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

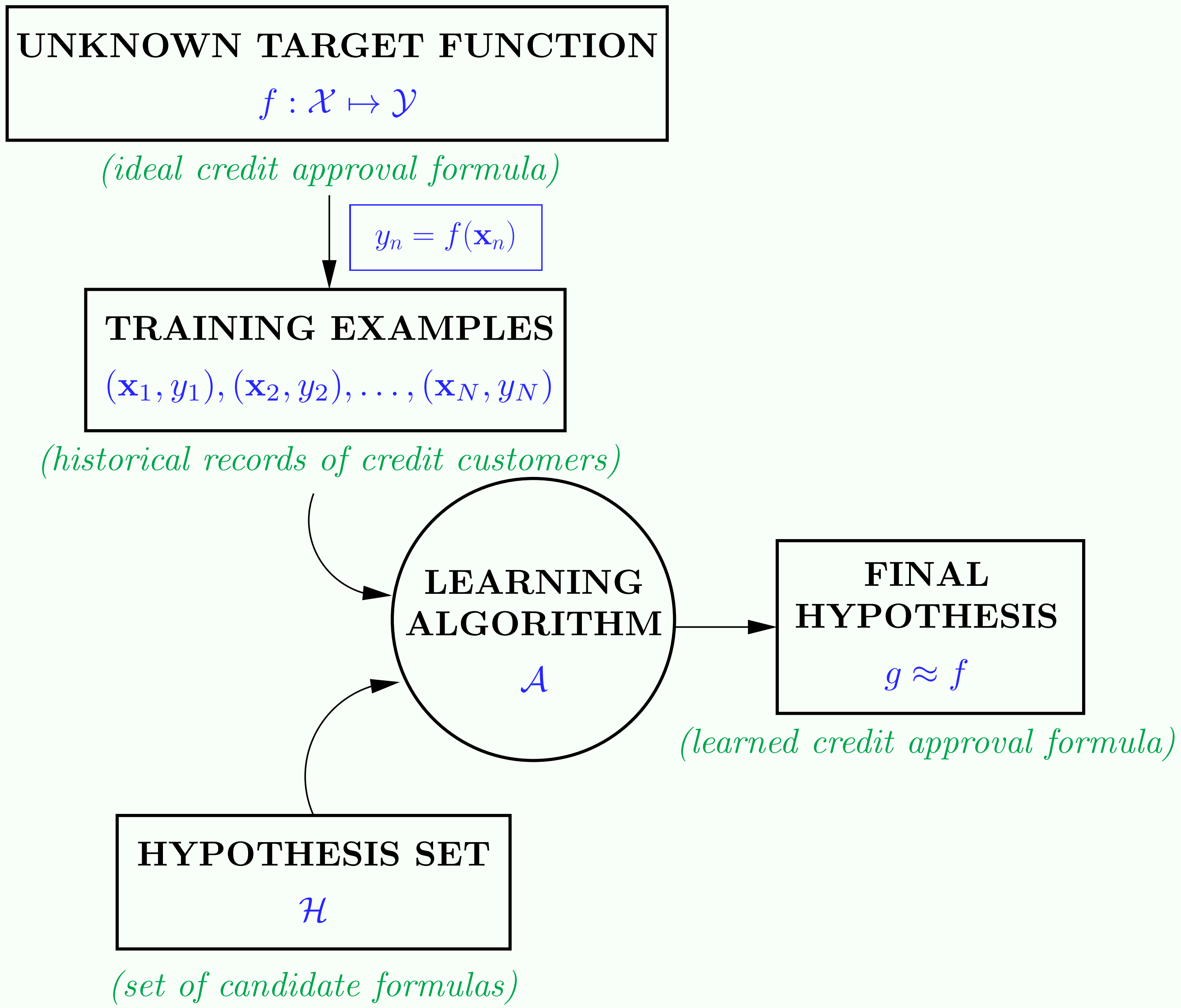
• Salary, debt, years in residence, ...	*input* $\mathbf{x} \in \mathbb{R}^d = \mathcal{X}$.
• Approve credit or not	*output* $y \in \{-1, +1\} = \mathcal{Y}$.
• True relationship between $\mathbf{x}$ and y	*target function* $f : \mathcal{X} \mapsto \mathcal{Y}$. **(The target f is *unknown*.)**
• Data on customers	*data set* $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. ($y_n = f(\mathbf{x}_n)$.)

