学习问题

刘 新 旺

Email: xinwangliu@nudt.edu.cn

国防科技大学 计算机学院
计算科学系 人工智能与大数据教研室

2019 年 10 月 15 日

Content

Learning: Acquiring skill

With experience accumulated from observations

observations $\longrightarrow$ **learning** $\longrightarrow$ skill

从“学习”到“机器学习”

Learning: Acquiring skill

With experience accumulated from observations

observations $\longrightarrow$ $\boxed{\textbf{learning}}$ $\longrightarrow$ skill

Machine Learning (ML): Acquiring skill

With experience accumulated/**computed** from data

data $\longrightarrow$ $\boxed{\textbf{machine learning}}$ $\longrightarrow$ skill

Learning: Acquiring skill

With experience accumulated from observations

observations ⟶ $\boxed{\textbf{learning}}$ ⟶ skill

Machine Learning (ML): Acquiring skill

With experience accumulated/**computed** from data

data ⟶ $\boxed{\textbf{machine learning}}$ ⟶ skill

$\boxed{\text{what is } \textbf{skill}}$?

skill

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

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

Machine Learning: improve some performance measure

With experience **computed** from data

data $\longrightarrow$ $\boxed{\textbf{machine learning}}$ $\longrightarrow$ $\boxed{\text{improved performance measure}}$

An application in Computational Finance

stock data $\longrightarrow$ $\boxed{\textbf{machine learning}}$ $\longrightarrow$ more investment gain

Content

机器学习的组成要素：以“信用卡申请”为例

申请者信息

age	23 years
gender	female
annual salary	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?”

Basic Notations

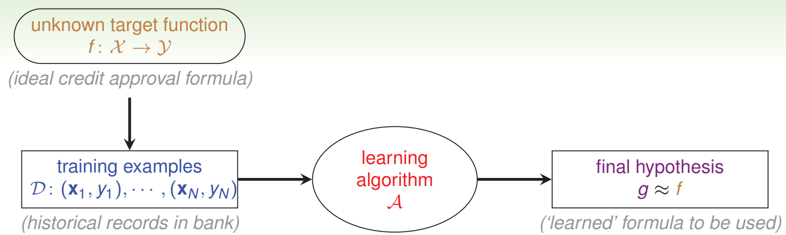
- input: $\mathbf{x} \in \mathcal{X}$ (customer application)
- output: $y \in \mathcal{Y}$ (good/bad after approving credit card)
- unknown pattern to be learned $\Leftrightarrow$ target function:
  $f: \mathcal{X} \mapsto \mathcal{Y}$ (ideal credit approval formula)
- data $\Leftrightarrow$ training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

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:
  $f: \mathcal{X} \mapsto \mathcal{Y}$ (ideal credit approval formula)
- data $\Leftrightarrow$ training examples: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$
  (historical records in bank)
- hypothesis $\Leftrightarrow$ skill with hopefully good performance:
  $g: \mathcal{X} \mapsto \mathcal{Y}$ (“learned” formula to be used)

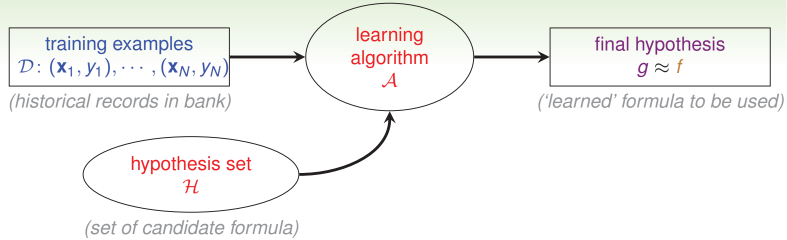
$\{(\mathbf{x}_n, y_n)\}$ from f $\longrightarrow$ $\boxed{\textbf{machine learning}}$ $\longrightarrow g$

信用卡申请的学习流程图



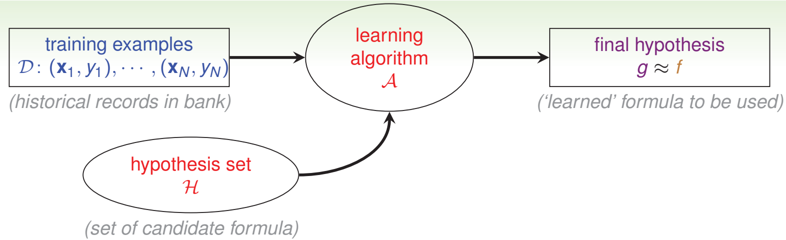
- target f **unknown**
  (i.e. no programmable definition)
- hypothesis g hopefully $\approx f$
  but possibly **different** from f
  (perfection 'impossible' when f unknown)

What does g look like?



