

# Learning From Data

## Lecture 2

### The Perceptron

The Learning Setup  
A Simple Learning Algorithm: PLA  
Other Views of Learning  
Is Learning Feasible: A Puzzle

**M. Magdon-Ismail**  
CSCI 4100/6100

---

RECAP: **The Key Players**

- Salary, debt, years in residence, ...
- Approve credit or not
- True relationship between  $\mathbf{x}$  and  $y$
- Data on customers

*input*  $\mathbf{x} \in \mathbb{R}^d = \mathcal{X}$ .

*output*  $y \in \{-1, +1\} = \mathcal{Y}$ .

*target function*  $f : \mathcal{X} \mapsto \mathcal{Y}$ .

(The target  $f$  is *unknown*.)

*data set*  $\mathcal{D} = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ .

( $y_n = f(\mathbf{x}_n)$ .)

$\mathcal{X}$   $\mathcal{Y}$  and  $\mathcal{D}$  are *given* by the learning problem;  
The target  $f$  is fixed but unknown.

We learn the function  $f$  from the data  $\mathcal{D}$ .

# A Simple Learning Model

- Input vector  $\mathbf{x} = [x_1, \dots, x_d]^T$ .
- Give importance weights to the different inputs and compute a “Credit Score”

$$\text{“Credit Score”} = \sum_{i=1}^d w_i x_i.$$

- Approve credit if the “Credit Score” is acceptable.

$$\text{Approve credit if } \sum_{i=1}^d w_i x_i > \text{threshold}, \quad (\text{“Credit Score” is good})$$

$$\text{Deny credit if } \sum_{i=1}^d w_i x_i < \text{threshold}. \quad (\text{“Credit Score” is bad})$$

- How to choose the importance weights  $w_i$

input  $x_i$  is important  $\implies$  large weight  $|w_i|$   
input  $x_i$  beneficial for credit  $\implies$  positive weight  $w_i > 0$   
input  $x_i$  detrimental for credit  $\implies$  negative weight  $w_i < 0$

# A Simple Learning Model

Approve credit if  $\sum_{i=1}^d w_i x_i > \text{threshold},$

Deny credit if  $\sum_{i=1}^d w_i x_i < \text{threshold}.$

can be written formally as

$$h(\mathbf{x}) = \text{sign} \left( \left( \sum_{i=1}^d w_i x_i \right) + w_0 \right)$$

The “bias weight”  $w_0$  corresponds to the threshold. (How?)

