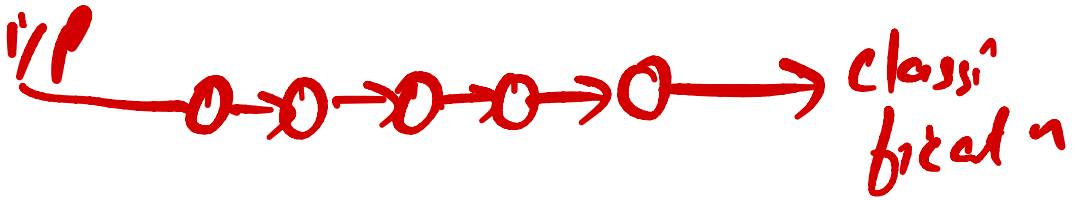
```
Hello,

Do you want free printr
cartriges?  Why pay more
when you can get them
ABSOLUTELY FREE!  Just
```

```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

SPAM
or
+

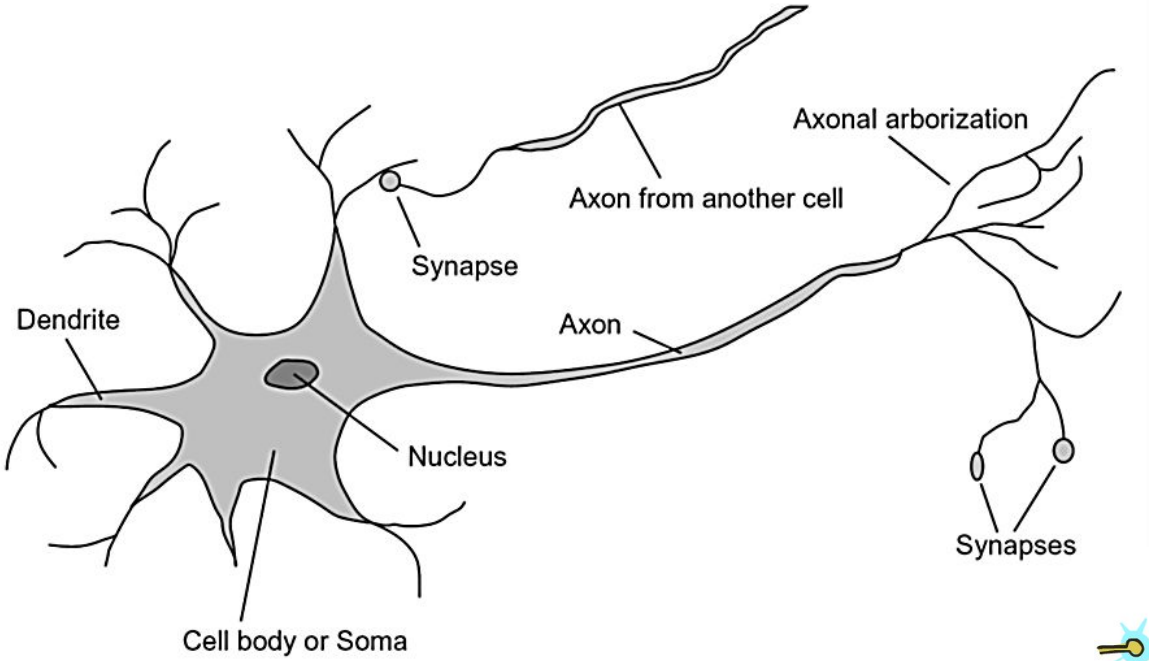
```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

“2”