- 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 ≤ 2 (really?)
- hypothesis set $\mathcal{H}$:
  - can contain **good or bad hypotheses**
  - up to $\mathcal{A}$ to pick the 'best' one as g

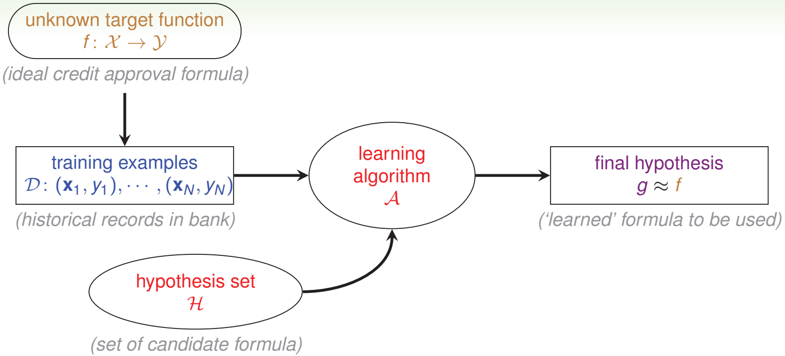
学习模型



- 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 ≤ 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 $\mathcal{H}$

机器学习的实用定义



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

How to use the four sets below to form a learning problem for song recommendation?

$$
\begin{aligned}
\mathcal{S}_1 &= [0, 100] \\
\mathcal{S}_2 &= \text{all possible (userid, songid) pairs} \\
\mathcal{S}_3 &= \text{all formula that 'multiplies' user factors \& song factors,} \\
&\quad\ \text{indexed by all possible combinations of such factors} \\
\mathcal{S}_4 &= \text{1,000,000 pairs of ((userid, songid), rating)}
\end{aligned}
$$

1. $\mathcal{S}_1 = \mathcal{X}, \mathcal{S}_2 = \mathcal{Y}, \mathcal{S}_3 = \mathcal{H}, \mathcal{S}_4 = \mathcal{D}$
2. $\mathcal{S}_1 = \mathcal{Y}, \mathcal{S}_2 = \mathcal{X}, \mathcal{S}_3 = \mathcal{H}, \mathcal{S}_4 = \mathcal{D}$
3. $\mathcal{S}_1 = \mathcal{D}, \mathcal{S}_2 = \mathcal{H}, \mathcal{S}_3 = \mathcal{Y}, \mathcal{S}_4 = \mathcal{X}$
4. $\mathcal{S}_1 = \mathcal{X}, \mathcal{S}_2 = \mathcal{D}, \mathcal{S}_3 = \mathcal{Y}, \mathcal{S}_4 = \mathcal{H}$

Machine Learning

use data to compute hypothesis g that approximates target f

Data Mining

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**

difficult to distinguish ML and DM in reality

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

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

Which of the following claim is not totally true?

1. machine learning is a route to realize artificial intelligence
2. machine learning, data mining and statistics all need data
3. data mining is just another name for machine learning
4. statistics can be used for data mining

Which of the following claim is not totally true?

1. machine learning is a route to realize artificial intelligence
2. machine learning, data mining and statistics all need data
3. data mining is just another name for machine learning
4. statistics can be used for data mining

Reference Answer: ③

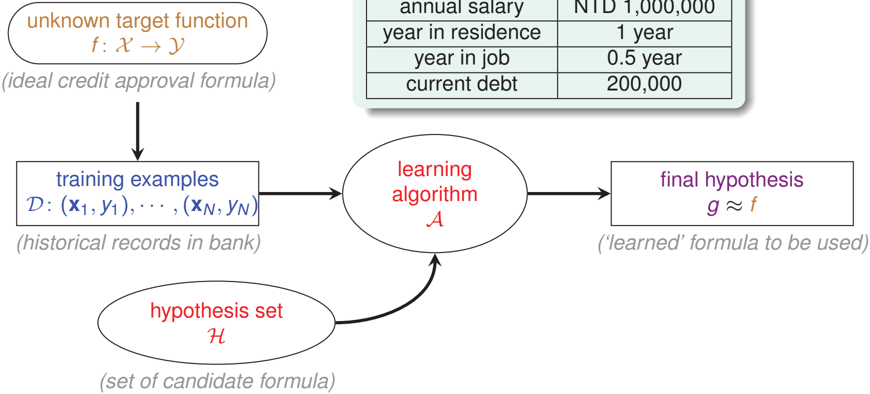
While data mining and machine learning do share a huge overlap, they are arguably not equivalent because of the difference of focus.

Content

回顾信用卡申请案例

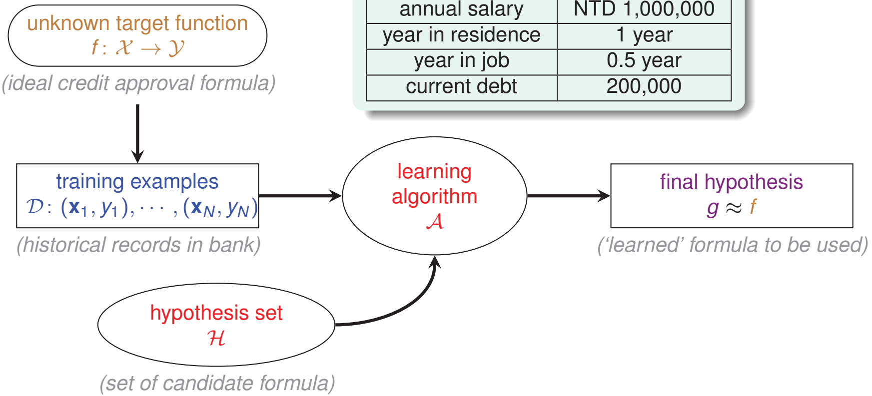
Applicant Information

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



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?