# The Perceptron Hypothesis Set

We have defined a *Hypothesis set*  $\mathcal{H}$

$$\mathcal{H} = \{h(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})\}$$

← uncountably infinite  $\mathcal{H}$

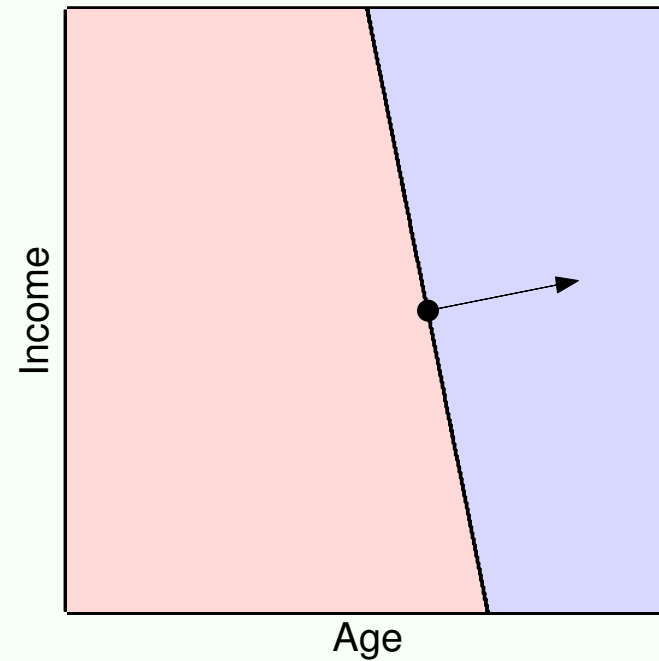
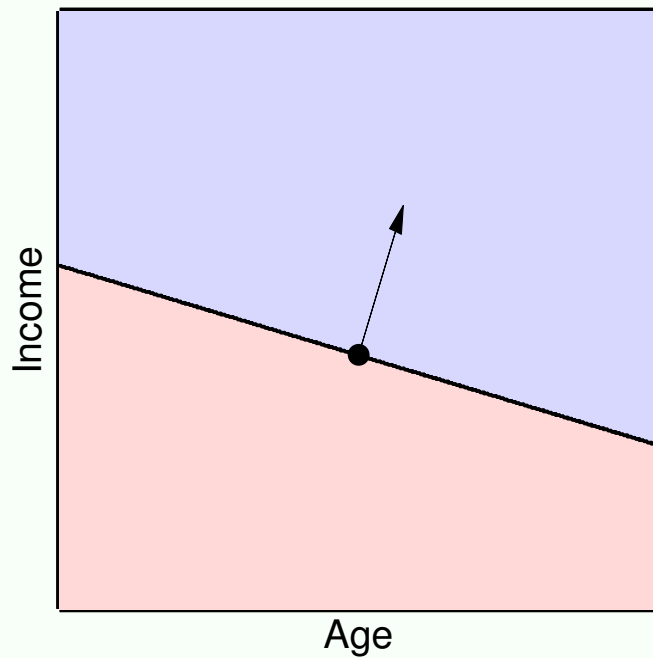
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \in \mathbb{R}^{d+1}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \in \{1\} \times \mathbb{R}^d.$$

This hypothesis set is called the *perceptron* or *linear separator*

# Geometry of The Perceptron

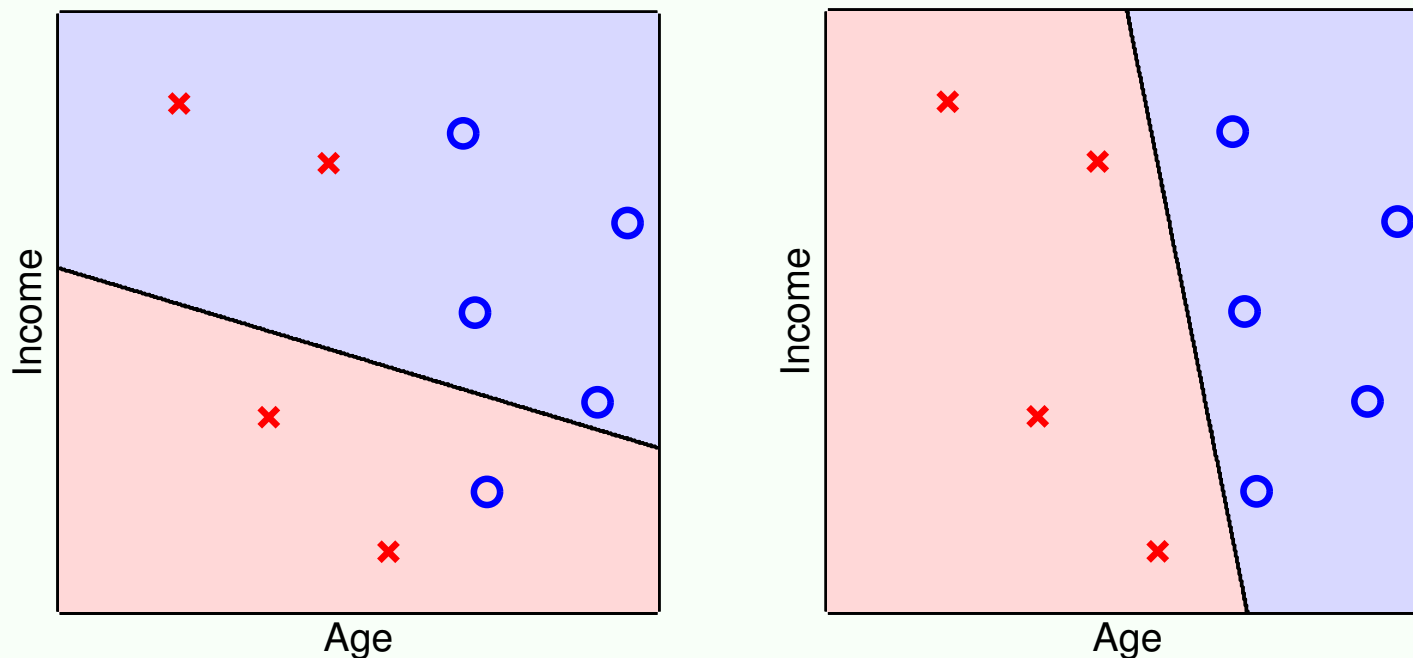
$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$$