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

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

RECAP:

Summary of the Learning Setup



A Simple Learning Model

- Input vector $\mathbf{x} = [x_1, \ldots, x_d]^{\mathrm{T}}$.
- 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.

$$\begin{aligned} &\text{Approve credit if} \quad \sum_{i=1}^{d} w_i x_i > \text{threshold}, \qquad &&\text{(“Credit Score” is good)} \\ &\text{Deny credit if} \quad \sum_{i=1}^{d} w_i x_i < \text{threshold}. \qquad &&\text{(“Credit Score” is bad)} \end{aligned}$$

- How to choose the importance weights w_i

input x_i is important	$\Longrightarrow$	large weight $\lvert w_i \rvert$
input x_i beneficial for credit	$\Longrightarrow$	positive weight $w_i > 0$
input x_i detrimental for credit	$\Longrightarrow$	negative weight $w_i < 0$

A Simple Learning Model

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

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

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 *Hyopthesis set* $\mathcal{H}$

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

$\longleftarrow$ uncountably infinite $\mathcal{H}$

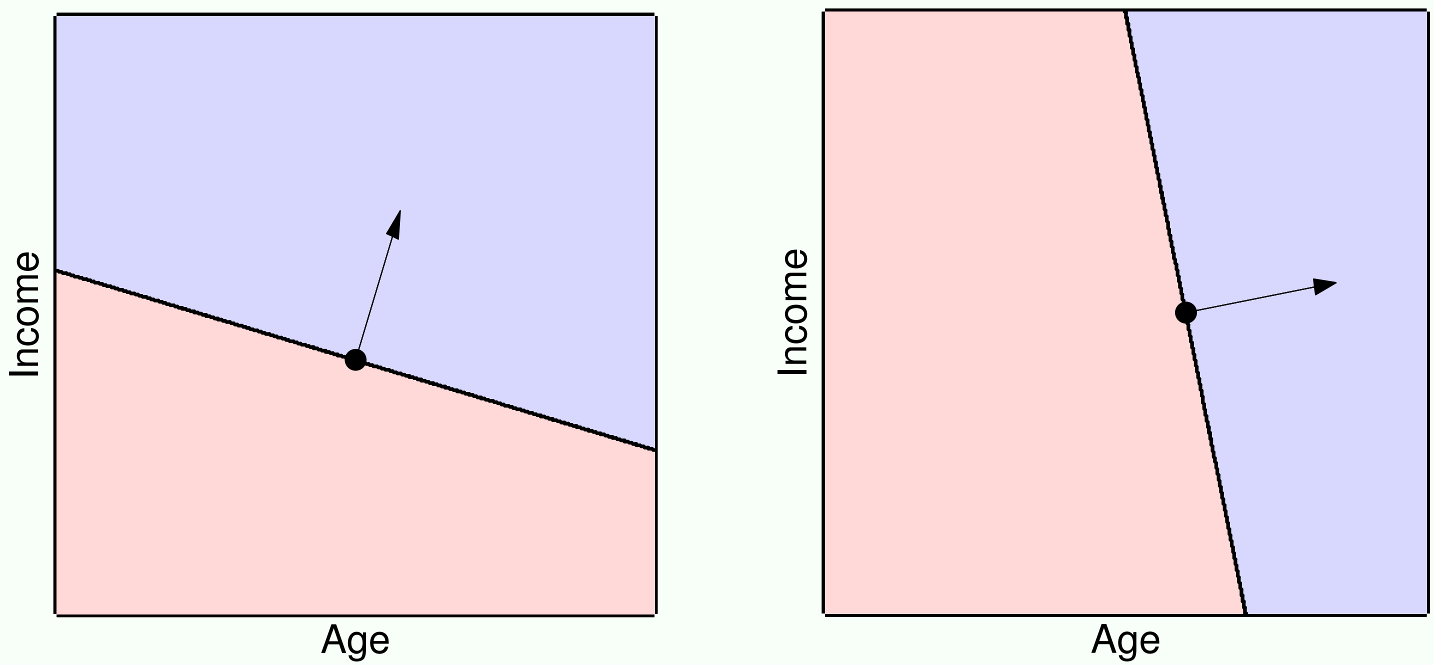
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \in \mathbb{R}^{d+1}, \qquad \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \in \{1\} \times \mathbb{R}^{d}.$$