一类简单的假设集合："感知机"

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

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

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

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

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

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

called "perceptron" hypothesis historically

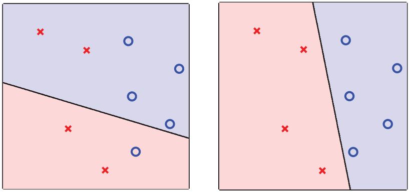
感知机假说的向量形式

$$\begin{aligned} h(\mathbf{x}) &= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right) \\ &= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + \underbrace{(-\text{threshold})}_{w_0} \cdot \underbrace{(+1)}_{x_0}\right) \\ &= \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right) \\ &= \text{sign}\left(\mathbf{w}^T \mathbf{x}\right) \end{aligned}$$

- 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"?

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

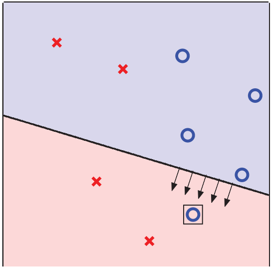


- customer features $\mathbf{x}$: points on the plane (or points in $\mathbb{R}^d$)
- labels y: $\circ$ (+1), $\times$ (-1)
- hypothesis h: **lines** (or hyperplanes in $\mathbb{R}^d$)
  —positive on one side of a line, negative on the other side
- different line classifies customers differently

perceptrons $\Leftrightarrow$ linear (binary) classifiers

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

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



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

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 $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

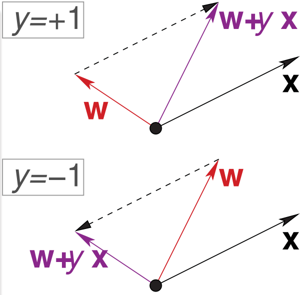
$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

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 $\mathbf{w}_{\text{PLA}}$) as g



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

Cyclic PLA

For $t = 0, 1, \ldots$

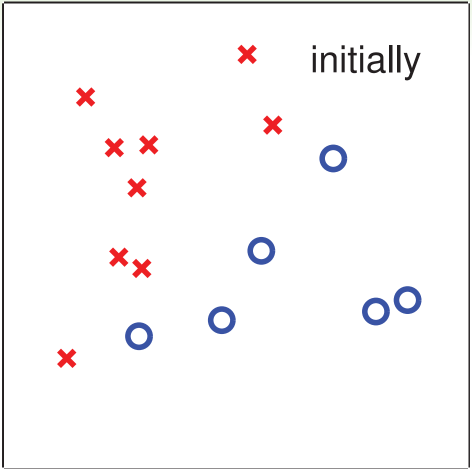
1. find **the next** mistake of $\mathbf{w}_t$ called $(\mathbf{x}_{n(t)}, y_{n(t)})$

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

2. correct the mistake by

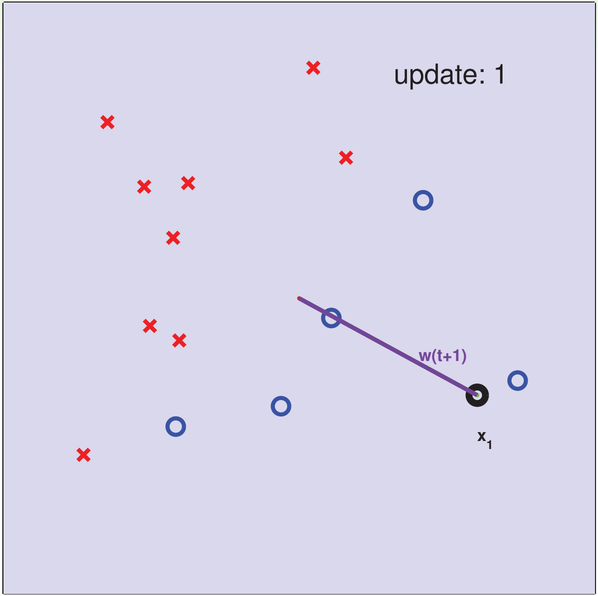
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