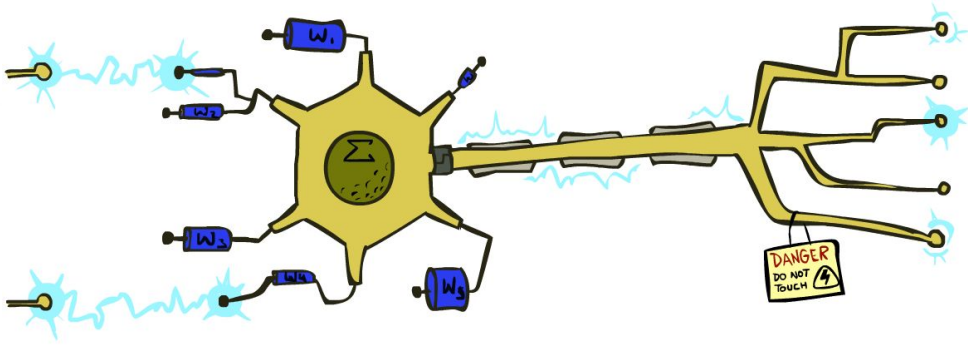



## Some (Simplified) Biology

- Very loose inspiration: human neurons




Perceptron is a single neuron

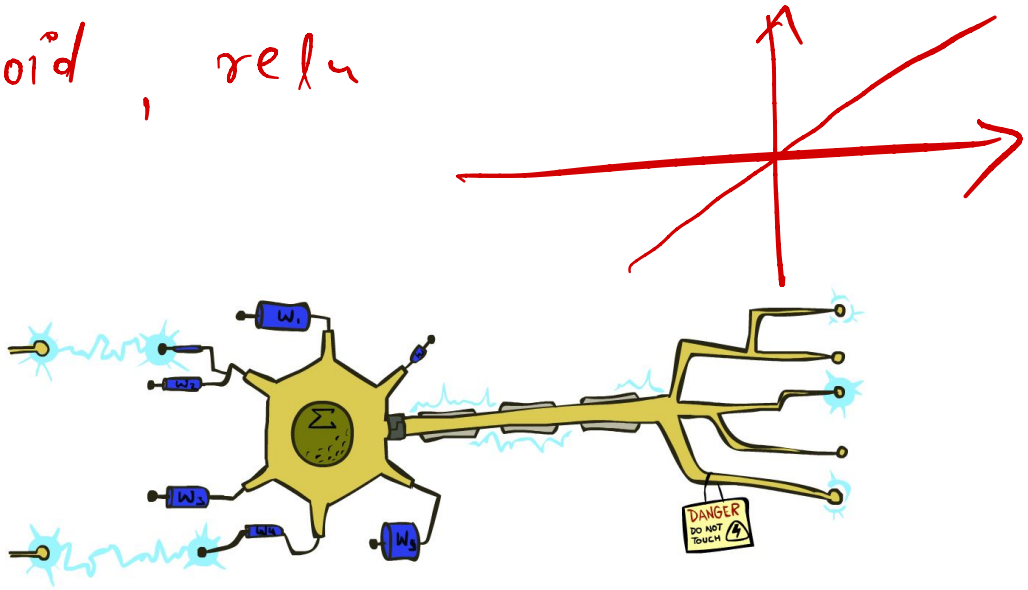


linear

sigmoid , relu

## Linear Classifiers

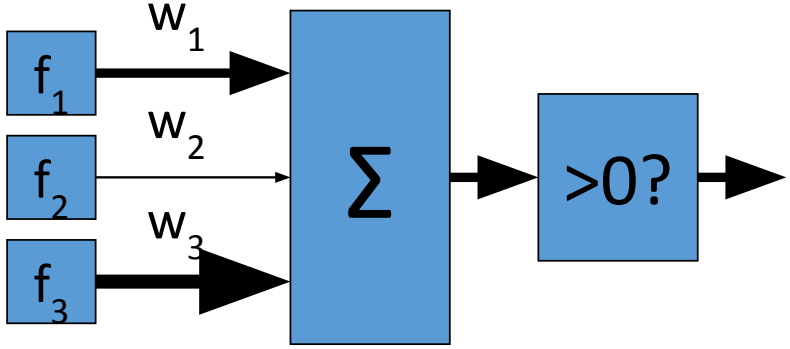
- Inputs are feature values
- Each feature has a weight
- Sum is the activation




$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

↳ feature from x

- If the activation is:
  - Positive, output +1
  - Negative, output -1



all example

$\sum f(x) w \rightarrow$ +ve

$\rightarrow$ -ve

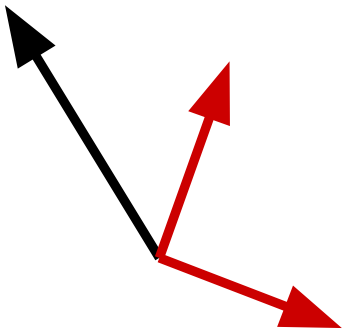
update weights

## Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples

```
# free       : 4
YOUR_NAME    :-1
MISSPELLED   : 1
FROM_FRIEND  :-3
...
```

$w$



$f(x_1)$

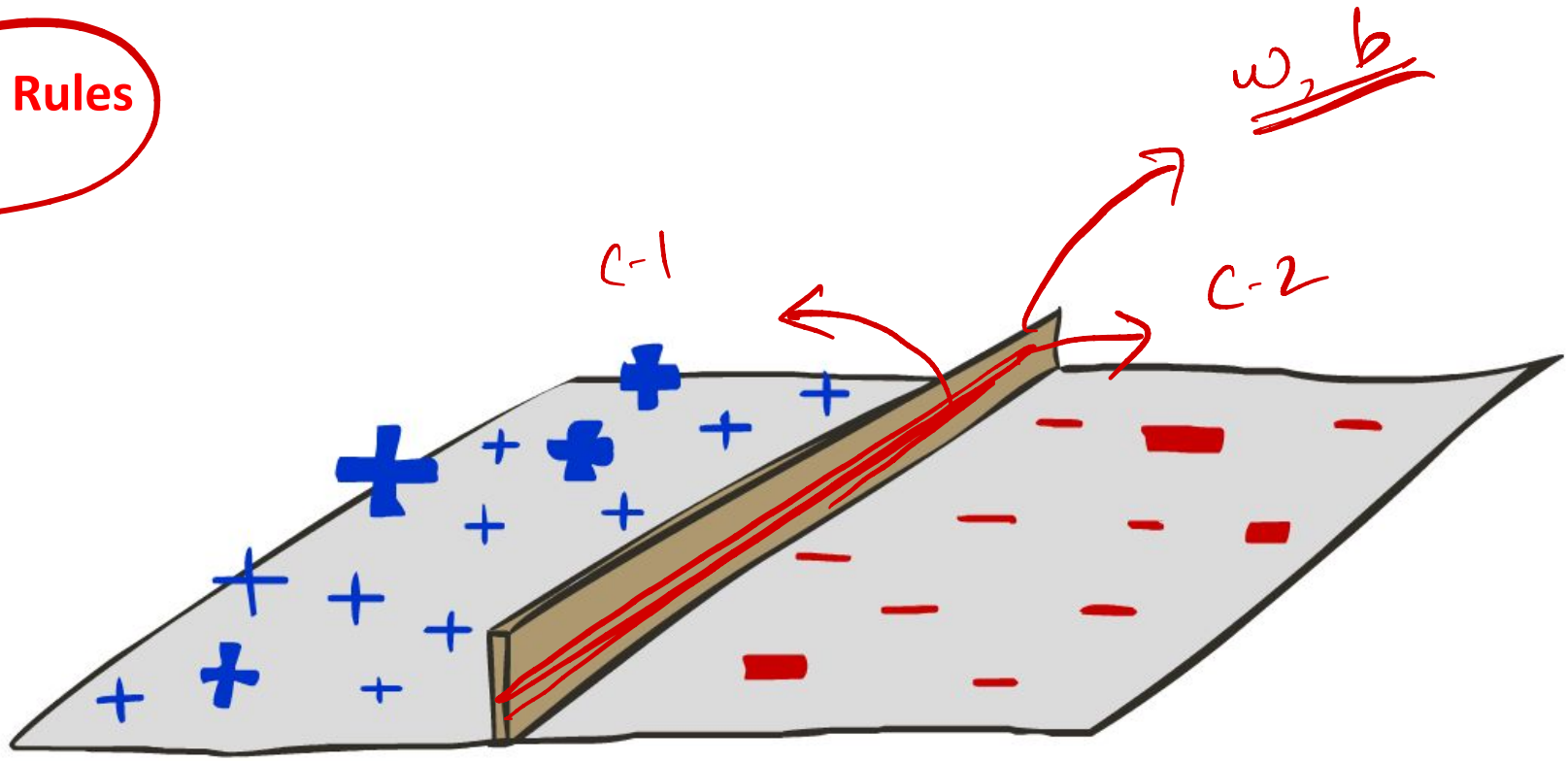
```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

$f(x_2)$

