Machine Learning
(機器學習)

Lecture 1: Basics of Machine Learning

Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Roadmap

1. **When** Can Machines Learn?

Lecture 1: Basics of Machine Learning

- What is Machine Learning
- Applications of Machine Learning
- Components of Machine Learning
- Machine Learning and Other Fields
- Perceptron Hypothesis Set
- Perceptron Learning Algorithm (PLA)
- Guarantee of PLA

From Learning to Machine Learning

learning: acquiring skill
with experience accumulated from observations

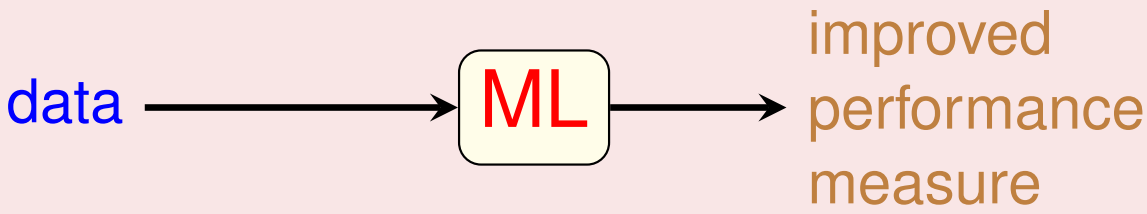
machine learning: acquiring skill
with experience accumulated/**computed** from data

What is skill?

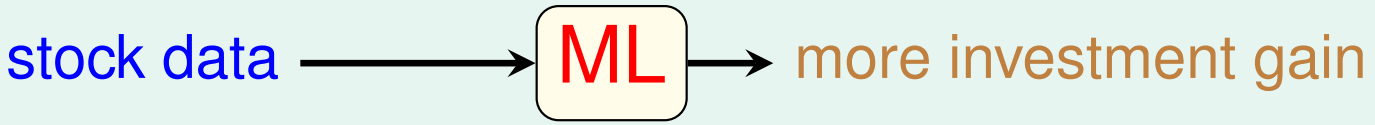
A More Concrete Definition

skill
$\Leftrightarrow$ improve some performance measure (e.g. prediction accuracy)

machine learning: improving some performance measure
with experience **computed** from data



An Application in Computational Finance



Why use machine learning?

Yet Another Application: Tree Recognition

- 'define' trees and hand-program: **difficult**
- learn from data (observations) and recognize: a **3-year-old can do so**
- 'ML-based tree recognition system' can be **easier to build** than hand-programmed system

ML: an **alternative route** to build complicated systems

The Machine Learning Route

ML: an **alternative route** to build complicated systems

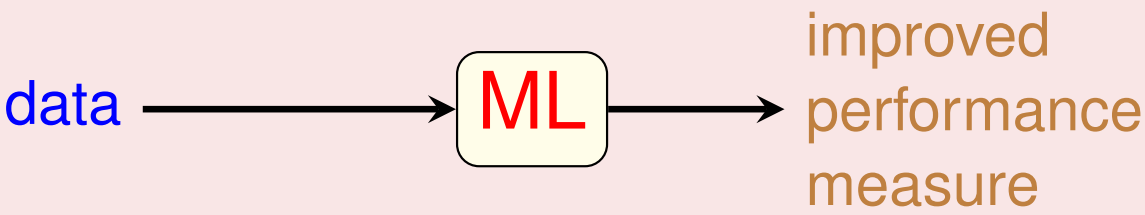
Some Use Scenarios

- when human cannot program the system manually
  —navigating on Mars
- when human cannot 'define the solution' easily
  —speech/visual recognition
- when needing rapid decisions that humans cannot do
  —high-frequency trading
- when needing to be user-oriented in a massive scale
  —consumer-targeted marketing

Give a **computer** a fish, you feed it for a day;
teach it how to fish, you feed it for a lifetime. :-)

Key Essence of Machine Learning

machine learning: improving some performance measure with experience **computed** from data



1. exists some 'underlying pattern' to be learned
   —so 'performance measure' can be improved
2. but no programmable (easy) definition
   —so 'ML' is needed
3. somehow there is data about the pattern
   —so ML has some 'inputs' to learn from

key essence: help decide whether to use ML

Questions?

ML Applications: Medicine

data → ML → skill

By DataBase Center for Life Science;
licensed under CC BY 4.0 via Wikimedia Commons

for computer-assisted diagnosis

- data:
  - **patient status**
  - **past diagnosis from doctors**
- skill: dialogue system that **efficiently identifies disease of patient**

my student's earlier work
as intern @ HTC DeepQ

ML-based Applications: Communication

for 4G LTE communication

- data:
  - **channel information** (the channel matrix representing mutual information)
  - **configuration** (precoding, modulation, etc.) that reaches the highest throughput
- skill: predict **best configuration to the base station** in a new environment

my student's earlier work as intern @ MTK

ML-based Applications: Manufacturing

data → ML → skill

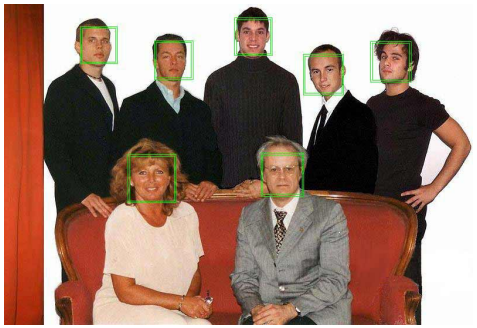
for PCB fault detection

- data: **PCB images of normal and abnormal PCBs** & maybe human-marked faulty locations
- skill: predict **which PCBs are faulty**

ongoing research for smart factory

ML-based Applications: Security

data → ML → skill



original picture by F.U.S.I.A. assistant and derivative work by Sylenius via Wikimedia Commons

face recognition

- data: **faces and non-faces**
- skill: predict **which boxes contain faces**

mature ML technique, but often need **tuning** for different needs

ML-based skill Applications: Education

- data: students' records on quizzes on a Math tutoring system
- skill: predict whether a student can give a correct answer to another quiz question

A Possible ML Solution

answer correctly $\approx$ ⟦recent strength of student $>$ difficulty of question⟧

- give ML 9 million records from 3000 students
- ML determines (**reverse-engineers**) strength and difficulty automatically

key part of the **world-champion** system from National Taiwan Univ. in KDDCup 2010

ML-based skill Applications: Recommender System (1/2)

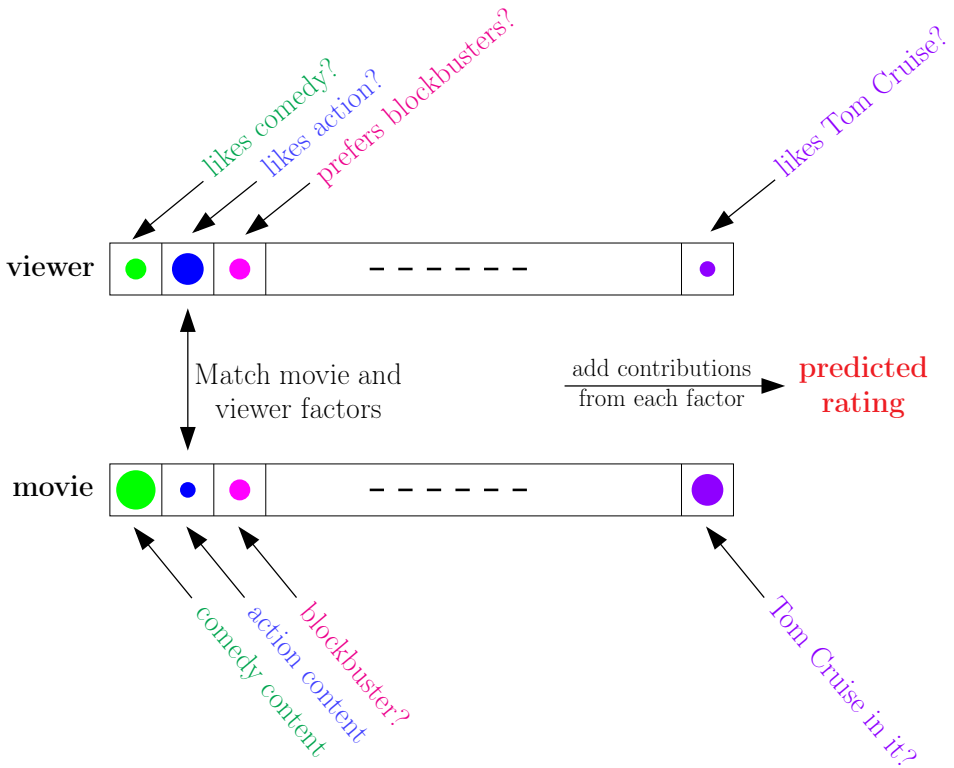
- data: how many users have rated some movies
- skill: predict how a user would rate an unrated movie

A Hot Problem

- competition held by Netflix in 2006
  - 100,480,507 ratings that 480,189 users gave to 17,770 movies
  - 10% improvement = **1 million dollar prize**
- similar competition (movies $\rightarrow$ songs) held by Yahoo! in KDDCup 2011
  - 252,800,275 ratings that 1,000,990 users gave to 624,961 songs

How can machines **learn our preferences**?

ML-based skill Applications: Recommender System (2/2)



A Possible ML Solution

- pattern:
  rating $\leftarrow$ viewer/movie factors
- learning:
  known rating
  $\rightarrow$ learned factors
  $\rightarrow$ unknown rating prediction

key part of the **world-champion** (again!) system from National Taiwan Univ. in KDDCup 2011

Questions?