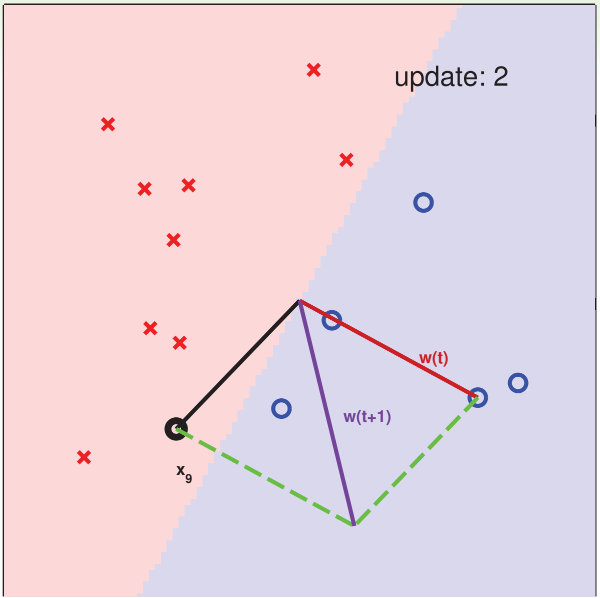
$\ldots$ until **a full cycle of not encountering mistakes**

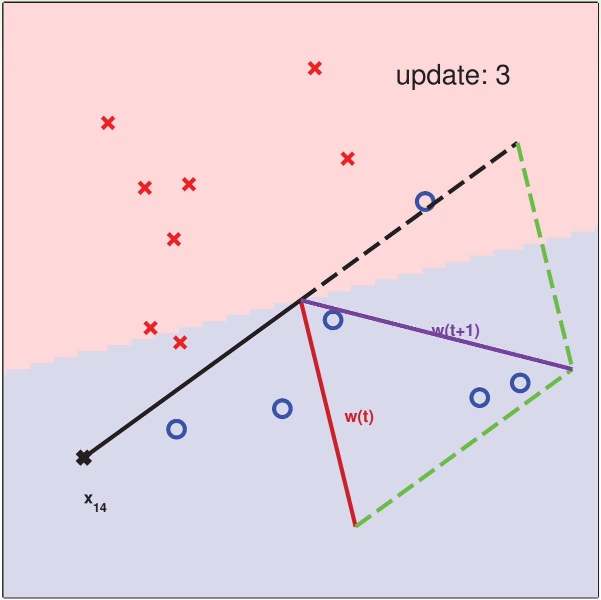


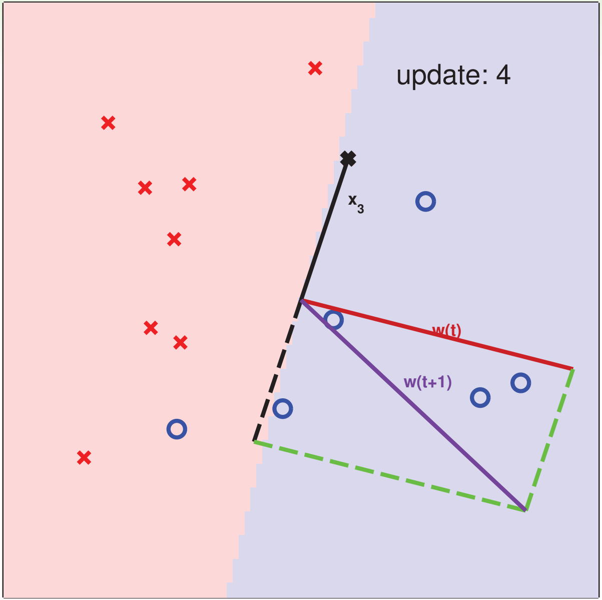
worked like a charm with < 20 lines!!

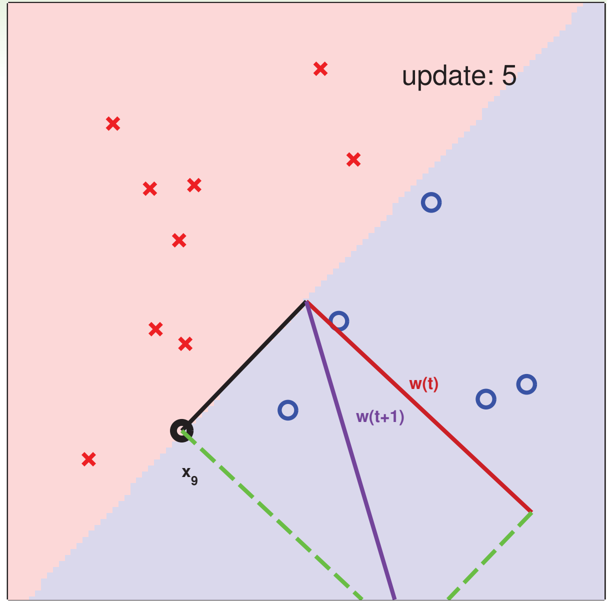
(note: made $\mathbf{x}_i \gg \mathbf{x}_0 = 1$ for visual purpose)