(Problem 1.2 in LFD)



Which one should we pick?

# Use the Data to Pick a Line



A perceptron fits the data by using a line to separate the  $+1$  from  $-1$  data.

**Fitting the data:** How to find a hyperplane that *separates* the data?  
(“It’s obvious - just look at the data and draw the line,” is not a valid solution.)

# How to *Learn* a Final Hypothesis $g$ from $\mathcal{H}$

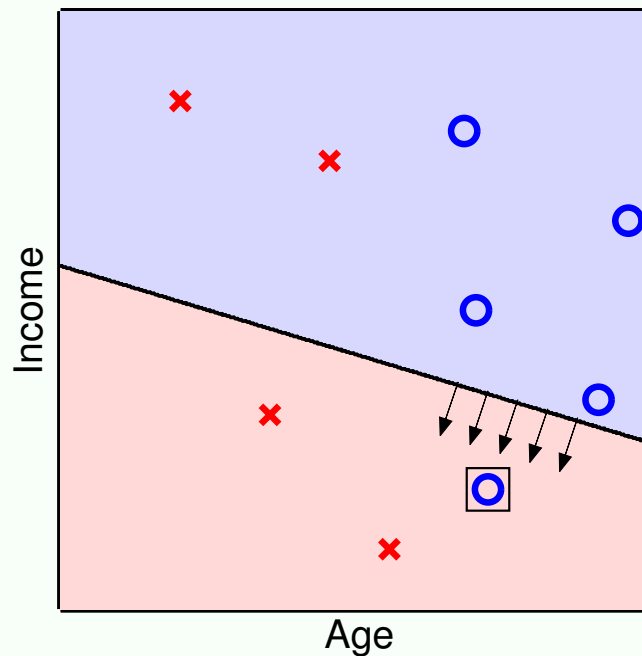
We want to select  $g \in \mathcal{H}$  so that  $g \approx f$ .

We certainly want  $g \approx f$  on the data set  $\mathcal{D}$ . Ideally,

$$g(\mathbf{x}_n) = y_n.$$

How do we find such a  $g$  in the *infinite* hypothesis set  $\mathcal{H}$ , if it exists?

**Idea!** Start with some weight vector and try to improve it.





# The Perceptron Learning Algorithm (PLA)

A simple iterative method.

1:  $\mathbf{w}(1) = \mathbf{0}$

如何演示下面的例子

2: **for** iteration  $t = 1, 2, 3, \dots$

3: the weight vector is  $\mathbf{w}(t)$ .

4: From  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$  pick any misclassified example.