```
# free       : 0
YOUR_NAME    : 1
MISSPELLED   : 1
FROM_FRIEND  : 1
...
```

*Dot product* $w \cdot f$ *positive means the positive class*

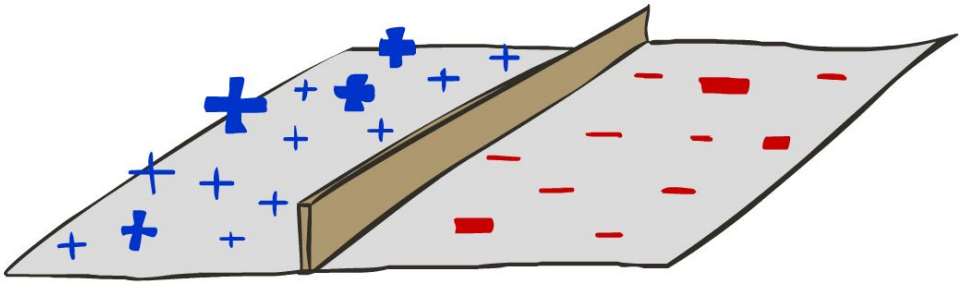
Decision Rules



true value
predicted value

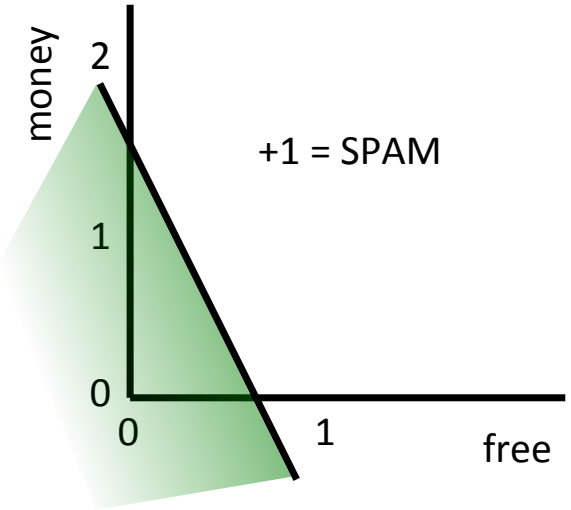
## Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - **Any** weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



$w$