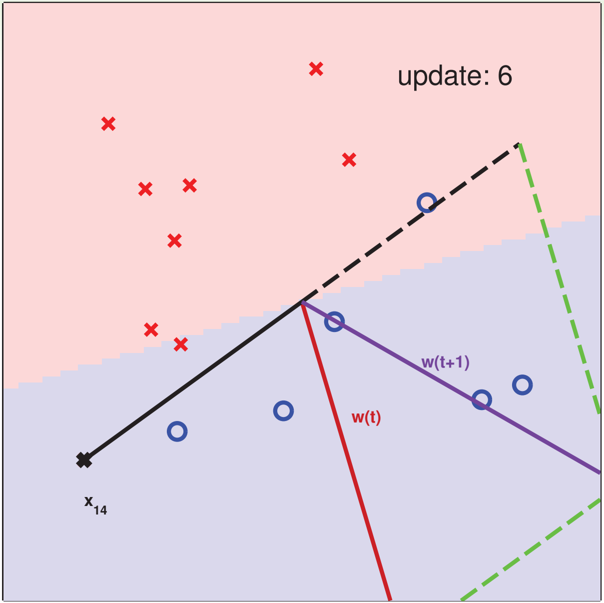

update: 1
w(t+1)
x_1

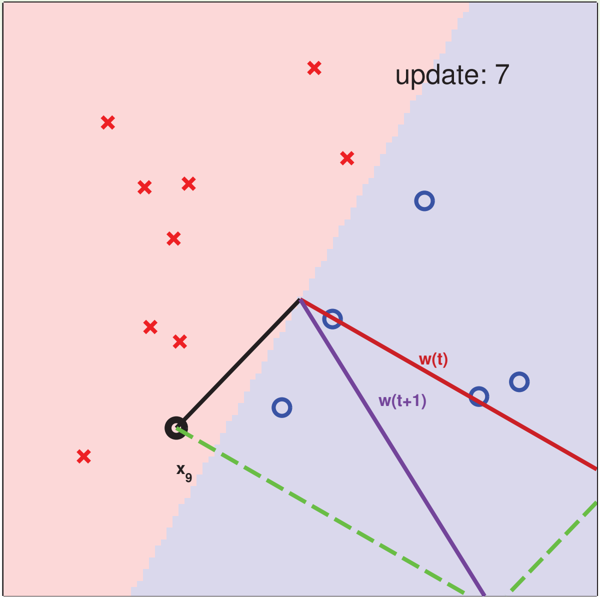

update: 2
w(t)
w(t+1)
x_9


update: 3
w(t+1)
w(t)
x_{14}

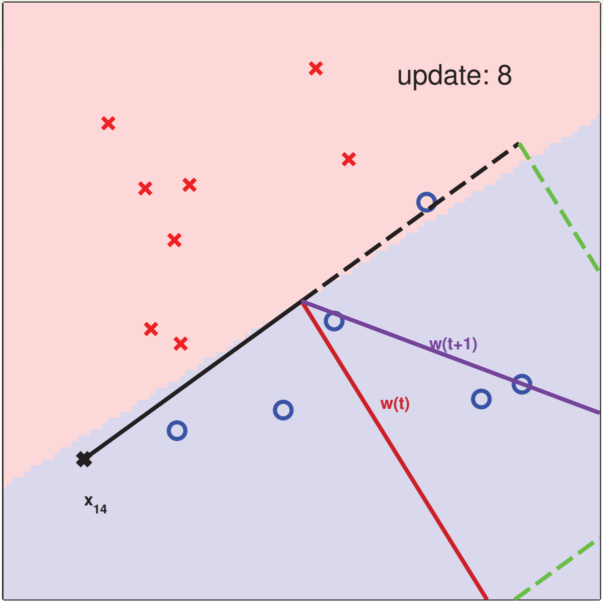

update: 4
x_3
w(t)
w(t+1)