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

Geometry of The Perceptron

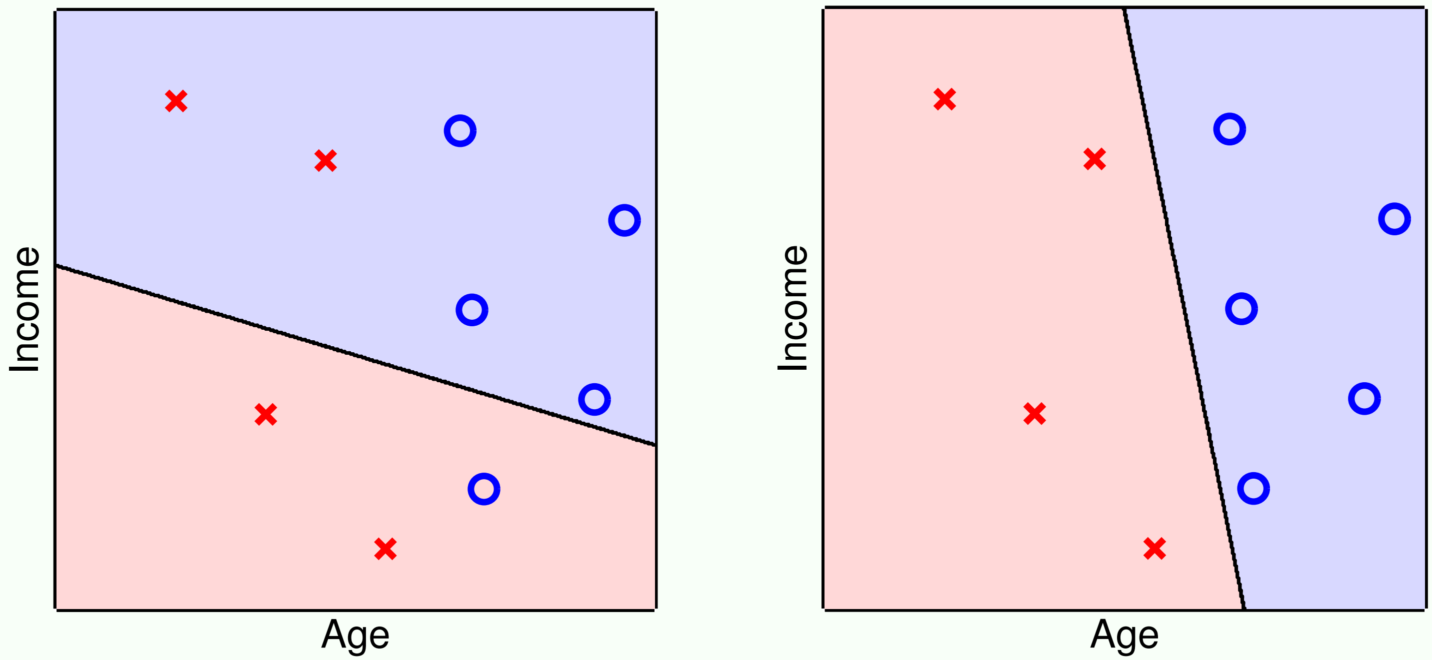
$$h(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