Components of Learning: Metaphor Using Credit Approval

Applicant Information

age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000

unknown pattern to be learned:
'approve credit card good for bank?'

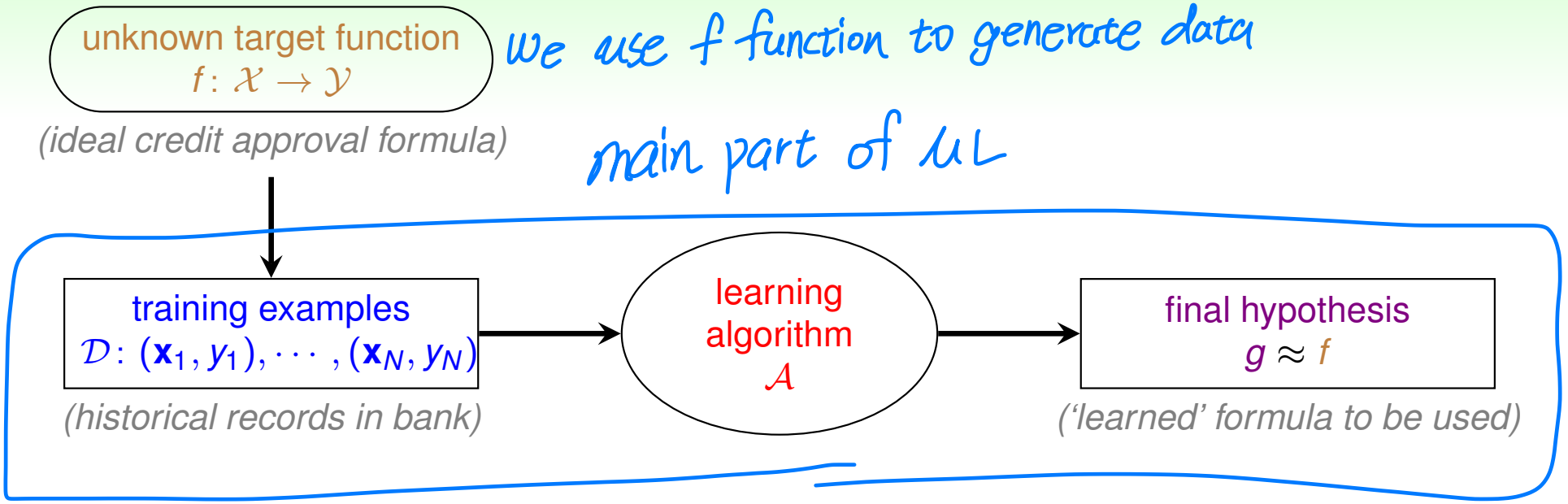
Formalize the Learning Problem

Basic Notations

- input: $\mathbf{x} \in \mathcal{X}$ (customer application)
- output: $y \in \mathcal{Y}$ (good/bad after approving credit card)
- unknown pattern to be learned $\Leftrightarrow$ target function: $f : \mathcal{X} \to \mathcal{Y}$ (ideal credit approval formula)
- data $\Leftrightarrow$ training examples: $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$ (historical records in bank)
- hypothesis $\Leftrightarrow$ skill with hopefully good performance: $g : \mathcal{X} \to \mathcal{Y}$ ('learned' formula to be used)

$$\{(\mathbf{x}_n, y_n)\} \text{ from } f \longrightarrow \boxed{\text{ML}} \longrightarrow g$$

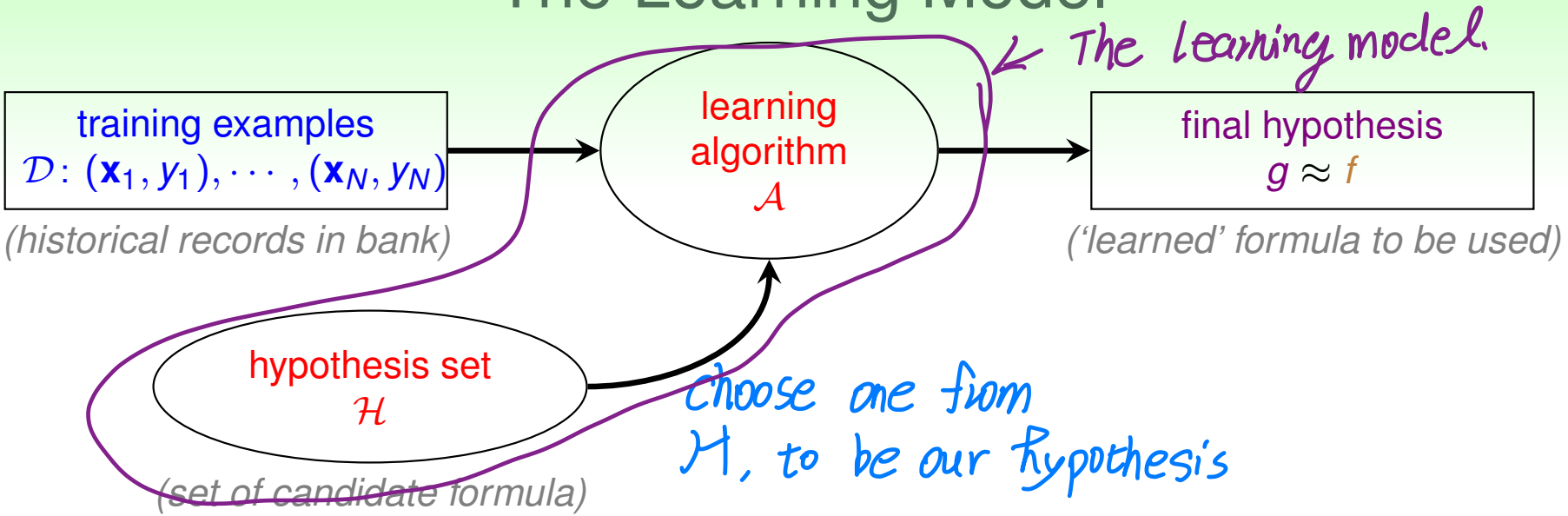
Learning Flow for Credit Approval



- target f **unknown** (real world)
  (i.e. no programmable definition)
- hypothesis g hopefully $\approx f$
  but possibly **different** from f
  (perfection 'impossible' when f unknown)

What does g look like?

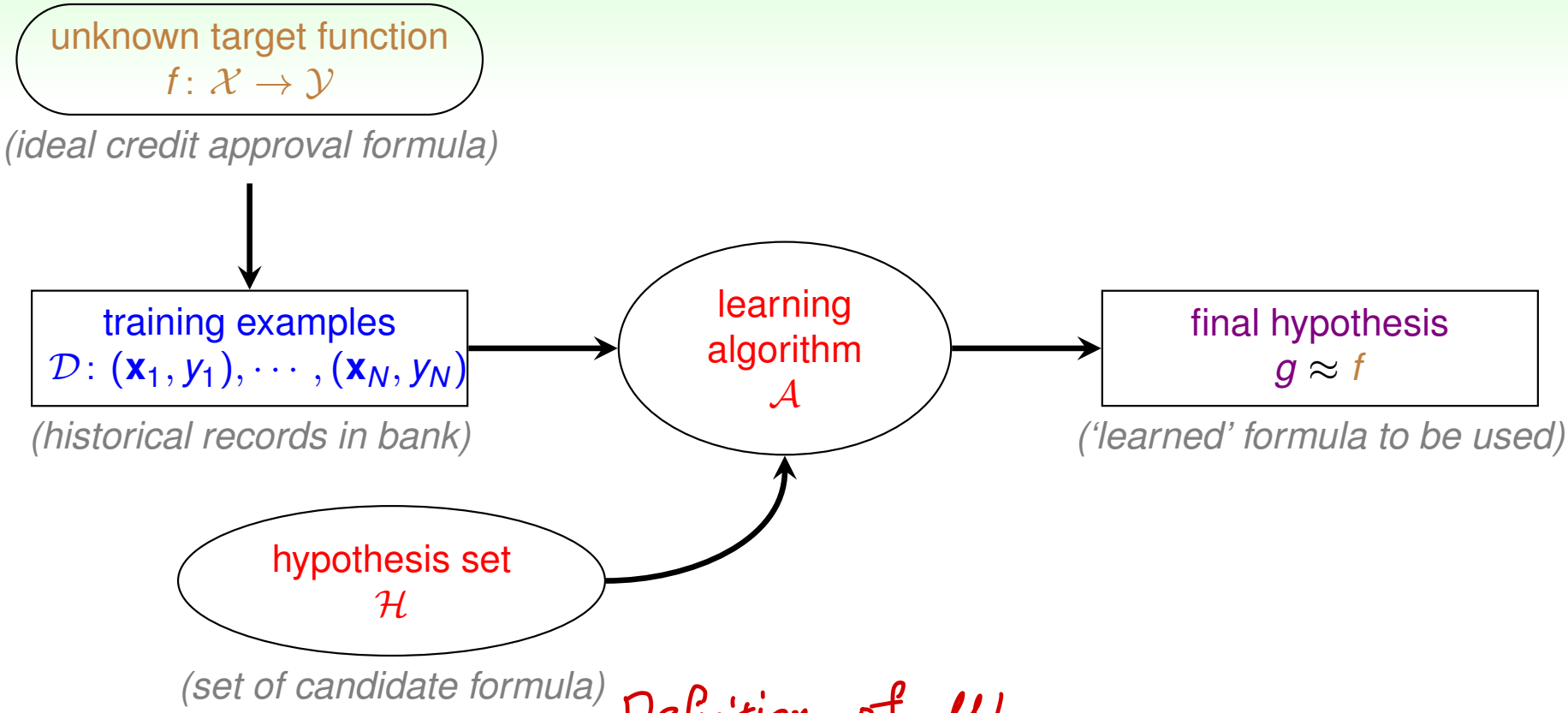
The Learning Model



- assume $g \in \mathcal{H} = \{h_k\}$, i.e. approving if
  - h_1: annual salary > NTD 800,000
  - h_2: debt > NTD 100,000 (really?)
  - h_3: year in job $\leq$ 2 (really?)
- hypothesis set $\mathcal{H}$:
  - can contain **good or bad hypotheses**
  - up to $\mathcal{A}$ to pick the 'best' one as g

learning model = $\mathcal{A}$ and $\mathcal{H}$

Practical Definition of Machine Learning



Definition of ML

machine learning:
use **data** to compute **hypothesis** g
that approximates **target** f

Questions?