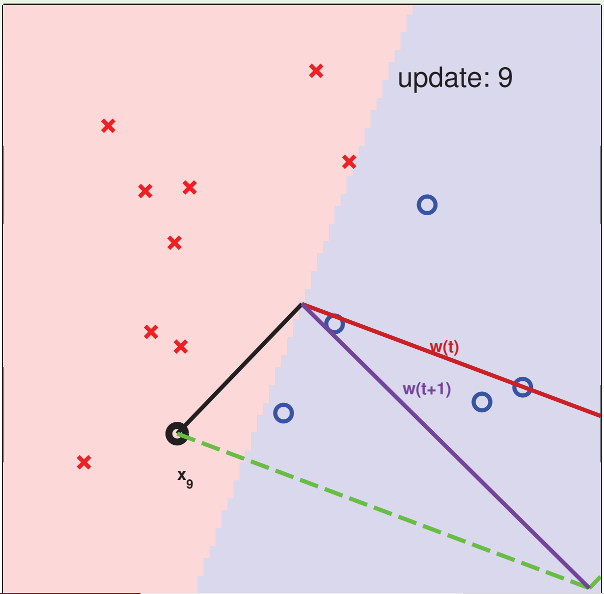

update: 5
w(t)
w(t+1)
x_9


update: 6
w(t+1)
w(t)
x_{14}

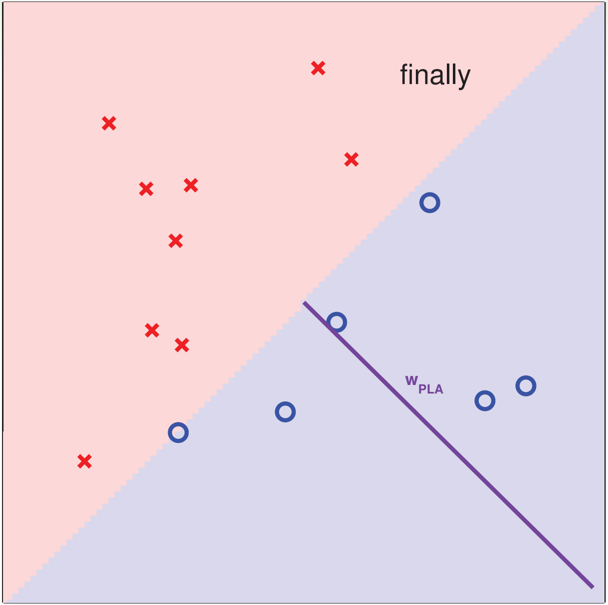

update: 7
w(t)
w(t+1)
x_9

Seeing is Believing




update: 9
w(t)
w(t+1)
x_9

Seeing is Believing



"correct" mistakes on $\mathcal{D}$ until no mistakes

Algorithmic: halt (with no mistake)?

- naive cyclic: ?
- random cyclic: ?
- other variant: ?

Learning: $g \approx f$?

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

Let's try to think about why PLA may work.

Let $n = n(t)$, according to the rule of PLA below, which formula is true?

$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n, \quad \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_n \mathbf{x}_n$$

1. $\mathbf{w}_{t+1}^T \mathbf{x}_n = y_n$
2. $\text{sign}(\mathbf{w}_{t+1}^T \mathbf{x}_n) = y_n$
3. $y_n \mathbf{w}_{t+1}^T \mathbf{x}_n \geq y_n \mathbf{w}_t^T \mathbf{x}_n$
4. $y_n \mathbf{w}_{t+1}^T \mathbf{x}_n < y_n \mathbf{w}_t^T \mathbf{x}_n$

Let's try to think about why PLA may work.

Let $n = n(t)$, according to the rule of PLA below, which formula is true?

$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n, \quad \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_n \mathbf{x}_n$$

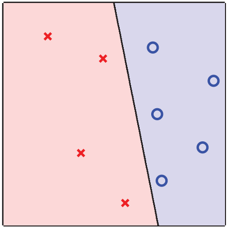
1. $\mathbf{w}_{t+1}^T \mathbf{x}_n = y_n$
2. $\text{sign}(\mathbf{w}_{t+1}^T \mathbf{x}_n) = y_n$
3. $y_n \mathbf{w}_{t+1}^T \mathbf{x}_n \geq y_n \mathbf{w}_t^T \mathbf{x}_n$
4. $y_n \mathbf{w}_{t+1}^T \mathbf{x}_n < y_n \mathbf{w}_t^T \mathbf{x}_n$

Reference Answer: ③

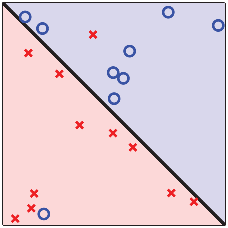
Simply multiply the second part of the rule by $y_n \mathbf{x}_n$. The result shows that the rule somewhat "tries to correct the mistake."

Content

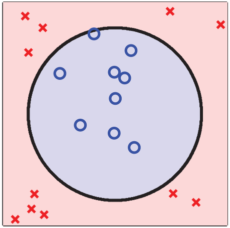
- if PLA halts (i.e. no more mistakes), (necessary condition) $\mathcal{D}$ allows some $\mathbf{w}$ to make no mistake.
- call such $\mathcal{D}$ linear separable



(linear separable)

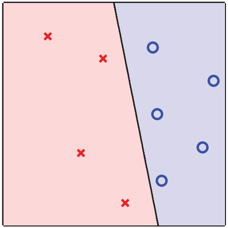


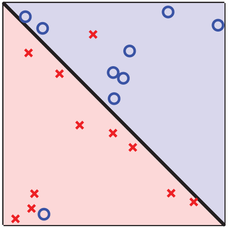
(not linear separable)

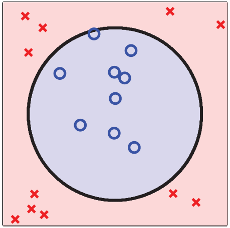


(not linear separable)

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


(linear separable)


(not linear separable)


(not linear separable)

assume linear separable $\mathcal{D}$, does PLA always halt?