(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 $\mathcal{H}$

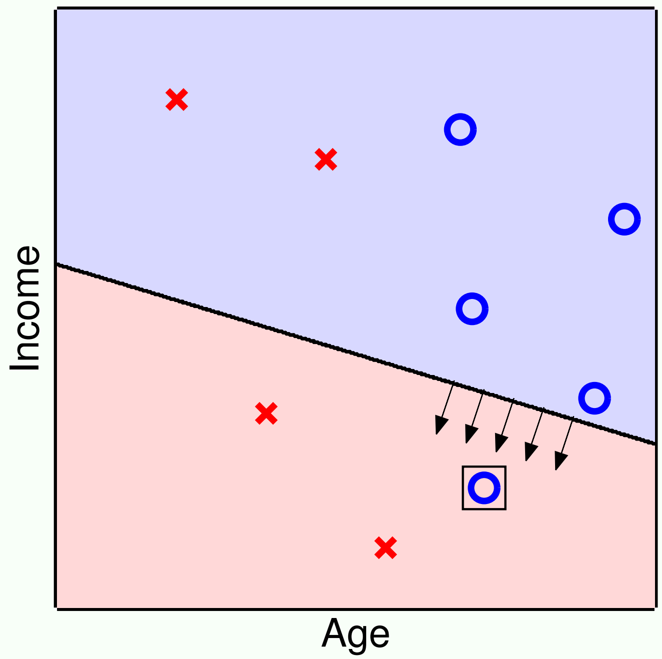
We want to select $g \in \mathcal{H}$ so that $g \approx f$.

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.

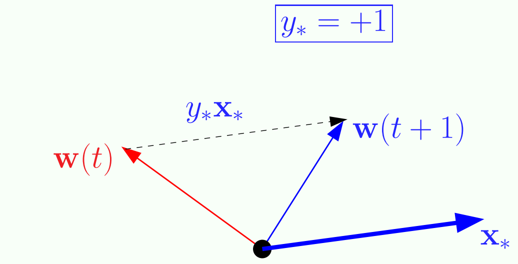
1: $\mathbf{w}(1) = \mathbf{0}$
2: **for** iteration $t = 1, 2, 3, \ldots$
3: the weight vector is $\mathbf{w}(t)$.
4: From $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ pick any misclassified example.
5: Call the misclassified example $(\mathbf{x}_*, y_*)$,

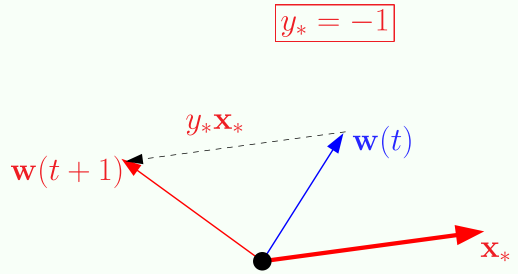
$$\text{sign}\left(\mathbf{w}(t) \bullet \mathbf{x}_*\right) \neq y_*.$$

6: Update the weight:

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

7: $t \leftarrow t + 1$



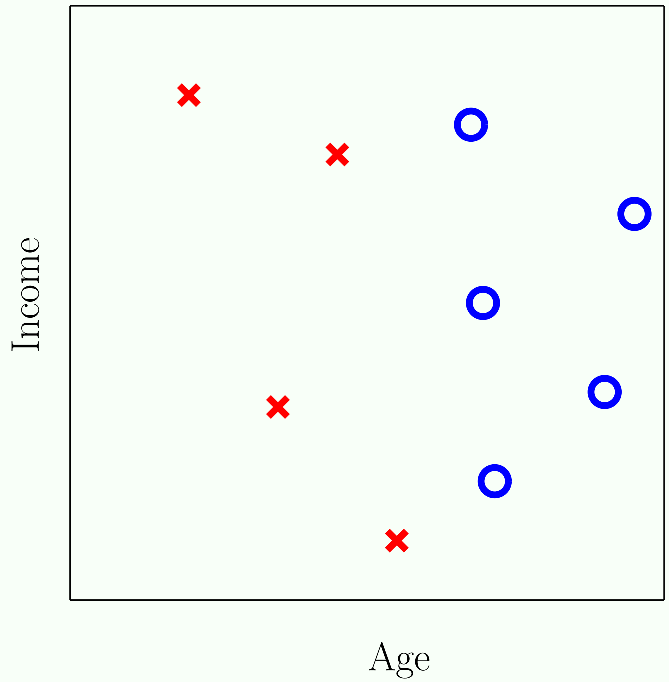


PLA implements our idea: start at some weights and try to improve.