```
BIAS  : -3
free  :  4
money :  2
...
```




$f \cdot w = 0$

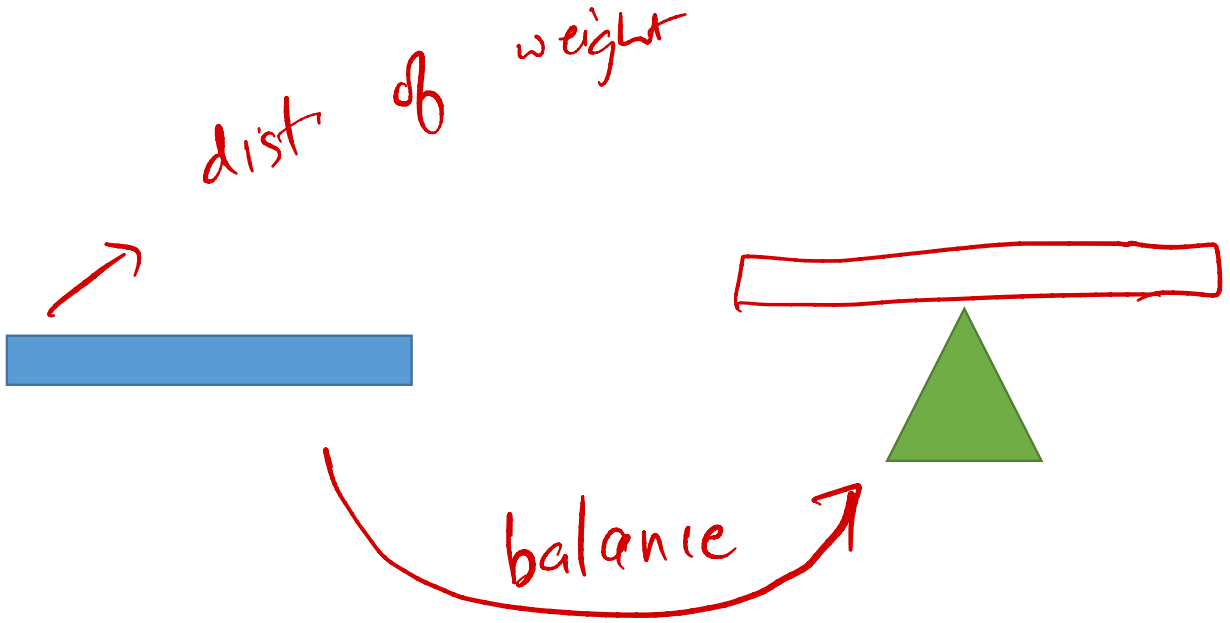
email
spam
not spam
feat
model (weight, bias
pred.

$$w = w + f \cdot y^*$$

## Weight Updates

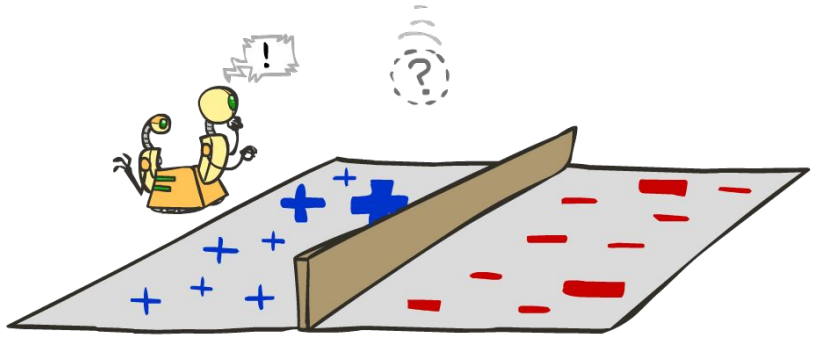
Example: to balance

Shishir Maheshwari

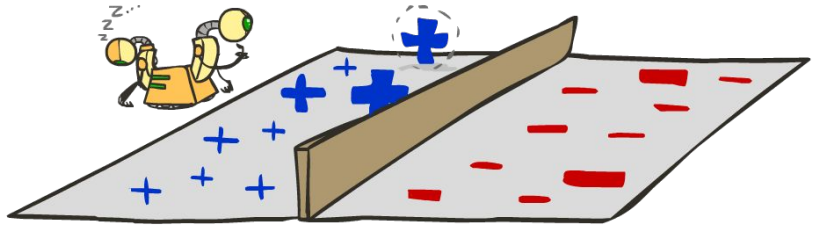


## Learning: Binary Perceptron

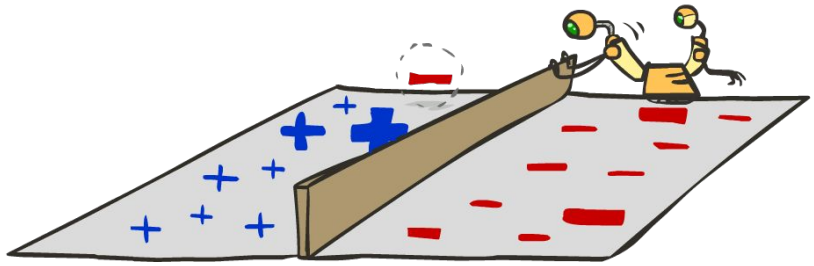
- Start with weights = 0
- For each training instance:
  - Classify with current weights




  - If correct (i.e., y=y*), no change!



  - If wrong: adjust the weight vector



# Mc Culloch pitts model (MP model)

→ 1949

→ AND, OR, XOR, NOT, ...

$$z = \sum x_i w_i$$

$$\hat{y} = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

$w \in [-1, 1]$

→ true value

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$$z = \sum_{i=1}^{n} x_i \cdot w_i \quad , \quad \hat{y} = \begin{cases} 1 , & z \geq \theta \\ 0 , & z < \theta \end{cases}$$

AND

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

find $w, \theta$ $\quad w \in [-1, 1]$

$\hat{y} = y \Rightarrow$ result , $w_1$ and $w_2$

→ initialise $w$

→ $z$, $\hat{y}$

→ find $\theta$ such that $y = \hat{y}$

$w_1 = 1 \quad w_2 = 1, \quad \theta = 2 \quad \hat{y} == y$

$0 \to 0$
$1 \to 0$
$1 \to 0$
$2 \to 1$

---

$w_1 = -1, \quad w_2 = 1, \quad \theta = ? \quad \hat{y} == y$

$0$
$-1$
$1$
$0$

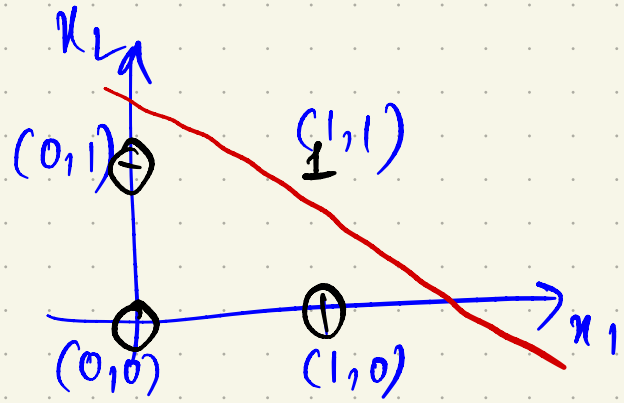
---

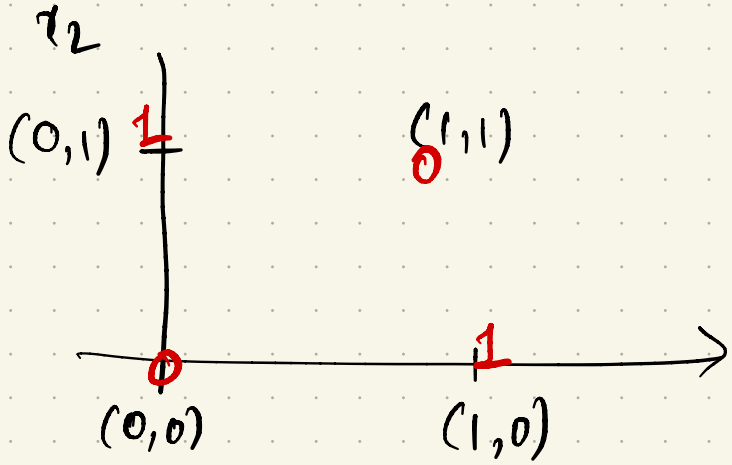