Machine Learning and Data Mining

Machine Learning	Data Mining
use data to compute hypothesis g that approximates target f	use **(huge)** data to **find property** that is interesting

- if 'interesting property' **same as** 'hypothesis that approximate target'
  —**ML = DM** (usually what KDDCup does)
- if 'interesting property' **related to** 'hypothesis that approximate target'
  —**DM can help ML, and vice versa** (often, but not always)
- traditional DM also focuses on **efficient computation in large database** ⇒ big database.

difficult to distinguish ML and DM in reality

Machine Learning and Statistics

Machine Learning

use data to compute hypothesis g that approximates target f

Statistics

use data to **make inference about an unknown process**

- g is an inference outcome; f is something unknown —statistics **can be used to achieve ML**
- traditional statistics also focus on **provable results with math assumptions**, and care less about computation

statistics: many useful tools for ML

Machine Learning and Artificial Intelligence

Machine Learning

use data to compute hypothesis g that approximates target f

Artificial Intelligence

compute **something that shows intelligent behavior**

- $g \approx f$ is something that shows intelligent behavior
  —**ML can realize AI**, among other routes
- e.g. chess playing
  - traditional AI: game tree
  - ML for AI: 'learning from board data'

ML is one possible route to realize AI

Machine Learning Connects (Big) Data and AI

skill ≈ artificial intelligence

(big) data → ML → artificial intelligence

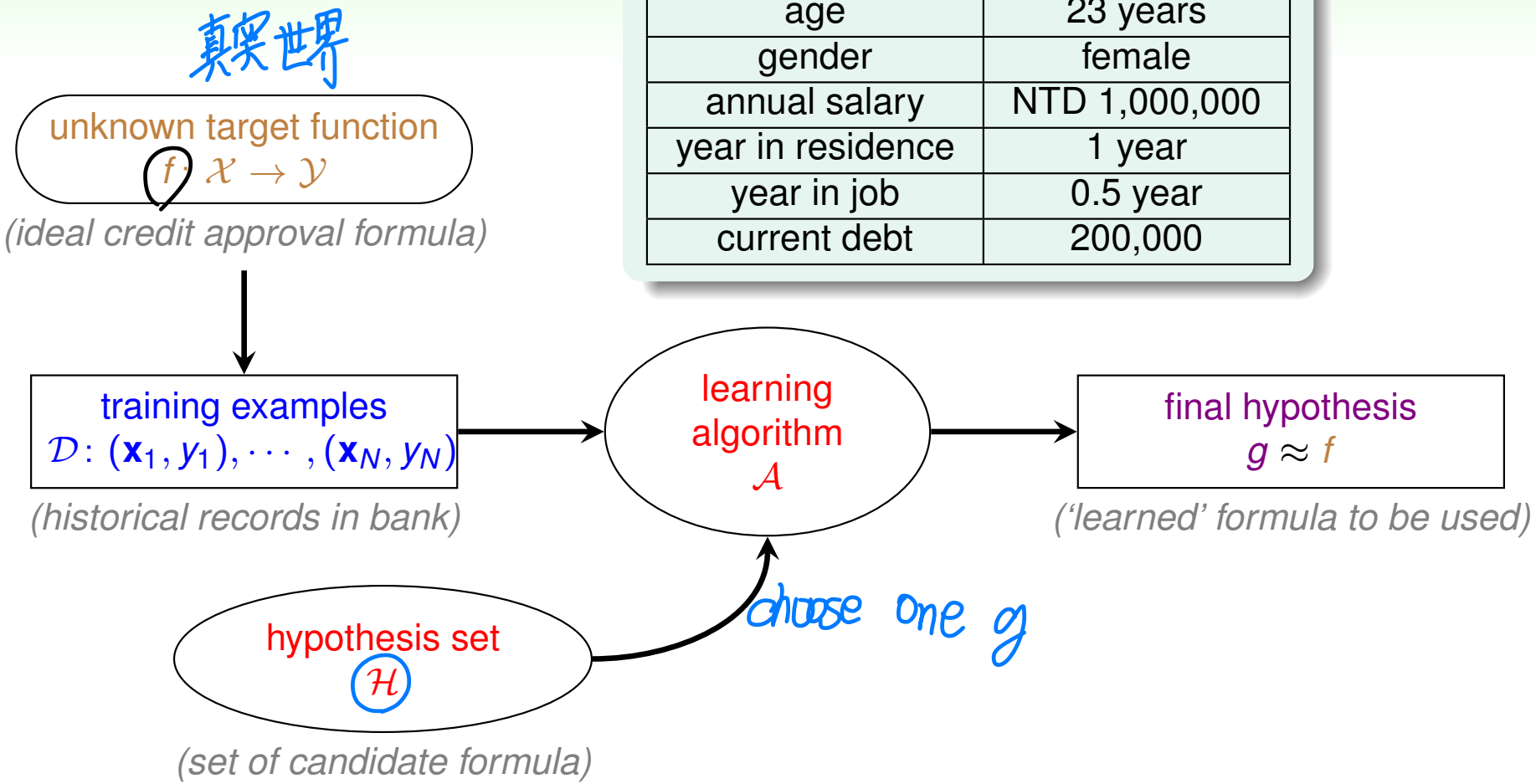
ML not the only tools, but
a **popular family of tools**

Questions?

Credit Approval Problem Revisited

Applicant Information

age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000



what hypothesis set can we use?

A Simple Hypothesis Set: the 'Perceptron'

age	23 years
annual salary	NTD 1,000,000
year in job	0.5 year
current debt	200,000

$d = 4$

- For $\mathbf{x} = (x_1, x_2, \cdots, x_d)$ '**features of customer**', compute a weighted 'score' and

  number of features

  score our customers — we can change this

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

  代表重要性

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

  it's just a if else judgement

- $\mathcal{Y}$: $\{+1(\textbf{good}), -1(\textbf{bad})\}$, 0 ignored—linear formula $h \in \mathcal{H}$ are

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

changable parts

what if it's exactly equal to zero? ⇒ not really possible. (+ or −)

it's called 'perceptron' hypothesis historically

Vector Form of Perceptron Hypothesis

把 threshold 當成特定的 weight

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right) \leftarrow \text{too complicated}$$

$$= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + \underbrace{(-\text{threshold})}_{w_0} \cdot \underbrace{(+1)}_{x_0}\right)$$

整合進來

dummy x_0

$$= \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right) \text{ intergral two terms}$$

$$= \text{sign}\left(\mathbf{w}^T \mathbf{x}\right) \Rightarrow \text{sign}\left([w_0\ w_1\ w_2 \cdots w_d]\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}\right)$$

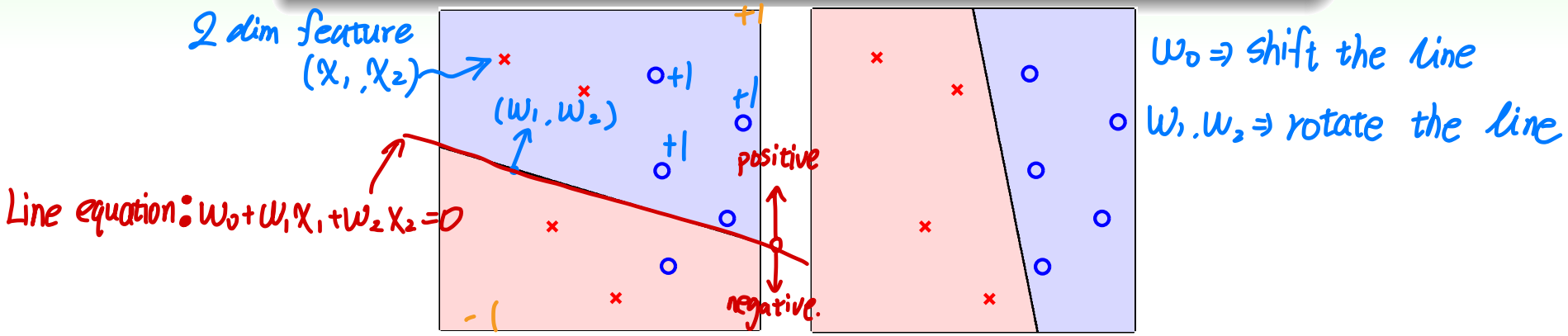
- each 'tall' **w** represents a hypothesis h & is multiplied with 'tall' **x** —**will use tall versions to simplify notation**

what do perceptrons h 'look like'?