"incremental learning" on a single example at a time

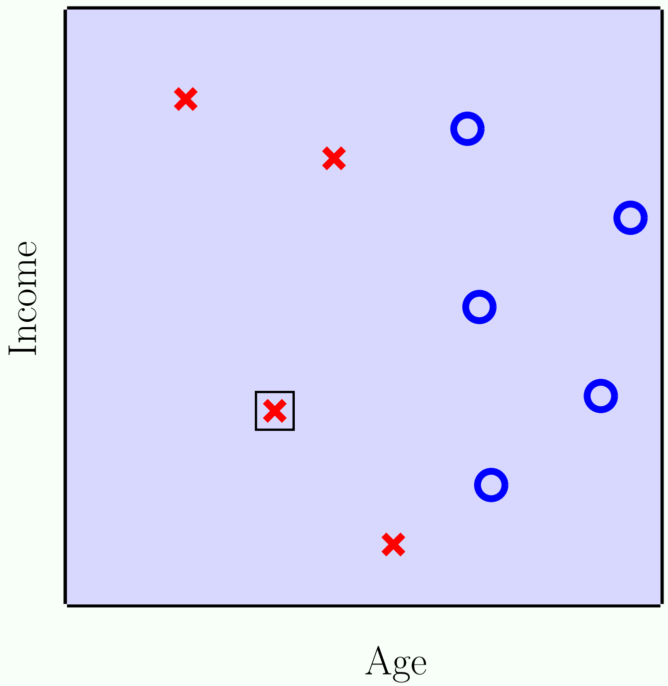
Does PLA Work?

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?

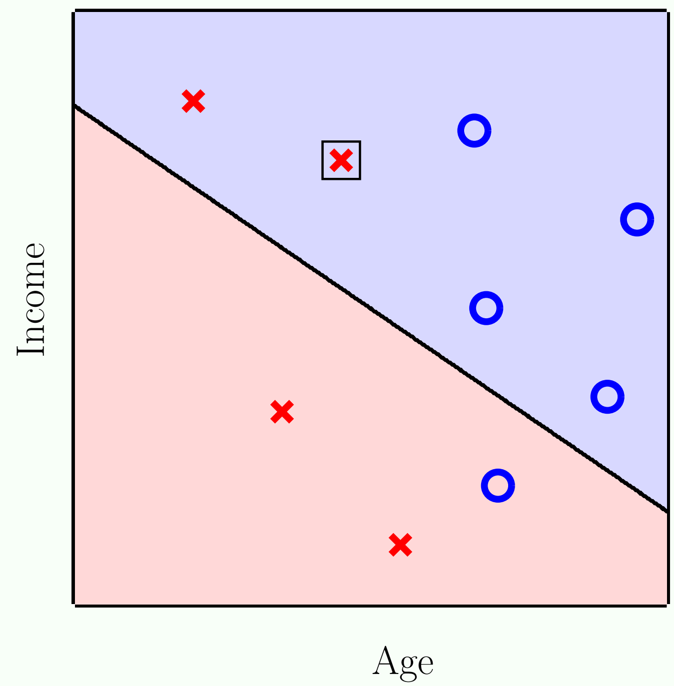
Theorem. If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



ITERATION 1

Does PLA Work?