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

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

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

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

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

观测一：$\mathbf{w}_t$与$\mathbf{w}_f$愈加靠近

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

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

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

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

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

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

观测二：$\mathbf{w}_t$并没有增长过快

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

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

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

$$
\begin{aligned}
\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\
&= \|\mathbf{w}_t\|^2 + 2y_{n(t)}\mathbf{w}_t^T\mathbf{x}_{n(t)} + \|y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\
&\leq \|\mathbf{w}_t\|^2 + 0 + \|y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\
&\leq \|\mathbf{w}_t\|^2 + \max_n \|y_n\mathbf{x}_n\|^2
\end{aligned}
$$

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

$$
\frac{\mathbf{w}_f^T}{\|\mathbf{w}_f\|}\frac{\mathbf{w}_T}{\|\mathbf{w}_T\|} \geq \sqrt{T} \cdot \text{constant}
$$

Let's upper-bound T, the number of mistakes that PLA 'corrects'.

$$\text{Define } R^2 = \max_n \|\mathbf{x}_n\|^2 \quad \rho = \min_n y_n \frac{\mathbf{w}_f^T}{\|\mathbf{w}_f\|}\mathbf{x}_n$$

We want to show that $T \leq \square$. Express the upper bound $\square$ by the two terms above.

1. R/ρ
2. R^2/ρ^2
3. R/ρ^2
4. ρ^2/R^2

Let's upper-bound T, the number of mistakes that PLA 'corrects'.

$$\text{Define } R^2 = \max_n \|\mathbf{x}_n\|^2 \quad \rho = \min_n y_n \frac{\mathbf{w}_f^T}{\|\mathbf{w}_f\|}\mathbf{x}_n$$

We want to show that $T \leq \square$. Express the upper bound $\square$ by the two terms above.

1. R/ρ
2. R^2/ρ^2
3. R/ρ^2
4. ρ^2/R^2

Reference Answer: ②

$T \leq R^2/\rho^2$.

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

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

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
- not fully sure how long halting takes (ρ depends on $\mathbf{w}_f$)
  - — though practically fast

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

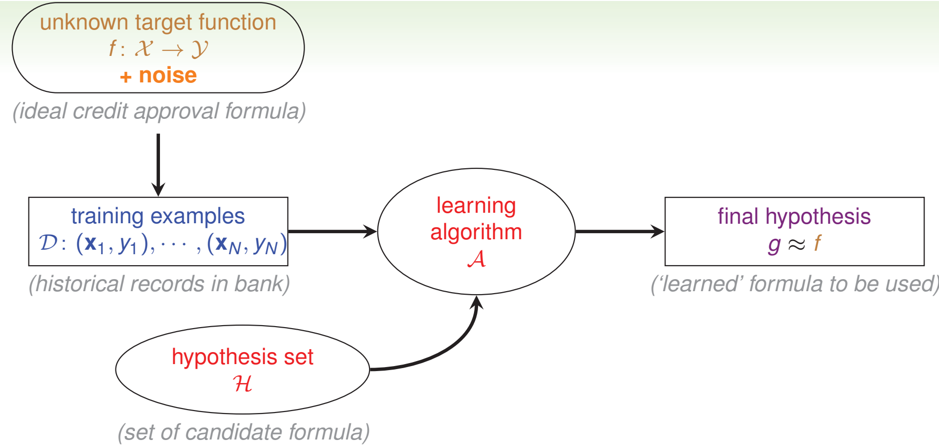
Cons

- “assumes” linear separable $\mathcal{D}$ to halt
  - — property unknown in advance
- not fully sure how long halting takes (ρ depends on $\mathbf{w}_f$)
  - — though practically fast

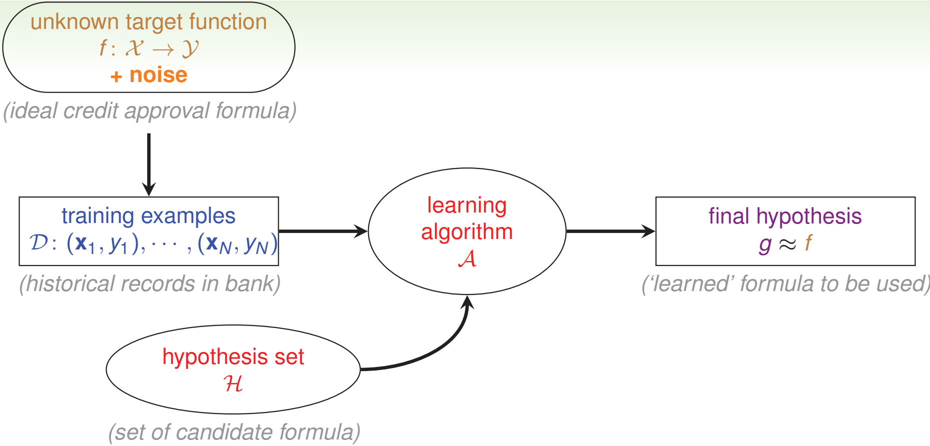
what if $\mathcal{D}$ not linear separable?

Content

噪声数据学习

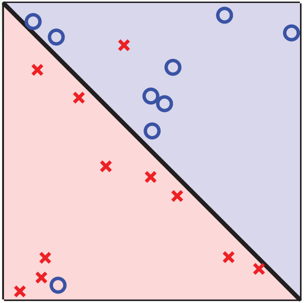

unknown target function
$f: \mathcal{X} \to \mathcal{Y}$
+ noise
(ideal credit approval formula)
training examples
$\mathcal{D}: (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$
(historical records in bank)
learning algorithm
$\mathcal{A}$
final hypothesis
$g \approx f$
('learned' formula to be used)
hypothesis set
$\mathcal{H}$
(set of candidate formula)

噪声数据学习



how to at least get $g \approx f$ on noisy data?