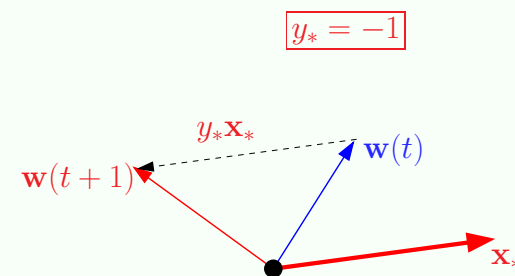
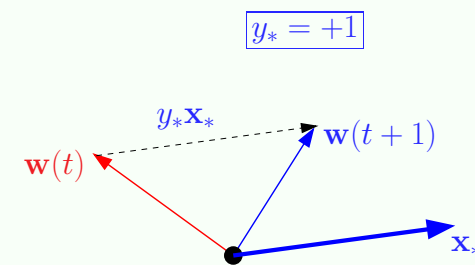
5: Call the misclassified example  $(\mathbf{x}_*, y_*)$ ,

$$\text{sign}(\mathbf{w}(t) \cdot \mathbf{x}_*) \neq y_*.$$

6: Update the weight:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y_* \mathbf{x}_*.$$

7:  $t \leftarrow t + 1$

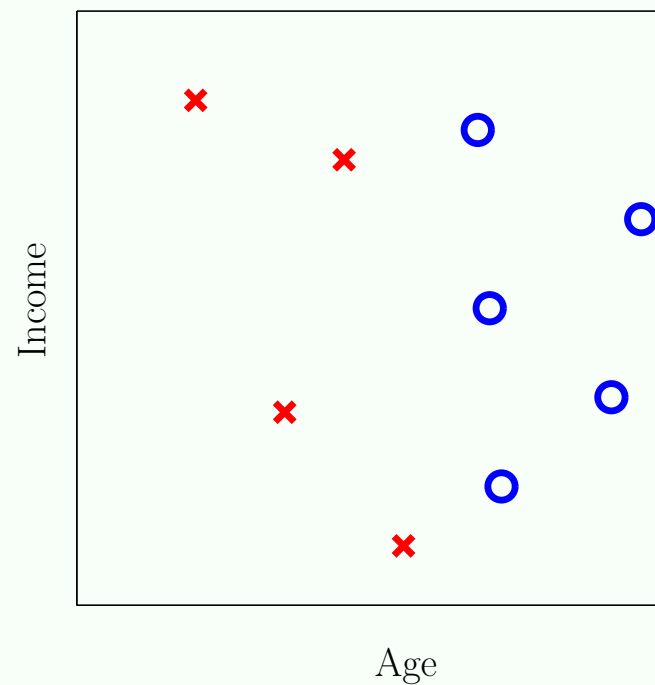


PLA implements our idea: start at some weights and try to improve.

“incremental learning” on a single example at a time

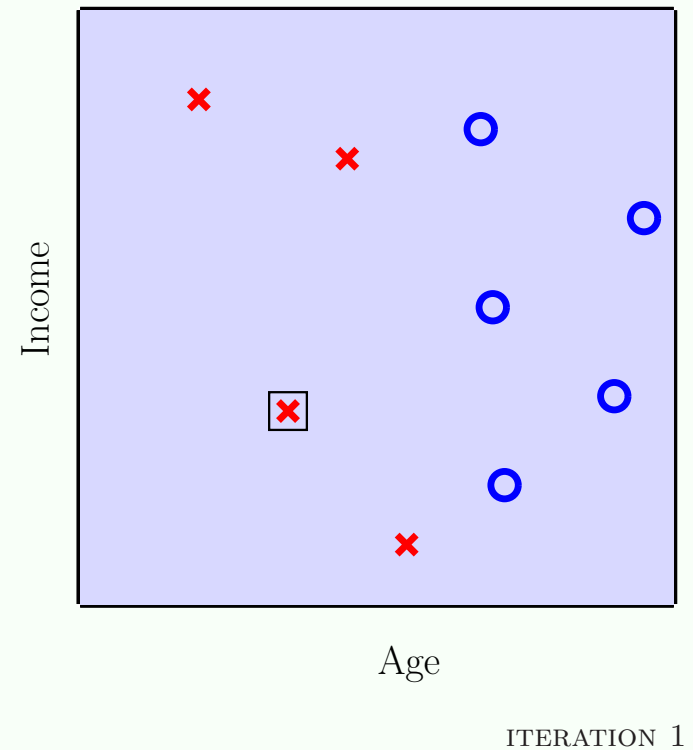
# Does PLA Work?