Perceptrons in $\mathbb{R}^2$, see our perception in $\mathbb{R}^2$

$$h(\mathbf{x}) = \text{sign}\left(w_0 + w_1 x_1 + w_2 x_2\right)$$

threshold



- customer features $\mathbf{x}$: points on the plane (or points in $\mathbb{R}^d$)
- labels y: $\circ$ (+1), $\times$ (-1)
- hypothesis h: **lines** (or hyperplanes in $\mathbb{R}^d$)
  —positive on one side of a line, negative on the other side
- different line classifies customers differently

perceptrons $\Leftrightarrow$ **linear (binary) classifiers**

↳ A Line on 2D plane

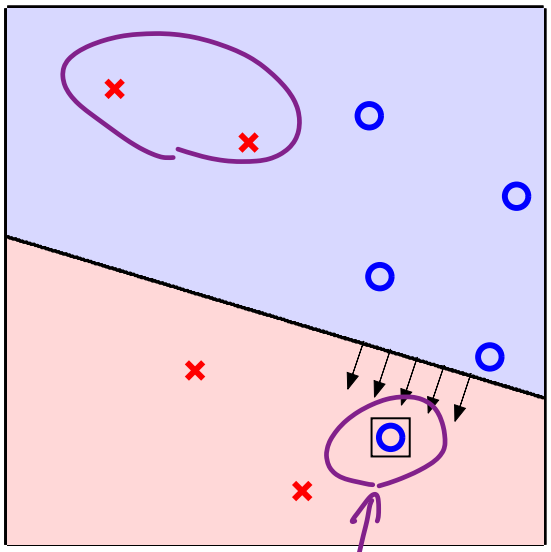
Questions?

Select g from $\mathcal{H}$

$\mathcal{H}$ = all possible perceptrons, g = ?

在traing data上結果完全一樣

- want: $g \approx f$ (hard when f unknown)
- almost necessary: $g \approx f$ on $\mathcal{D}$, ideally $g(\mathbf{x}_n) = f(\mathbf{x}_n) = y_n$ ⇒ fit well at training data
- difficult: $\mathcal{H}$ is of **infinite** size
- idea: start from some g_0, and 'correct' its mistakes on $\mathcal{D}$



training loss

* try to make line closer to the wrong data.

will represent g_0 by its weight vector $\mathbf{w}_0$

vector

choose a hypothesis ⇒ choose a better weight vector

Perceptron Learning Algorithm

start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $(\mathbf{x}_{n(t)}, y_{n(t)})$

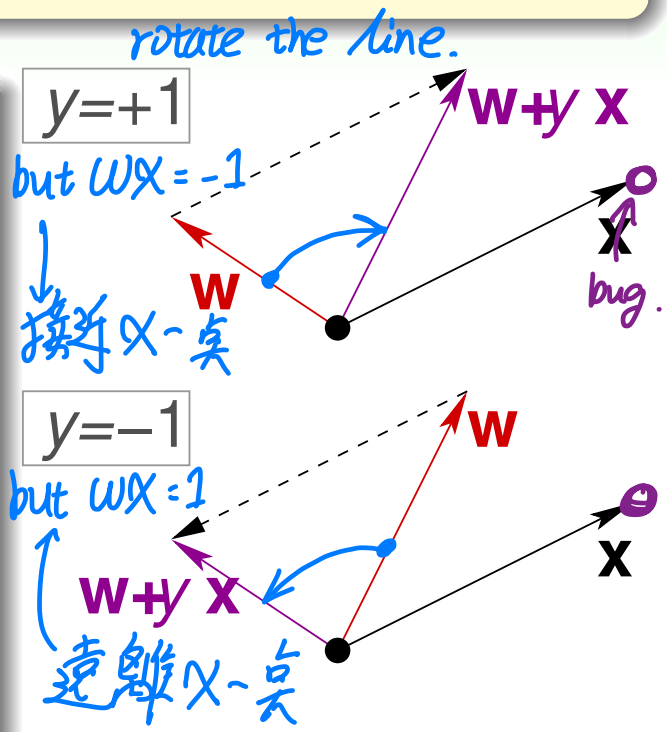
$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

. . . until no more mistakes

return last $\mathbf{w}$ (called $\mathbf{w}_{PLA}$) as g



That's it!

—*A fault confessed is half redressed.* :-)

Practical Implementation of PLA

start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

HW needed

Cyclic PLA ← update until no mistake.

For $t = 0, 1, \ldots$ 輪流看所有的点,直到沒有人犯錯

1. find **the next** mistake of $\mathbf{w}_t$ called $(\mathbf{x}_{n(t)}, y_{n(t)})$

$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

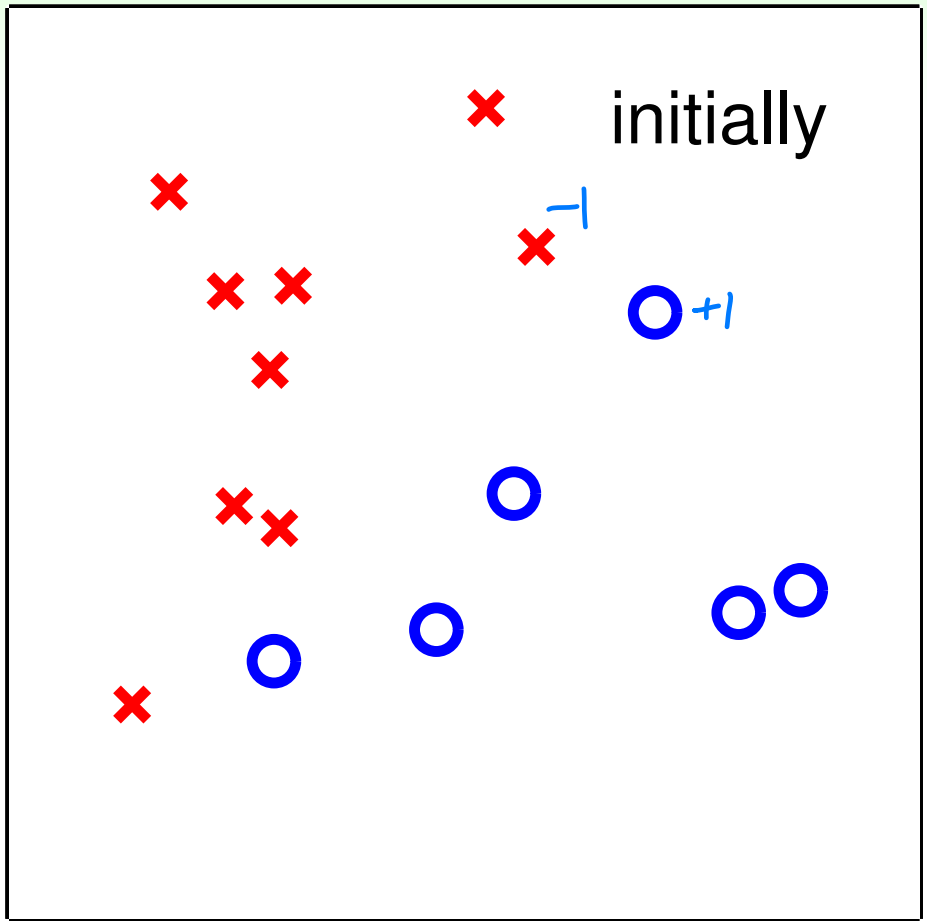
2. correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