XOR

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$w_1 = ? \quad w_2 = ? \Rightarrow \theta = ? \Rightarrow$

$\hat{y} == y$




non - linear

XOR

more than one

---

issue with MP model ?

⇒ no auto update

⇒ $w_1$, $b$, or any other ⇒ auto update

## Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights

$\hat{y}$

$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

$\hat{y} \neq y^* \Rightarrow$ update

$\hat{y} == y^* \Rightarrow$ no update

  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

$$w = w + y^* \cdot f \Rightarrow \text{update rule}$$

$\hookrightarrow$ true label

| | ① | ② | ③ | ④ | ⑤ | ⑥ |
|---|---|---|---|---|---|---|
| $x =$ | -1 | -2 | -3 | 1 | 2 | 3 |
| $y^* =$ | -1 | -1 | -1 | 1 | 1 | 1 |

w= initialse , y = ? , update

x == f(x)

$y^* == y$ achieve ?

(1) w = -3

(2) y = for each

(3) update

$y_1 = 3 \Rightarrow 1 \neq -1$

$w = (-3) + (-1)(-1)$
$= -2$

$y_2 = 4 \Rightarrow 1 \neq -1$

$w = (-2) + (-1)(-1) = 0$

$y_3 = 0 \Rightarrow 1 \neq -1$

$w = (0) + (3) = 3$

---

$y_4 =$ ✓

$y_5 =$ ✓

$y_6 =$ ✓

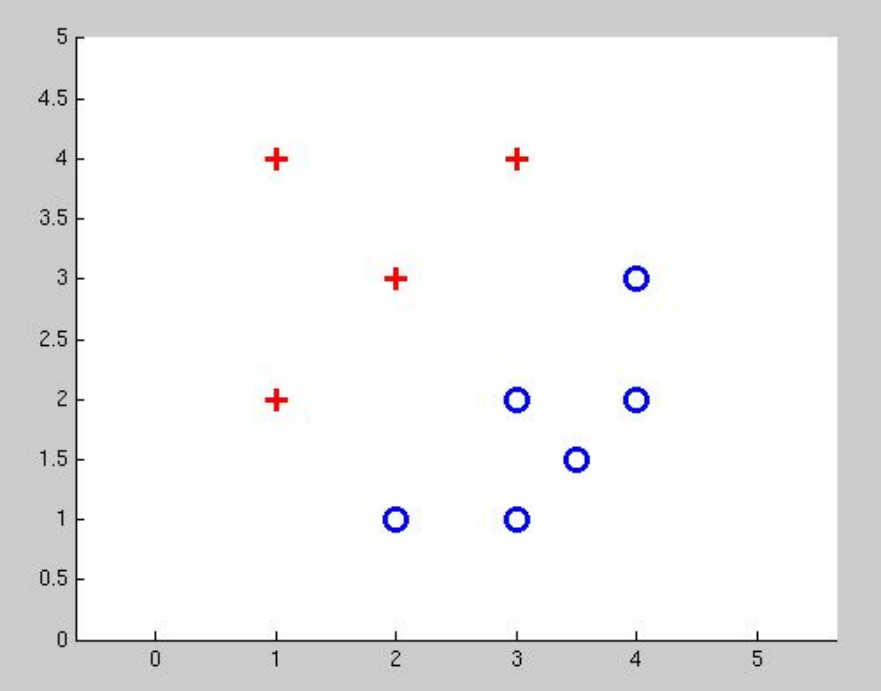
---

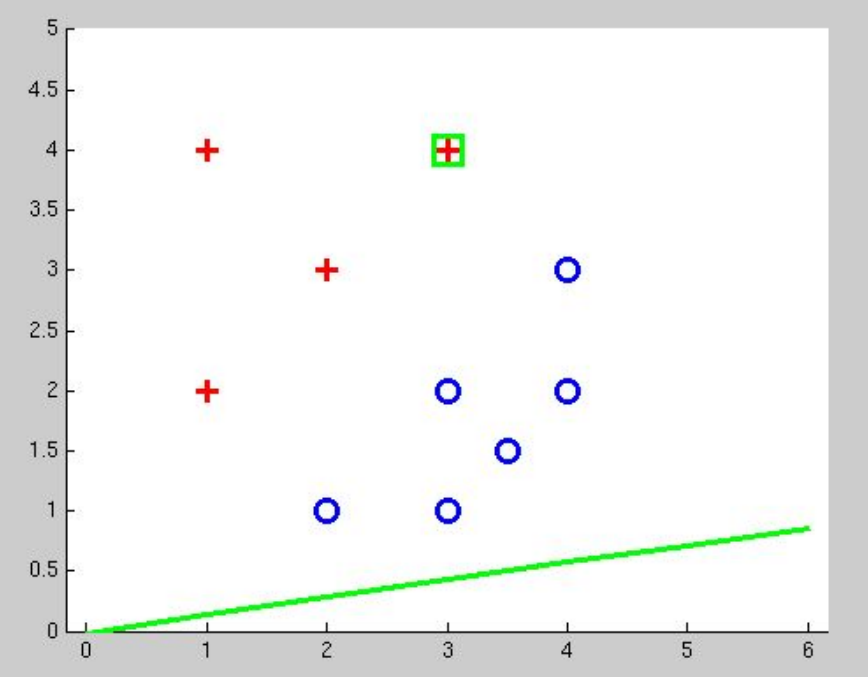
$y_1 =$ ✓

$y_2 =$ ✓

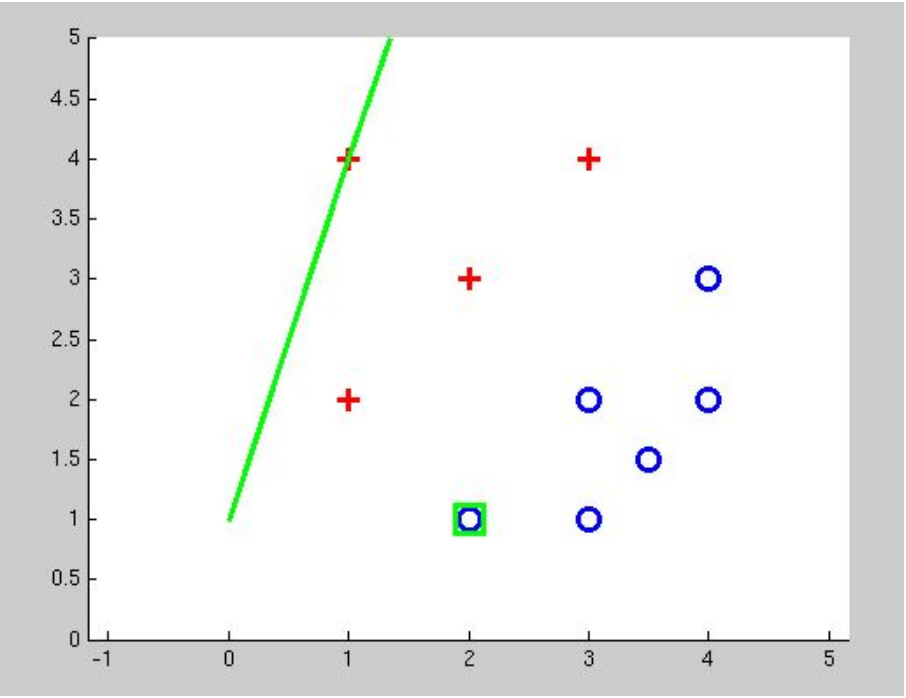
$y_3 =$ ✓

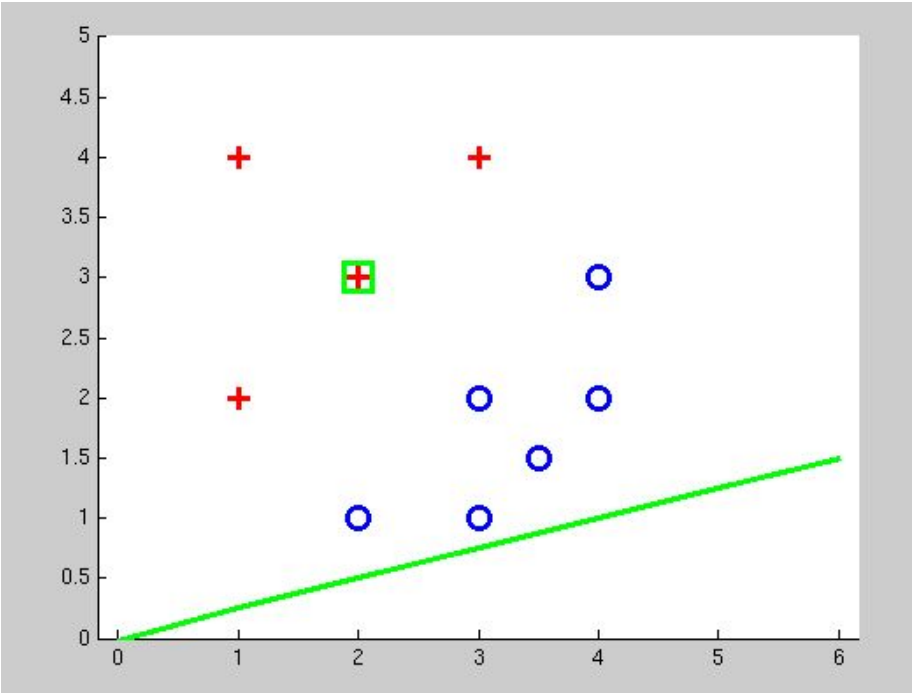
# Examples: Perceptron



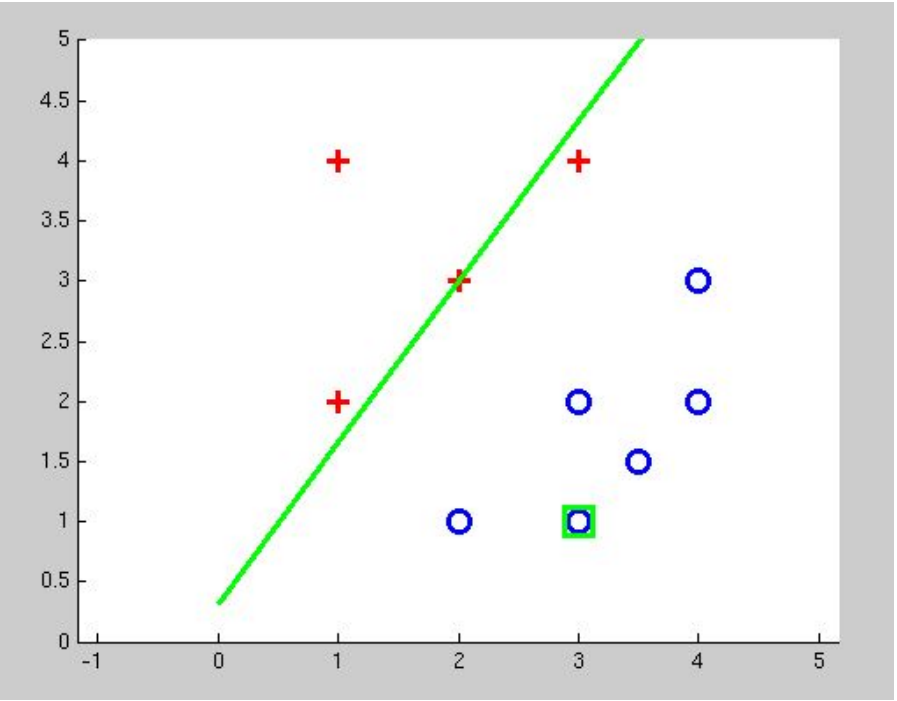



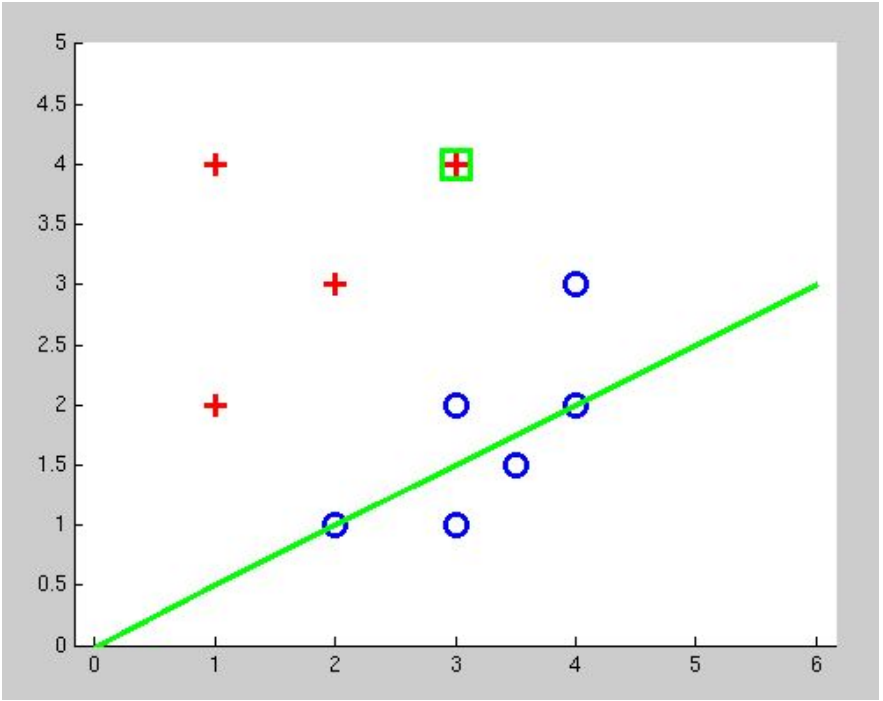
# Examples: Perceptron



# Examples: Perceptron

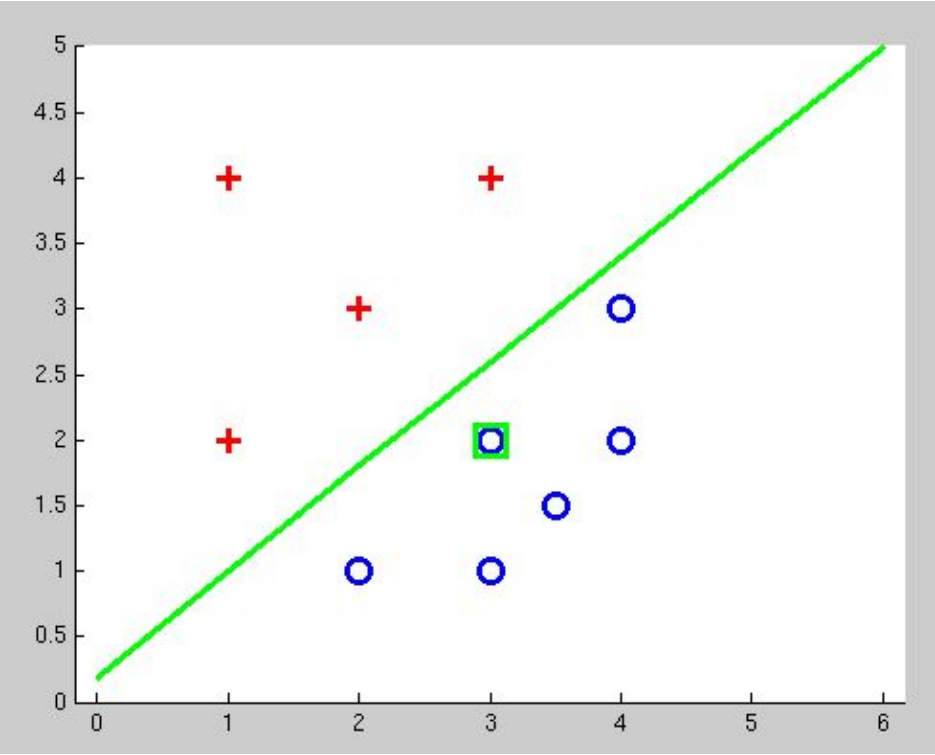