**Theorem.** If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



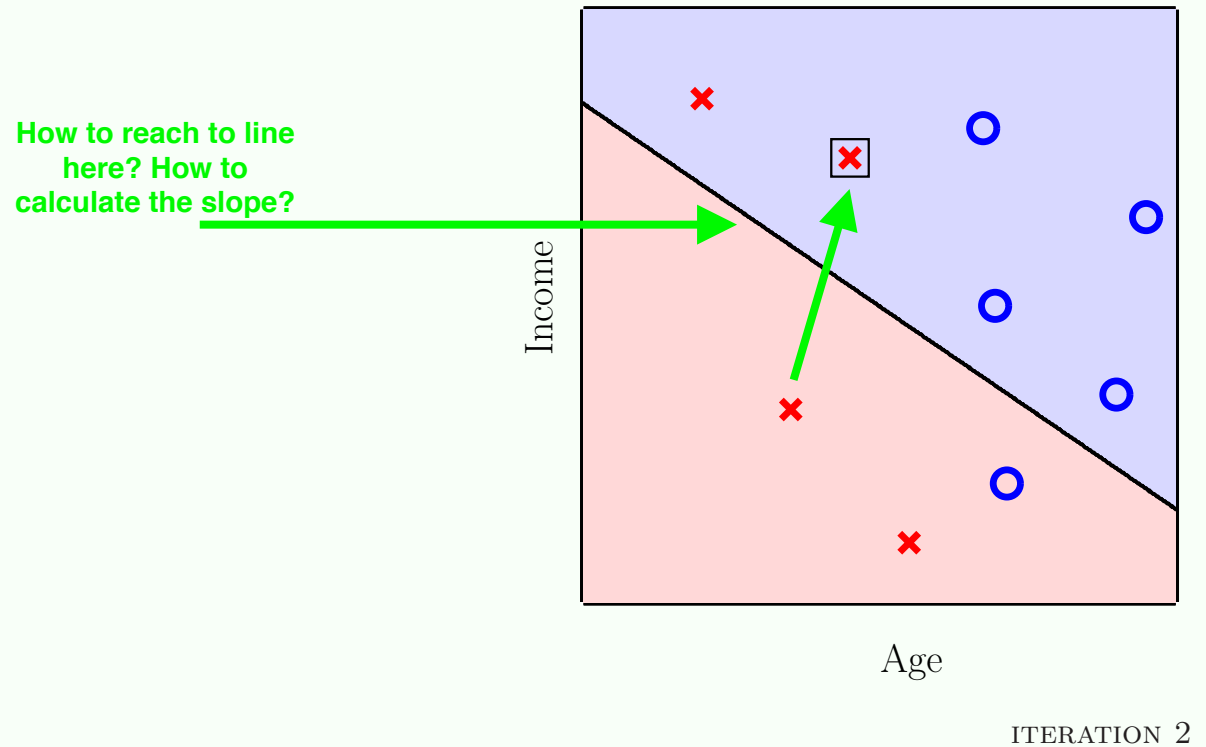
# Does PLA Work?