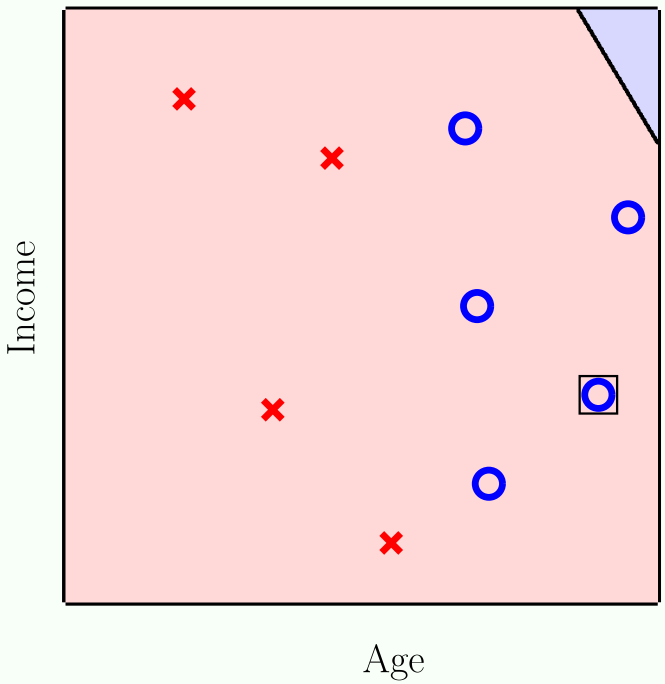
Theorem. If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



ITERATION 2

Does PLA Work?

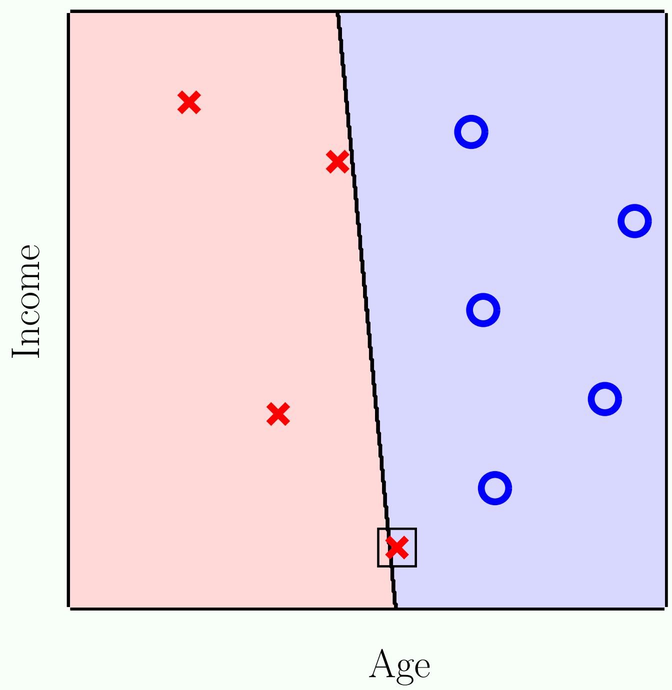
Theorem. If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



ITERATION 3

Does PLA Work?

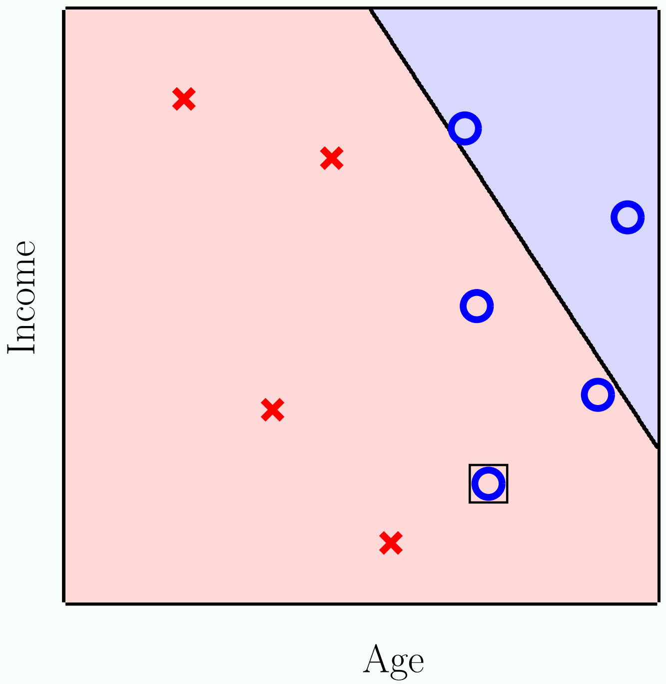
Theorem. If the data can be fit by a linear separator, then after some *finite* number of steps, PLA will find one.