all input ⇒ one epoch

many epochs ⇒ find ans

## Examples: Perceptron



| | | | | | | |
|---|---|---|---|---|---|---|
| x = | 10 | 11 | 12 | 1 | 2 | 3 |
| $y^*$ = | -1 | -1 | -1 | 1 | 1 | 1 |

w = -1/-2/-3

b = 5/6/8

---

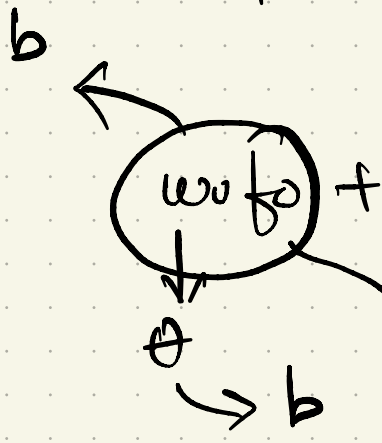
w = -3 , find w = ? for $y == y^*$

---

bias = ?

---

$\sum_{i=1}^{N} w_i f_i(x) > \theta \Rightarrow \sum_{i=1}^{N} w_i f_i + \theta = 0 \Rightarrow$



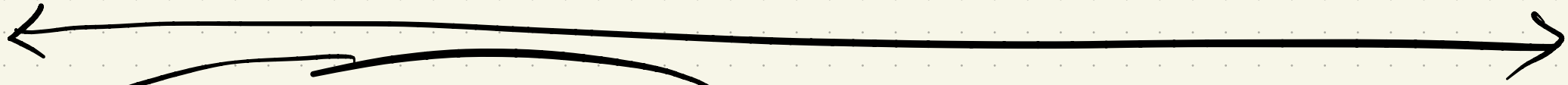

$+ w_1 f_1 + w_2 f_2 + \dots =$

$$\sum_{i=0}^{N} w_i f_i = y_{pred}$$

$$y_{pred} = b + \sum_{i=1}^{N} w_i * f_i(x)$$

or

$$\sum_{i=0}^{N} w_i * f_i(x)$$



$$w = w + y^* f$$

$$b = b + y^*$$

2- binary

## Multiclass Decision Rule

- If we have multiple classes:
  - A weight vector for each class:

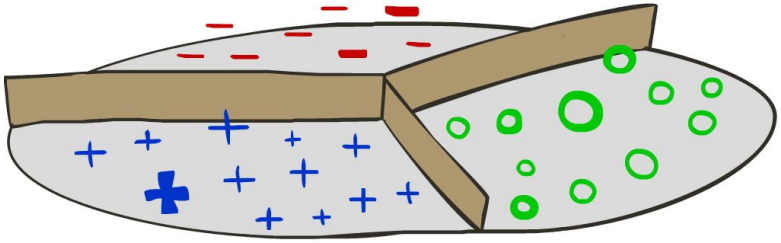
$$w_y$$

  - Score (activation) of a class y:

$$w_y \cdot f(x)$$

  - Prediction highest score wins

$$y = \arg\max_y \left[ w_y \cdot f(x) \right]$$



3- class $C_1$ $C_2$ $C_3$

$w_{C1}$ $w_{C2}$ $w_{C3}$

→ initial value of w

→ f(x). w

→ $y =$ argmax $[w \cdot f(x)]$

## Learning: Multiclass Perceptron

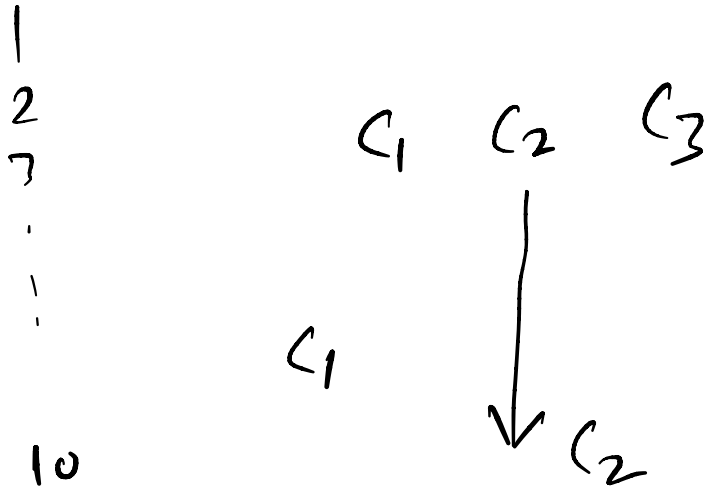
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

$$y = \arg\max_y \; w_y \cdot f(x)$$

- If correct, no change!

- If wrong: lower score of wrong answer, raise score of right answer

$$\left[\begin{array}{l} w_y = w_y - f(x) \\ \\ w_{y^*} = w_{y^*} + f(x) \end{array}\right]$$