**Theorem.** If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



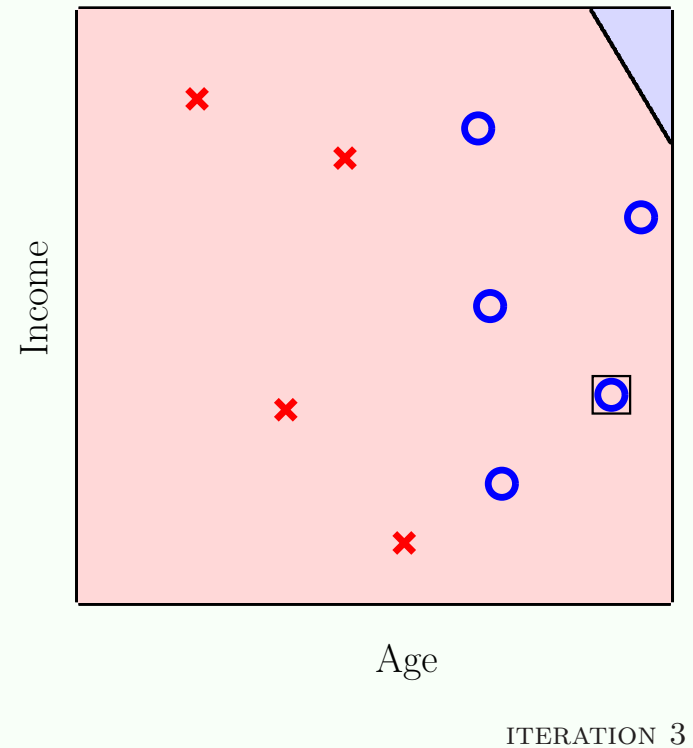
# Does PLA Work?

**Theorem.** If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



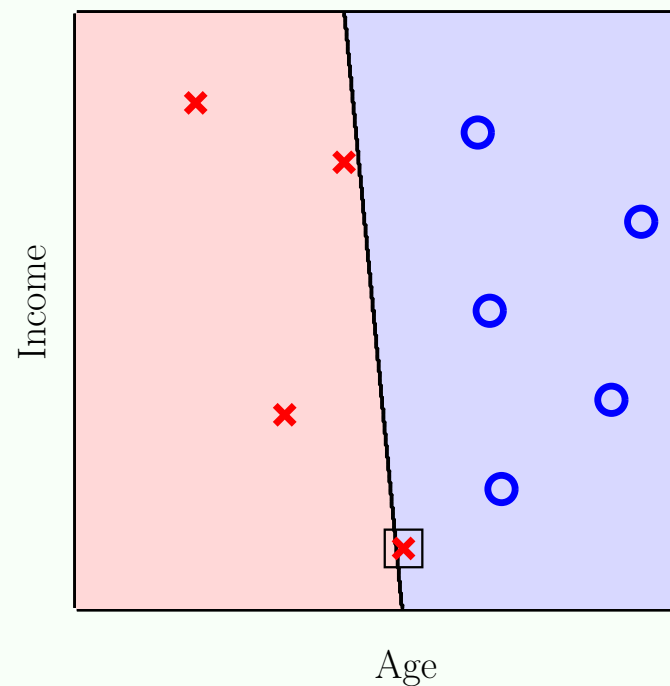
# Does PLA Work?

**Theorem.** If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



# Does PLA Work?

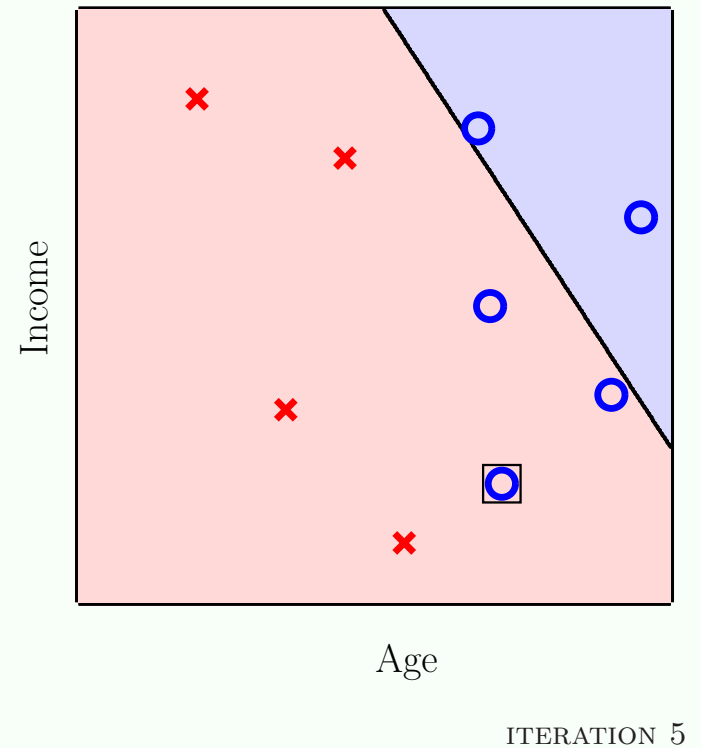
**Theorem.** If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