. . . until **a full cycle of not encountering mistakes**

next can follow naïve cycle $(1, \cdots, N)$
or precomputed random cycle

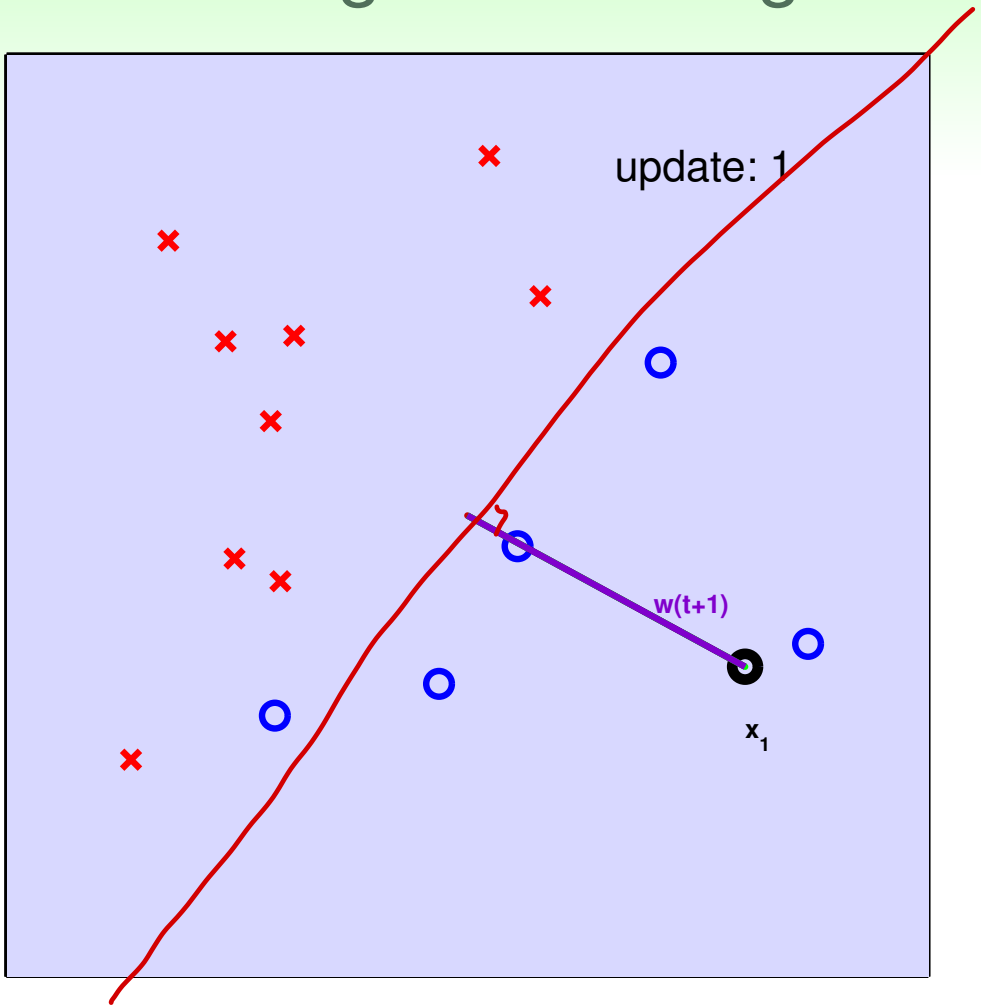
Seeing is Believing



worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

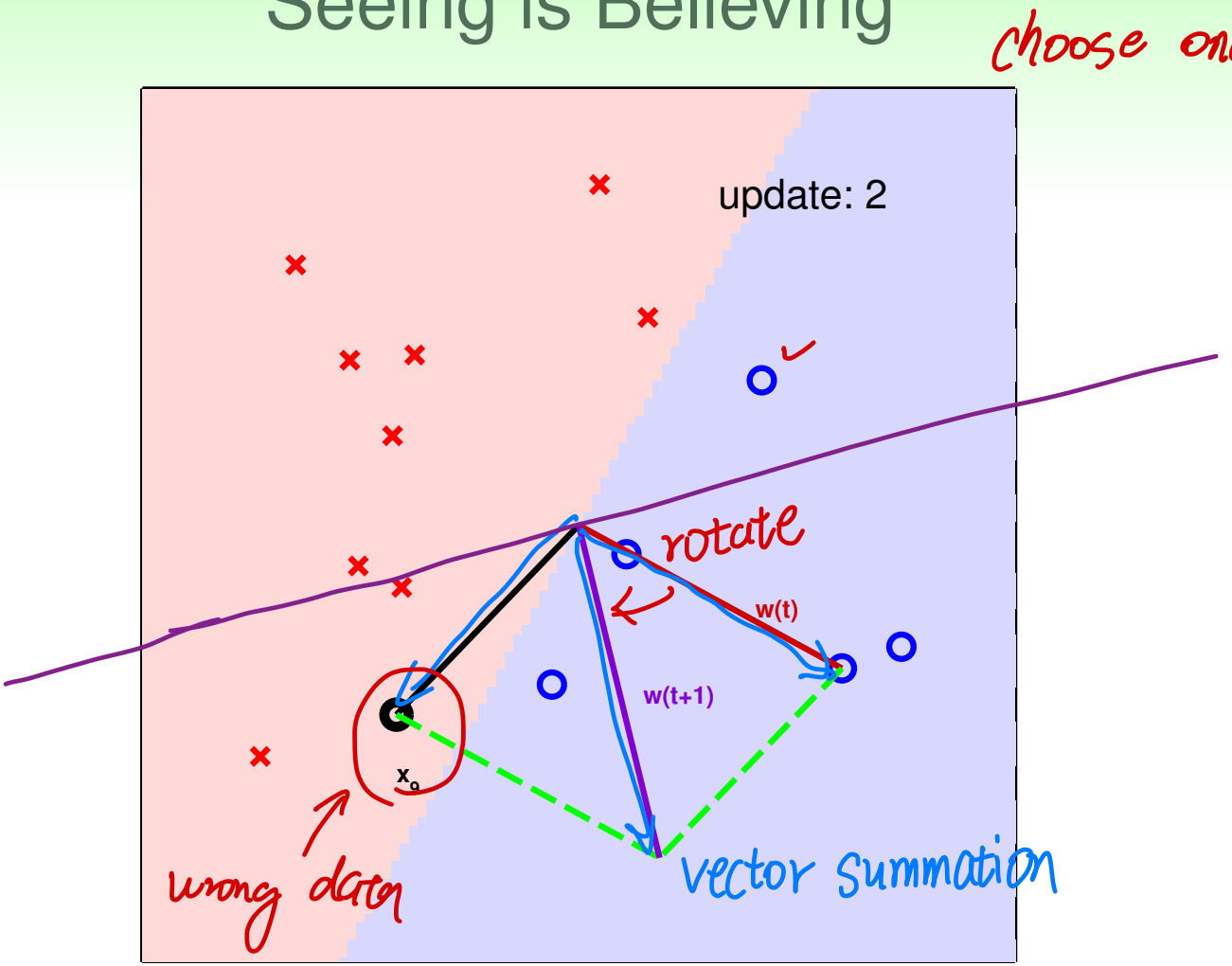
Seeing is Believing



worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

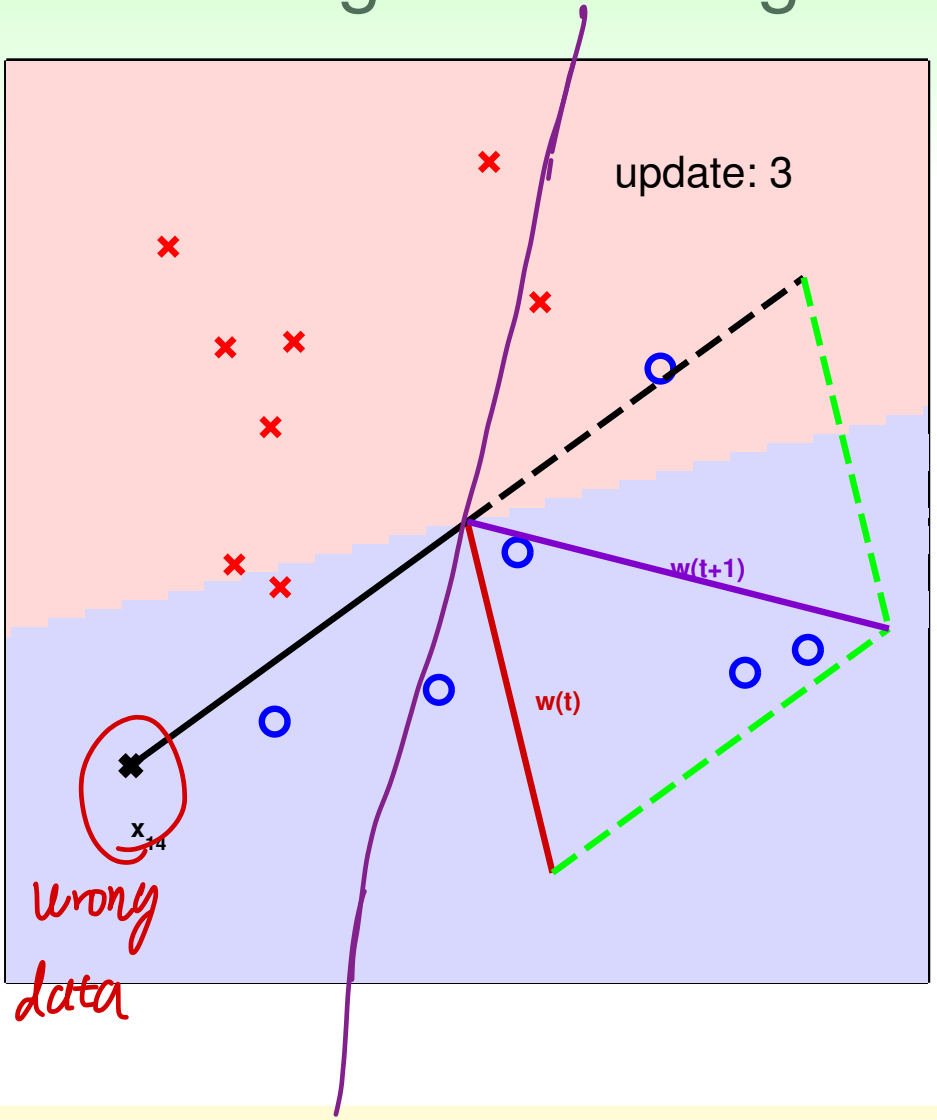
Seeing is Believing



worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

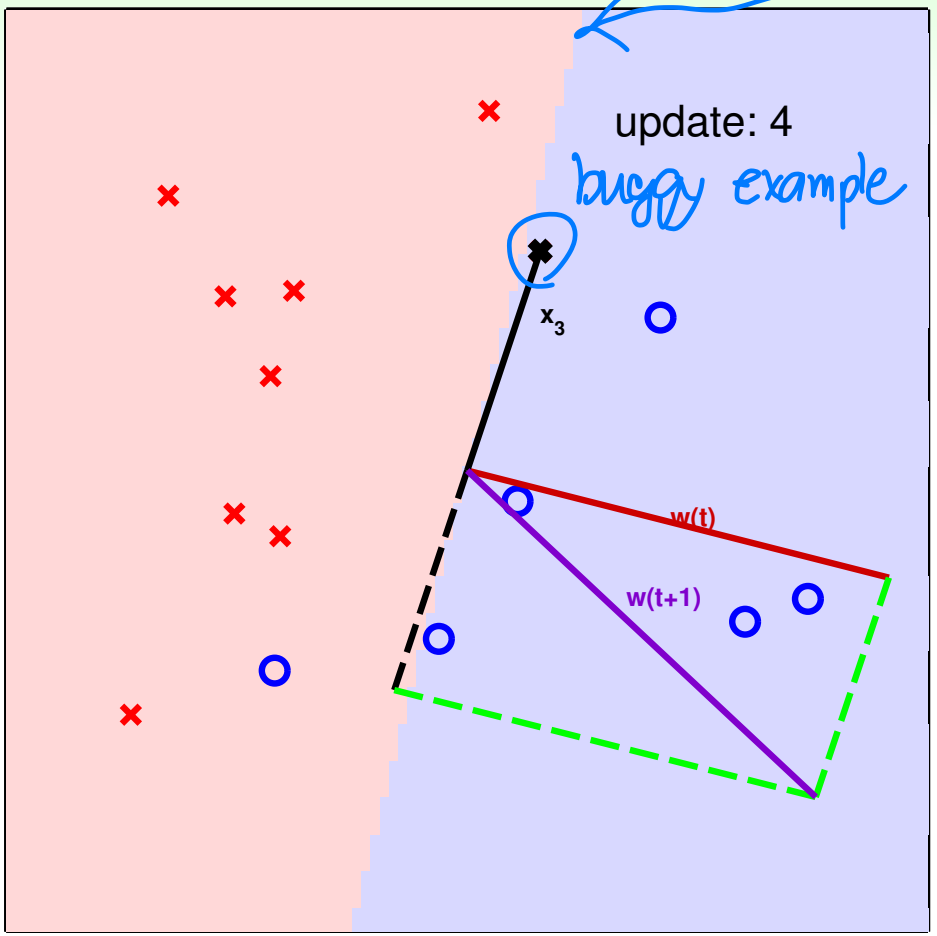
Seeing is Believing



worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

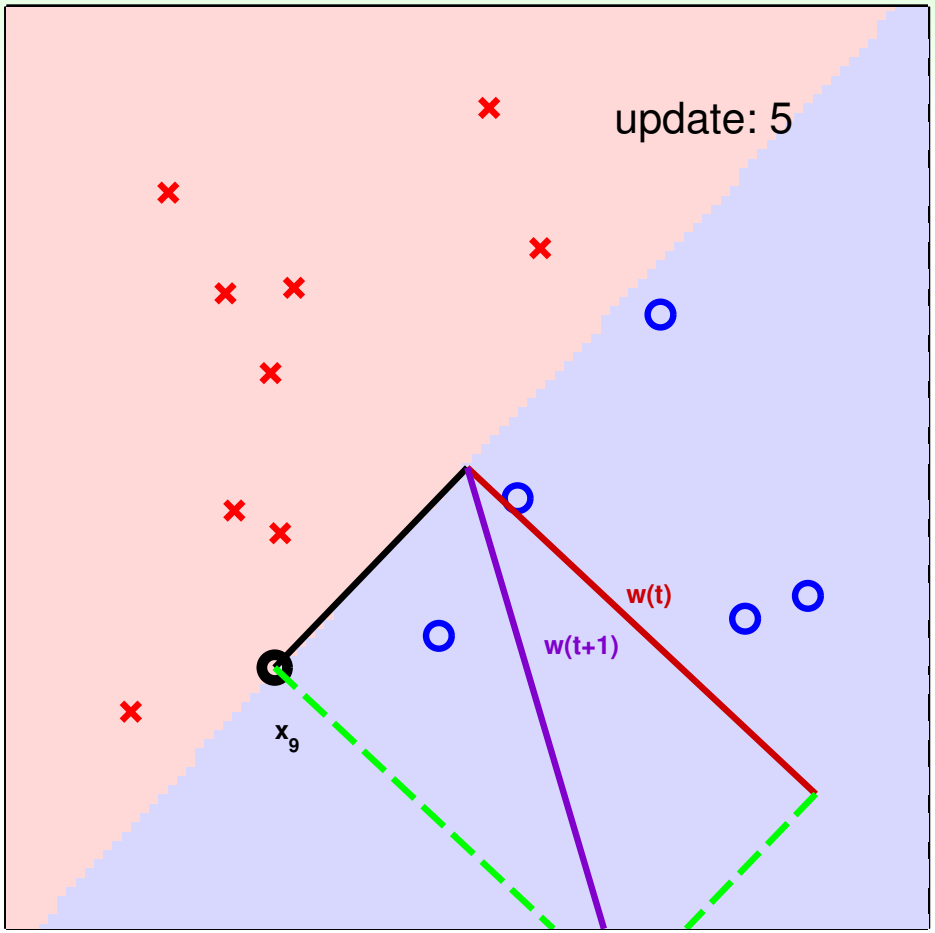
Seeing is Believing



worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

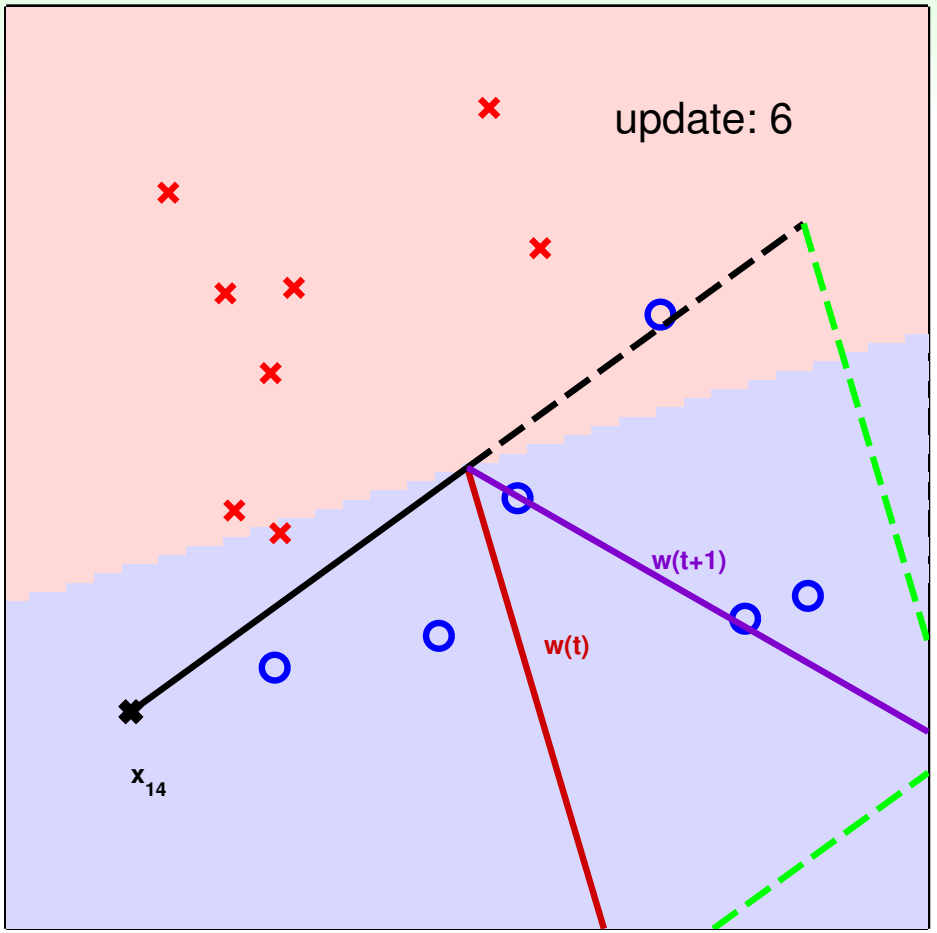
Seeing is Believing



一次只看一個点

worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

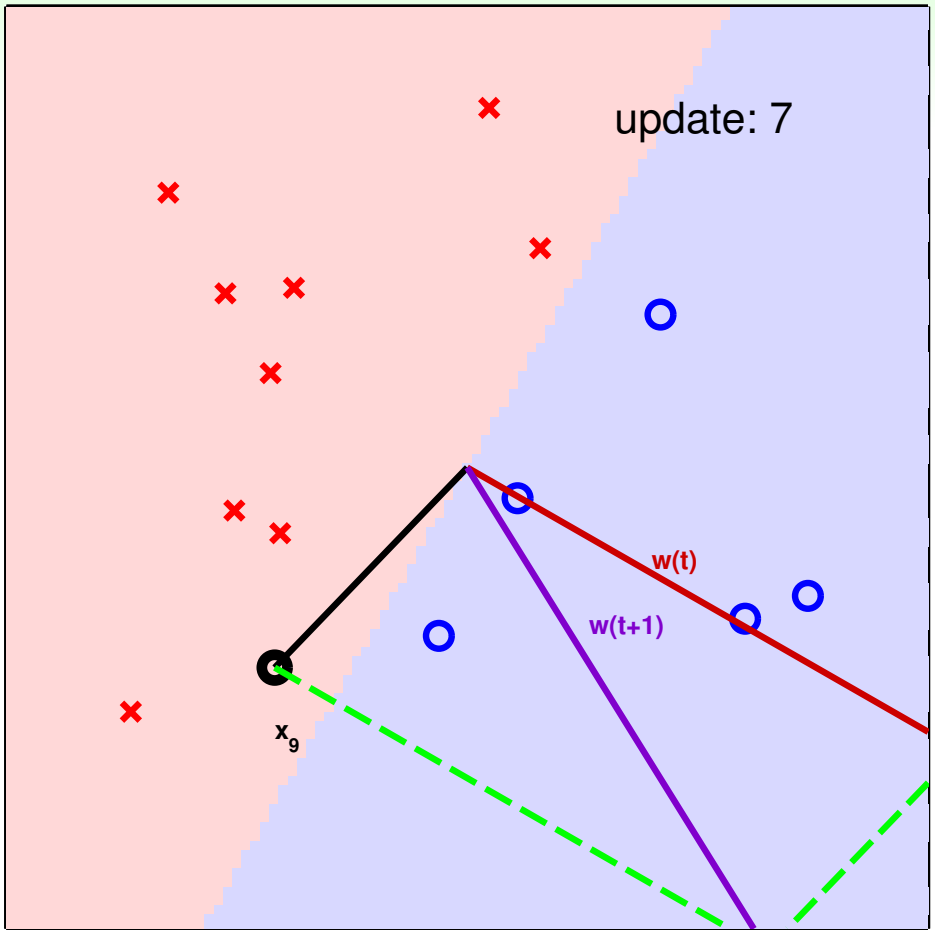
Seeing is Believing



worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

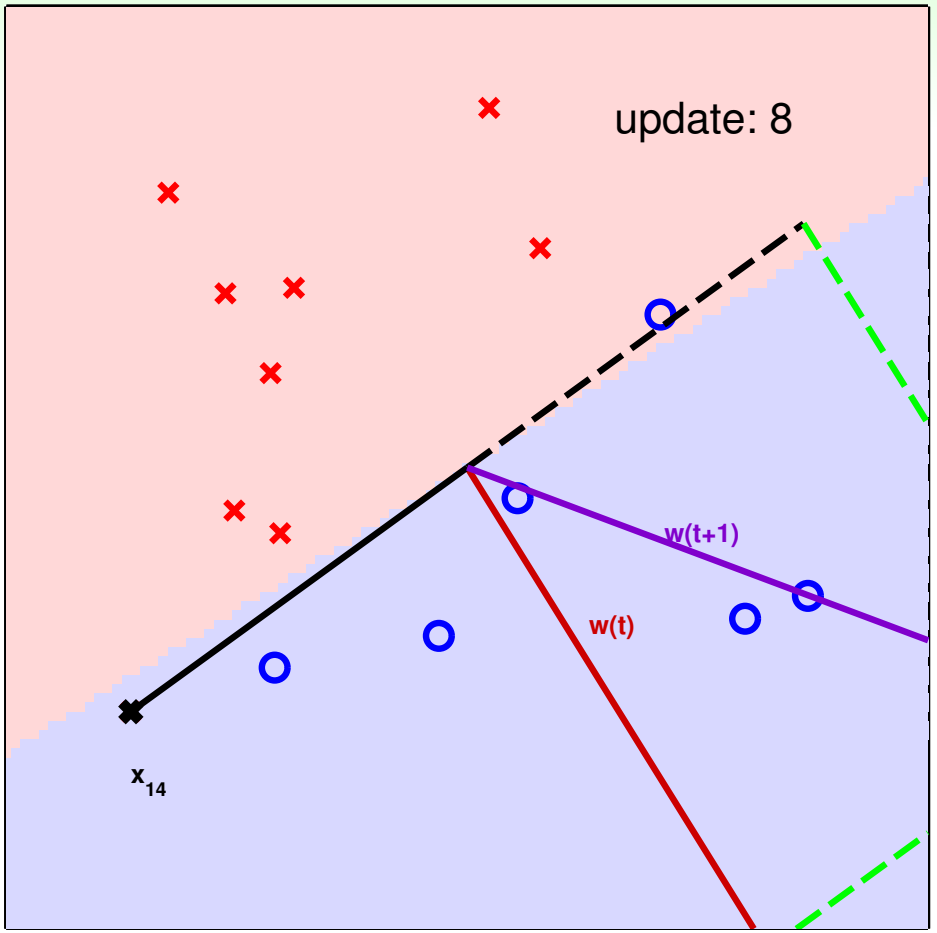
Seeing is Believing



worked like a charm with < 20 lines!!

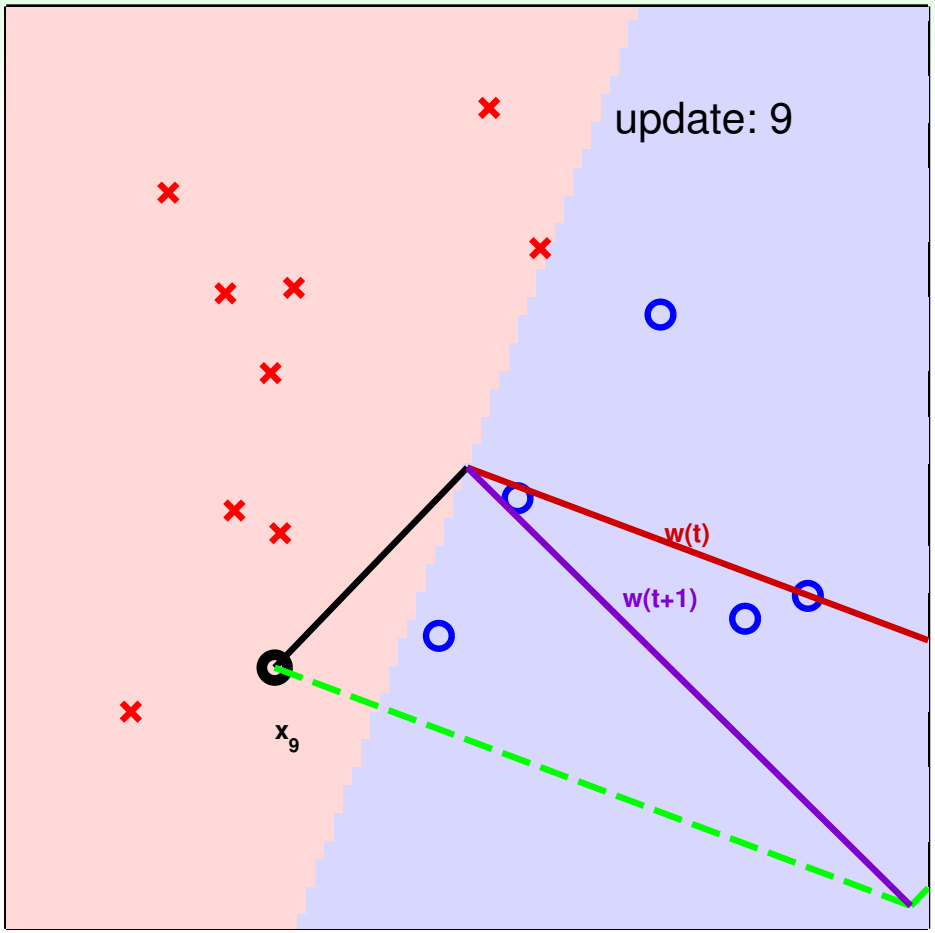