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.

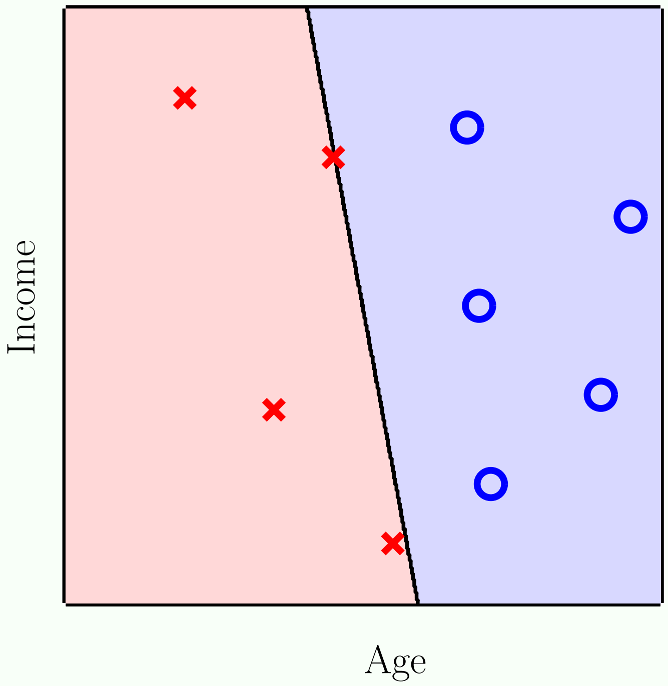


ITERATION 5

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?



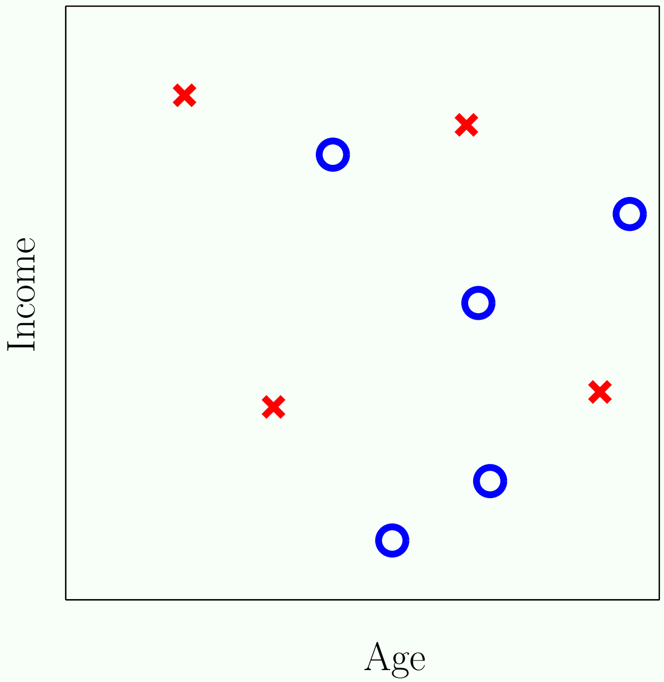
ITERATION 6

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?

Other Views of Learning

- Design: learning is from data, design is from specs and a model.
- Statistics, Function Approximation.
- Data Mining: find patterns in massive data (typically unsupervised).
- Three Learning Paradigms
  - Supervised: the data is $(\mathbf{x}_n, f(\mathbf{x}_n))$ – you are told the answer.
  - Reinforcement: you get feedback on potential answers you try:

    $$\mathbf{x} \rightarrow \text{ try something } \rightarrow \text{ get feedback.}$$

  - Unsupervised: only given $\mathbf{x}_n$, learn to "organize" the data.