ITERATION 4

# Does PLA Work?

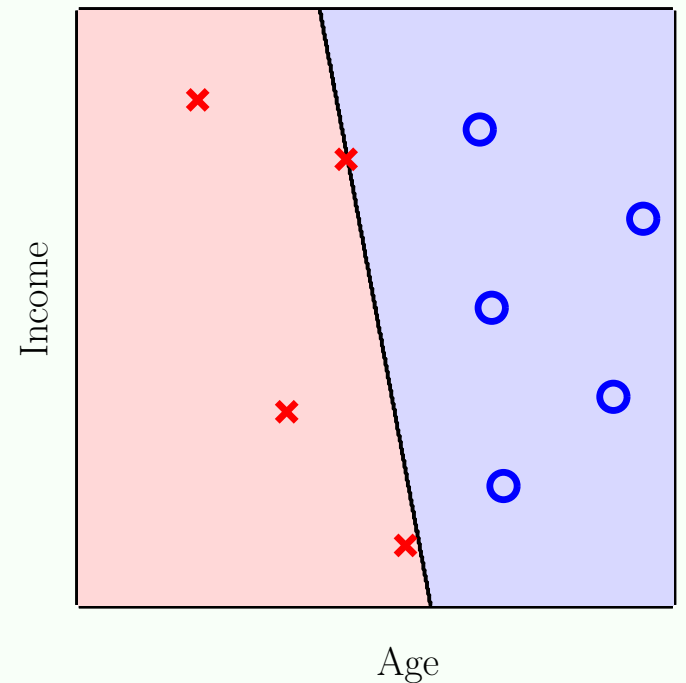
**Theorem.** If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



# Does PLA Work?

**Theorem.** If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.

After how long?



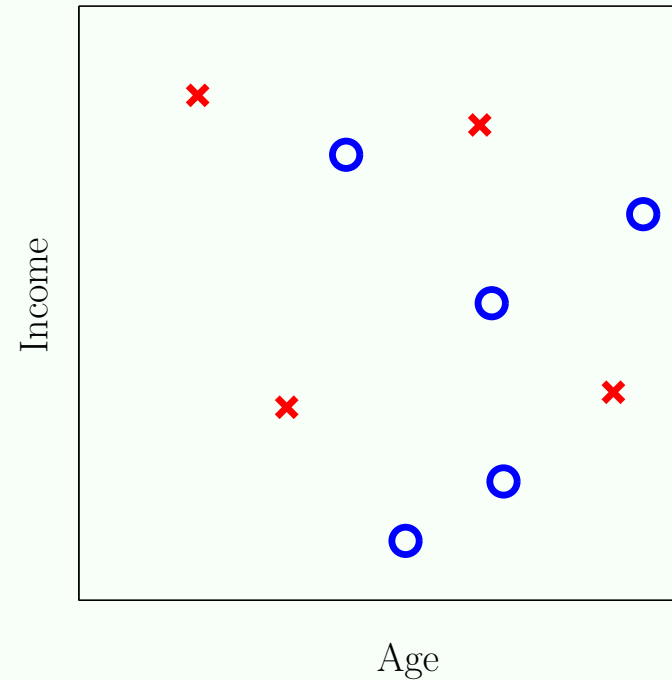


# Does PLA Work?

**Theorem.** If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.

After how long?

What if the data cannot be fit by a perceptron?



# We can Fit the Data

- We can find an  $h$  that works from infinitely many (for the perceptron).  
(So computationally, things seem good.)
- Ultimately, remember that we want to *predict*.  
We don't care about the data, we care about “*outside the data*”.

**Can a limited data set reveal enough information to pin down an entire target function, so that we can predict outside the data?**