C1 C2

≥ 0 C1

< 0 C2

true == pred ⇒ no update

≠ ⇒ update

$x_1 \rightarrow f(x_1) \Rightarrow$ $f(x_1) = \begin{bmatrix} 1 \\ win=1 \\ game=0 \\ vote=1 \\ the=1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

data $\Rightarrow$ feature

$x_1 \Rightarrow f(x_1)$

## Example: Multiclass Perceptron

(1) "win the vote"

(2) "win the election"

(3) "win the game"

$\Rightarrow f(x) \Rightarrow f(x) \Rightarrow \max \Rightarrow$ == no

$\neq$ update

$w_{SPORTS}$ initial value

| | | |
|---|---|---|
| BIAS | : | 1 |
| win | : | 0 |
| game | : | 0 |
| vote | : | 0 |
| the | : | 0 |
| ... | | |

$w_{POLITICS}$

| | | |
|---|---|---|
| BIAS | : | 0 |
| win | : | 0 |
| game | : | 0 |
| vote | : | 0 |
| the | : | 0 |
| ... | | |

$w_{TECH}$

| | | |
|---|---|---|
| BIAS | : | 0 |
| win | : | 0 |
| game | : | 0 |
| vote | : | 0 |
| the | : | 0 |
| ... | | |

$f(x_1) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ $w_{sp} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $w_{poli} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $w_{tech} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$x_1 \rightarrow$ sports
predicted

true label $\Rightarrow$ politics

$\Rightarrow$

$f(x_1) \cdot w_{sp}$ $\quad$ $f(x_1) \cdot w_{poli}$ $\quad$ $f(x_1) \cdot w_{tech}$

$1 \times$ $\quad$ $0 \times$ $\quad$ $0$

$\downarrow$ $\quad$ $\uparrow$ $\quad$ nc

$w_{sp} = w_{sp} -$

$= \begin{bmatrix} 0 \\ -1 \\ 0 \\ -1 \\ -1 \end{bmatrix}$