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



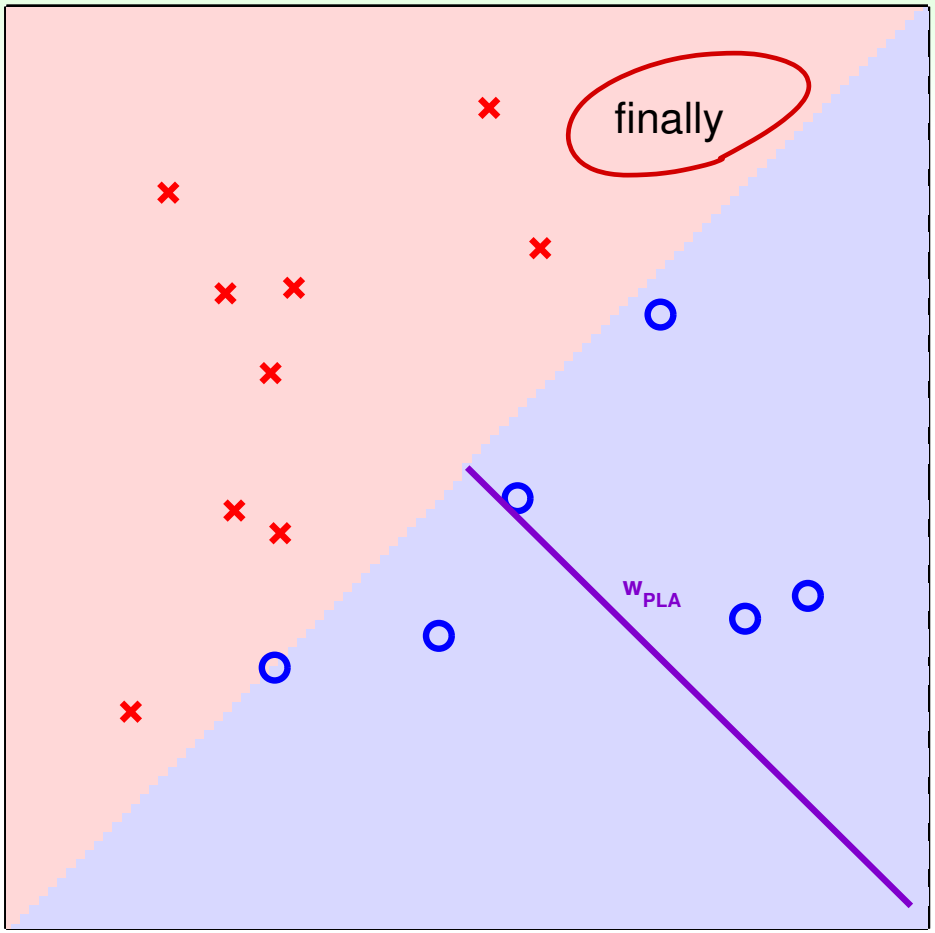
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



make threshold weight very unimportant

worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

Some Remaining Issues of PLA

'correct' mistakes on $\mathcal{D}$ **until no mistakes**

Algorithmic: halt (with no mistake)?

- naïve cyclic: ??
- random cyclic: ??
- other variant: ??

Learning: $g \approx f$?

- on $\mathcal{D}$, if halt, yes (no mistake)
- outside $\mathcal{D}$: ?? ← overfitting?
- if not halting: ??