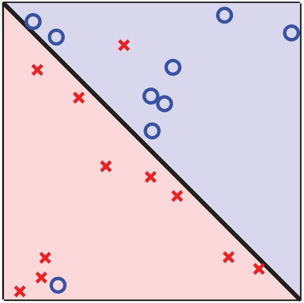
容忍噪声的线性分类器



- assume 'little' noise: $y_n = f(\mathbf{x}_n)$ **usually**
- if so, $g \approx f$ on $\mathcal{D} \Leftrightarrow y_n = g(\mathbf{x}_n)$ **usually**
- how about

$$\mathbf{w}_g \leftarrow \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} [\![y_n \neq \operatorname{sign}(\mathbf{w}^T \mathbf{x}_n)]\!]$$

—**NP-hard to solve, unfortunately**



- assume 'little' noise: $y_n = f(\mathbf{x}_n)$ **usually**
- if so, $g \approx f$ on $\mathcal{D} \Leftrightarrow y_n = g(\mathbf{x}_n)$ **usually**
- how about

$$\mathbf{w}_g \leftarrow \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} [\![y_n \neq \operatorname{sign}(\mathbf{w}^T \mathbf{x}_n)]\!]$$

—**NP-hard to solve, unfortunately**

can we modify PLA to get an approximately good g?

modify PLA algorithm by keeping best weights in pocket

initialize pocket weights $\hat{\mathbf{w}}$

For $t = 0, 1, \cdots$

1. find a (random) mistake of $\mathbf{w}_t$ called $(\mathbf{x}_{n(t)}, y_{n(t)})$
2. (try to) correct the mistake by

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

3. **if $\mathbf{w}_{t+1}$ makes fewer mistakes than $\hat{\mathbf{w}}$, replace $\hat{\mathbf{w}}$ by $\mathbf{w}_{t+1}$**

...until **enough iterations**

return **$\hat{\mathbf{w}}$ (called $\mathbf{w}_{\text{POCKET}}$) as g**

modify PLA algorithm by keeping best weights in pocket

initialize pocket weights $\hat{\mathbf{w}}$

For $t = 0, 1, \cdots$

1. find a (random) mistake of $\mathbf{w}_t$ called $(\mathbf{x}_{n(t)}, y_{n(t)})$
2. (try to) correct the mistake by

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

3. **if $\mathbf{w}_{t+1}$ makes fewer mistakes than $\hat{\mathbf{w}}$, replace $\hat{\mathbf{w}}$ by $\mathbf{w}_{t+1}$**

...until **enough iterations**

return **$\hat{\mathbf{w}}$ (called $\mathbf{w}_{\text{POCKET}}$) as g**

a simple modification of PLA to find (somewhat) “best” weights

Should we use pocket or PLA?

Since we do not know whether $\mathcal{D}$ is linear separable in advance, we may decide to just go with pocket instead of PLA. If $\mathcal{D}$ is actually linear separable, what's the difference between the two?

1. pocket on $\mathcal{D}$ is slower than PLA
2. pocket on $\mathcal{D}$ is faster than PLA
3. pocket on $\mathcal{D}$ returns a better g in approximating f than PLA
4. pocket on $\mathcal{D}$ returns a worse g in approximating f than PLA

Should we use pocket or PLA?

Since we do not know whether $\mathcal{D}$ is linear separable in advance, we may decide to just go with pocket instead of PLA. If $\mathcal{D}$ is actually linear separable, what's the difference between the two?

1. pocket on $\mathcal{D}$ is slower than PLA
2. pocket on $\mathcal{D}$ is faster than PLA
3. pocket on $\mathcal{D}$ returns a better g in approximating f than PLA
4. pocket on $\mathcal{D}$ returns a worse g in approximating f than PLA

Reference Answer: ①

Because pocket needs to check whether $\mathbf{w}_{t+1}$ is better than $\hat{\mathbf{w}}$ at each iteration, it is slower than PLA. On linear separable $\mathcal{D}$, $\mathbf{w}_{\text{POCKET}}$ is the same as $\mathbf{w}_{\text{PLA}}$, both making no mistakes.

Lecture 2: Learning to Answer Yes/No

- 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
- Non-Separable Data
  hold somewhat 'best' weights in pocket

Lecture 2: Learning to Answer Yes/No

- 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
- Non-Separable Data
  hold somewhat 'best' weights in pocket

Homework 1: Implement PLA (Pocket) algorithm.

Lecture 2: Learning to Answer Yes/No

- 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
- Non-Separable Data
  hold somewhat 'best' weights in pocket

Homework 1: Implement PLA (Pocket) algorithm.

Next: Support Vector Machines and Kernel Methods.