$w_{poli} = w_{poli} +$

$w_{poli} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

$f(x_2) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ $w_{sp}$ $w_{poli}$ $w_{tech}$

$-2$ $3$ $0$

$f(x_2) \rightarrow$ politics

true label $\rightarrow$ poli

---

$f(x_3) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$ $w_{sp}$ $w_{poli}$ $w_{tech}$

$-2$ $3$ $0$

$x$ $x$ NC

$f(x_3) \rightarrow$ polit

true $\rightarrow$ sports

$w_{sp} = w_{sp} +$

$w_{poli} = w_{poli} -$

$w_{sp} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}$ $w_{poli} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}$ $w_{tech} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

→ initi w (given | random)

→ find score

→ from score find pred. class

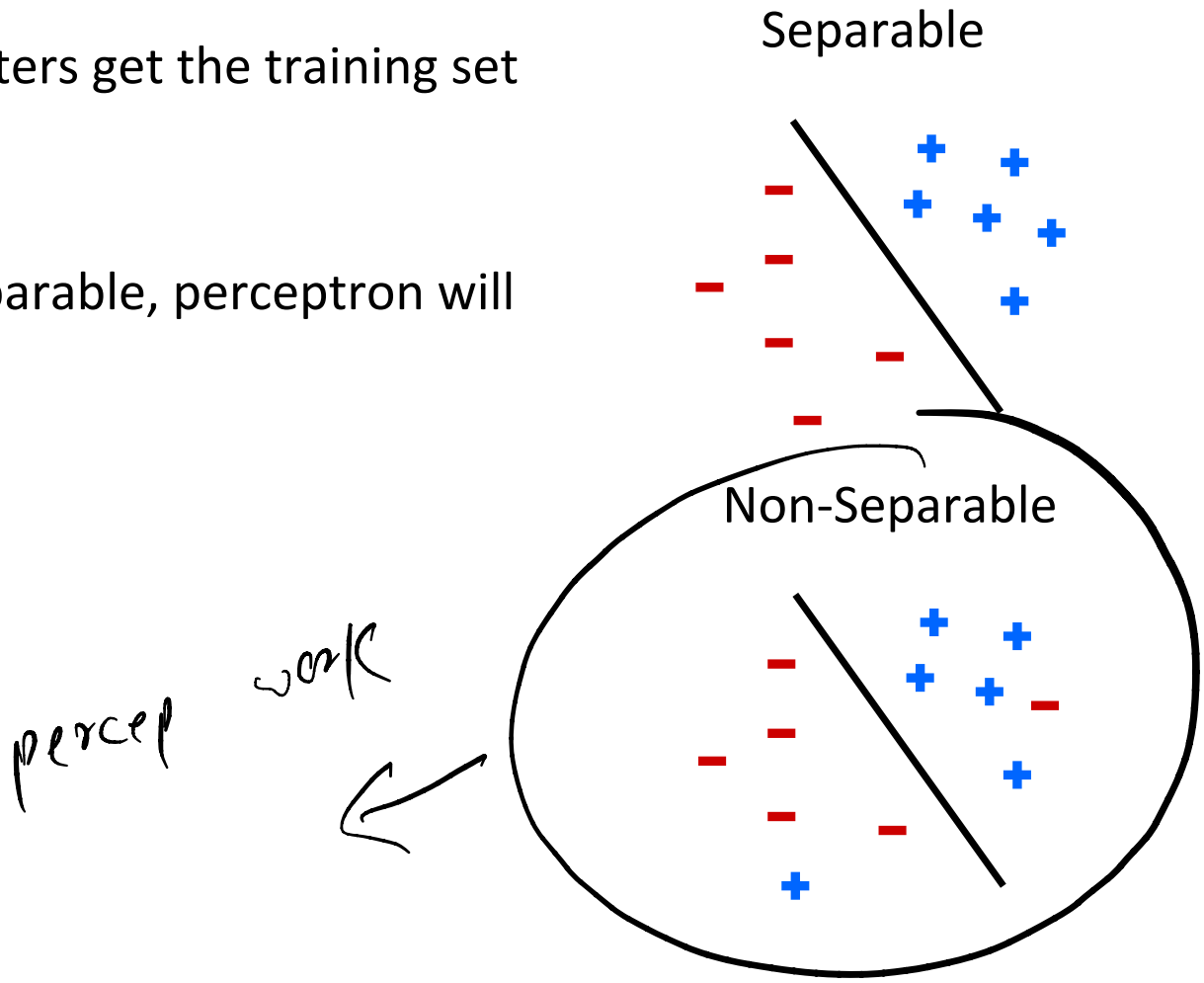
⇒ pred == true , no update

⇒ ≠ update

## Properties of Perceptrons

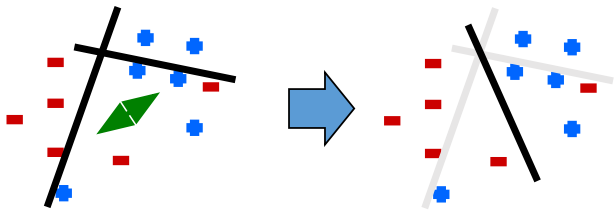
- Separability: true if some parameters get the training set perfectly correct

- Convergence: if the training is separable, perceptron will eventually converge (binary case)

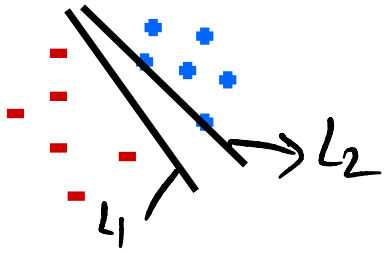


## Problems with the Perceptron

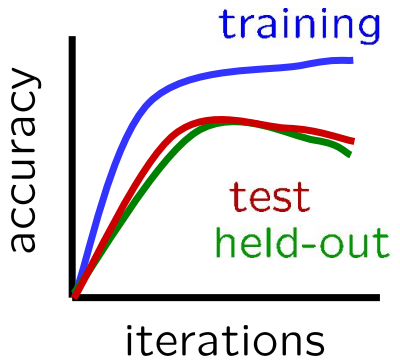
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



- Mediocre generalization: finds a "barely" separating solution




- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



## Improving the Perceptron

mapp i/p ⟶ o/p [ ]