[to be shown] if (...), after 'enough' corrections,
any PLA variant halts

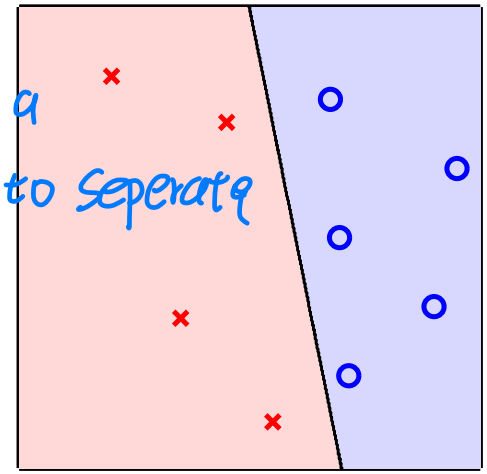
Questions?

Linear Separability

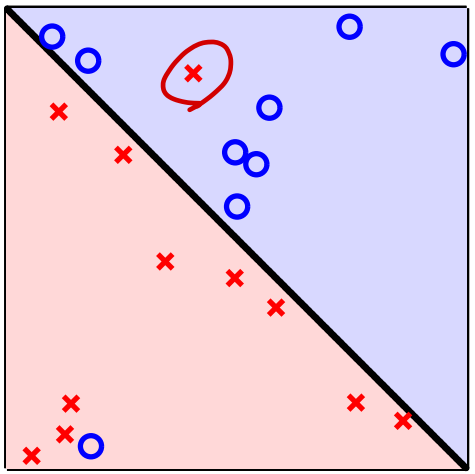
- **if** PLA halts (i.e. no more mistakes), ← Is this true
  (necessary condition) $\mathcal{D}$ allows some $\mathbf{w}$ to make no mistake
- call such $\mathcal{D}$ **linear separable**

dataset must allow it

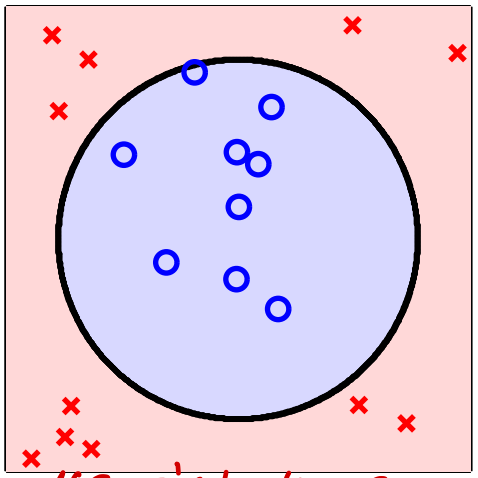
exist a line to seperate data



(linear separable)



(not linear separable)



use circle to separate

(not linear separable)

assume linear separable $\mathcal{D}$,
does PLA always **halt**? ← need to proof it

PLA Fact: $\mathbf{w}_t$ Gets More Aligned with $\mathbf{w}_f$

linear separable $\mathcal{D}$ $\Leftrightarrow$ **exists perfect $\mathbf{w}_f$ such that** $y_n = \text{sign}(\mathbf{w}_f^T \mathbf{x}_n)$

*need to proof W_t will be closed to W_f for every iteration

- $\mathbf{w}_f$ perfect hence every $\mathbf{x}_n$ correctly away from line:

the perfect weight

label prediction

$$y_{n(t)} \mathbf{w}_f^T \mathbf{x}_{n(t)} \geq \min_n y_n \mathbf{w}_f^T \mathbf{x}_n > 0$$

因為是 W_f 是 perfect line 所以可遠. 二者同号

not just buggy example

離 decision boundary 最近的 data

- $\mathbf{w}_f^T \mathbf{w}_t \uparrow$ by updating with any $(\mathbf{x}_{n(t)}, y_{n(t)})$

perfect current

inner product to check W_f close to W_{t+1}? update rule.

$$\begin{aligned} \mathbf{w}_f^T \mathbf{w}_{t+1} &= \mathbf{w}_f^T \left(\mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}\right) \\ &\geq \mathbf{w}_f^T \mathbf{w}_t + \min_n y_n \mathbf{w}_f^T \mathbf{x}_n \\ &> \mathbf{w}_f^T \mathbf{w}_t + 0. \end{aligned}$$

離线最近的人

inner product of W_f^T and W_t getting bigger and bigger ⇒ closer and closer

$\mathbf{w}_t$ appears more aligned with $\mathbf{w}_f$ after update

(really?) ⇒ but scaling the vector also increase inner product

PLA Fact: $\mathbf{w}_t$ Does Not Grow Too Fast

$\mathbf{w}_t$ changed only when mistake

$$\Leftrightarrow \text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)} \Leftrightarrow y_{n(t)} \mathbf{w}_t^T \mathbf{x}_{n(t)} \leq 0$$

異号→有錯

label　prediction

- mistake 'limits' $\|\mathbf{w}_t\|^2$ growth, even when updating with 'longest' $\mathbf{x}_n$

update rule

$$\begin{aligned}
\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\
&= \|\mathbf{w}_t\|^2 + 2y_{n(t)}\mathbf{w}_t^T\mathbf{x}_{n(t)} + \|y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\
&\leq \|\mathbf{w}_t\|^2 + 0 + \|y_{n(t)}\mathbf{x}_{n(t)}\|^2 \quad \leftarrow \text{犯錯的点的長度} \\
&\leq \|\mathbf{w}_t\|^2 + \max_n \|y_n\mathbf{x}_n\|^2 \quad \leftarrow \text{最長的 } \mathbf{x}_n
\end{aligned}$$

*結論 $\|w_t\|^2$ 每次 update 最多增長 $\max_n \|x_n\|^2$ 這麼多

Let's put everything together!

PLA Mistake Bound

2 proof in previous slides

inner product grows fast

$$\mathbf{w}_f^T \mathbf{w}_{t+1} \geq \mathbf{w}_f^T \mathbf{w}_t + \underbrace{\min_n y_n \mathbf{w}_f^T \mathbf{x}_n}_{\rho}$$

length^2 grows slow

$$\|\mathbf{w}_{t+1}\|^2 \leq \|\mathbf{w}_t\|^2 + \underbrace{\max_n \|\mathbf{x}_n\|^2}_{R^2}$$

Magic Chain!

inner product grows by ρ

$$\begin{aligned}
\mathbf{w}_f^T \mathbf{w}_1 &\geq \mathbf{w}_f^T \mathbf{w}_0 + \rho \\
\mathbf{w}_f^T \mathbf{w}_2 &\geq \mathbf{w}_f^T \mathbf{w}_1 + \rho \\
\mathbf{w}_f^T \mathbf{w}_3 &\geq \mathbf{w}_f^T \mathbf{w}_2 + \rho \\
&\cdots \\
\mathbf{w}_f^T \mathbf{w}_T &\geq \mathbf{w}_f^T \mathbf{w}_{T-1} + \rho
\end{aligned}$$

transpose; iteration number

$$\Rightarrow \mathbf{w}_f^T \mathbf{w}_T \geq \mathbf{w}_f^T \mathbf{w}_0 + T \cdot \rho$$

Magic Chain!

grow of R^2

$$\begin{aligned}
\|\mathbf{w}_1\|^2 &\leq \|\mathbf{w}_0\|^2 + R^2 \\
\|\mathbf{w}_2\|^2 &\leq \|\mathbf{w}_1\|^2 + R^2 \\
\|\mathbf{w}_3\|^2 &\leq \|\mathbf{w}_2\|^2 + R^2 \\
&\cdots \\
\|\mathbf{w}_T\|^2 &\leq \|\mathbf{w}_{T-1}\|^2 + R^2
\end{aligned}$$

$$\Rightarrow \|\mathbf{w}_T\|^2 \leq \|\mathbf{w}_0\|^2 + T \cdot R^2$$

start from $\mathbf{w}_0 = \mathbf{0}$, after T mistake corrections,

Transpose. num of iteration

$$1 \geq \frac{\mathbf{w}_f^T}{\|\mathbf{w}_f\|} \frac{\mathbf{w}_T}{\|\mathbf{w}_T\|} \geq \frac{T\rho}{1\sqrt{T}R} \Longrightarrow T \leq \left(\frac{R}{\rho}\right)^2$$

$\cos\theta$; upper bound of # of mistake correction

if PLA run more than $\left(\frac{R}{\rho}\right)^2$, it's not linear seperable

More about PLA

Guarantee

as long as linear separable and correct by mistake

- inner product of $\mathbf{w}_f$ and $\mathbf{w}_t$ grows fast; length of $\mathbf{w}_t$ grows slowly
- PLA 'lines' are more and more aligned with $\mathbf{w}_f \Rightarrow$ halts

Pros

simple to implement, fast, works in any dimension d

Cons

- **'assumes' linear separable $\mathcal{D}$** to halt
  —property unknown in advance (no need for PLA if we know $\mathbf{w}_f$)
- not fully sure **how long halting takes** (ρ depends on $\mathbf{w}_f$)
  —though practically fast

don't know if the perfect w_f exist

but anyway, a **simple and promising** start

Summary

1. **When** Can Machines Learn?

Lecture 1: Basics of Machine Learning

- What is Machine Learning
  use data to approximate target
- Applications of Machine Learning
  almost everywhere
- Components of Machine Learning
  $\mathcal{A}$ takes $\mathcal{D}$ and $\mathcal{H}$ to get g
- Machine Learning and Other Fields
  related to DM, Stats and AI
- Perceptron Hypothesis Set
  hyperplanes/linear classifiers in $\mathbb{R}^d$
- Perceptron Learning Algorithm (PLA)
  correct mistakes and improve iteratively
- Guarantee of PLA
  no mistake eventually if linear separable

- **next: other machine learning problems**