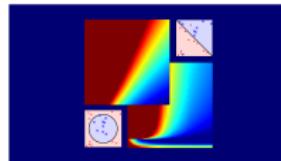


Machine Learning Foundations (機器學習基石)



Lecture 1: The Learning Problem

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)



Course Design (1/2)

Machine Learning: a mixture of theoretical and practical tools

- theory oriented
 - derive everything **deeply** for solid understanding
 - less interesting to general audience
- techniques **s** oriented
 - flash over the sexiest techniques **broadly** for shiny coverage
 - too many techniques, hard to choose, hard to use properly

our approach: **foundation oriented**

Course Design (2/2)

Foundation Oriented ML Course

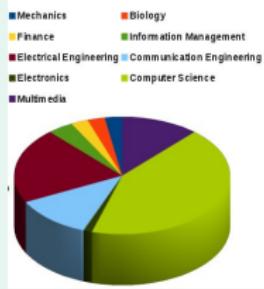
- mixture of philosophical illustrations, key theory, core techniques, usage in practice, and hopefully jokes :-)
 - what **every machine learning user** should know
- story-like:
 - **When** Can Machines Learn? (illustrative + technical)
 - **Why** Can Machines Learn? (theoretical + illustrative)
 - **How** Can Machines Learn? (technical + practical)
 - How Can Machines Learn **Better**? (practical + theoretical)

allows students to **learn ‘future/untaught’ techniques or study deeper theory easily**

Course History

NTU Version

- 15-17 weeks (2+ hours)
- highly-praised with English and blackboard teaching



Coursera Version

- 8 weeks of ‘foundation’ (**this course**) + 7 weeks of ‘techniques’ (coming course)
- **Mandarin teaching** to reach more audience in need
- **slides teaching** improved with Coursera’s quiz and homework mechanisms

goal: **try** making Coursera version even better than NTU version

Fun Time

Which of the following description of this course is true?

- ① the course will be taught in Taiwanese
- ② the course will tell me the techniques that create the android Lieutenant Commander Data in Star Trek
- ③ the course will be 15 weeks long
- ④ the course will be story-like

Fun Time

Which of the following description of this course is true?

- ① the course will be taught in Taiwanese
- ② the course will tell me the techniques that create the android Lieutenant Commander Data in Star Trek
- ③ the course will be 15 weeks long
- ④ the course will be story-like

Reference Answer: ④

- ① no, my Taiwanese is unfortunately not good enough for teaching (yet)
- ② no, although what we teach may serve as foundations of those (future) techniques
- ③ no, unless you choose to join the next course
- ④ yes, **let's begin the story**

Roadmap

① When Can Machines Learn?

Lecture 1: The Learning Problem

- Course Introduction
- What is Machine Learning
- Applications of Machine Learning
- Components of Machine Learning
- Machine Learning and Other Fields

② Why Can Machines Learn?

③ How Can Machines Learn?

④ How Can Machines Learn Better?

From Learning to Machine Learning

learning: acquiring **skill**

with experience accumulated from **observations**



machine learning: acquiring **skill**

with experience accumulated/**computed** from **data**



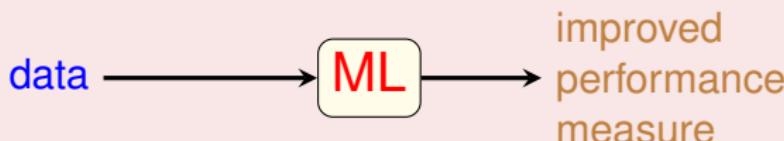
What is **skill**?

A More Concrete Definition

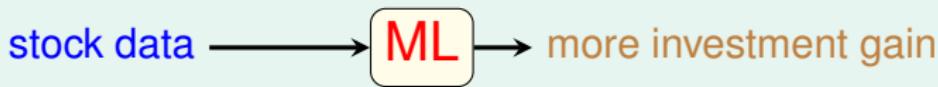
skill

↔ 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

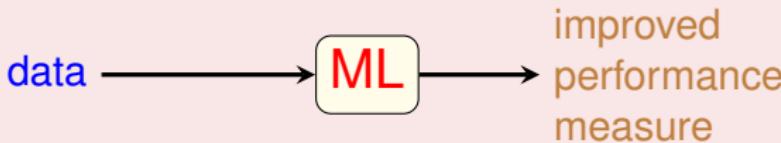
Some Use Scenarios

- 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

Fun Time

Which of the following is best suited for machine learning?

- ① predicting whether the next cry of the baby girl happens at an even-numbered minute or not
- ② determining whether a given graph contains a cycle
- ③ deciding whether to approve credit card to some customer
- ④ guessing whether the earth will be destroyed by the misuse of nuclear power in the next ten years

Fun Time

Which of the following is best suited for machine learning?

- ① predicting whether the next cry of the baby girl happens at an even-numbered minute or not
- ② determining whether a given graph contains a cycle
- ③ deciding whether to approve credit card to some customer
- ④ guessing whether the earth will be destroyed by the misuse of nuclear power in the next ten years

Reference Answer: ③

- ① no pattern
- ② programmable definition
- ③ pattern: customer behavior;
definition: not easily programmable;
data: history of bank operation
- ④ arguably no (or not enough) data yet

Daily Needs: Food, Clothing, Housing, Transportation



1 Food (Sadilek et al., 2013)

- **data**: Twitter data (words + location)
- **skill**: tell food poisoning likeliness of restaurant properly

2 Clothing (Abu-Mostafa, 2012)

- **data**: sales figures + client surveys
- **skill**: give good fashion recommendations to clients

3 Housing (Tsanas and Xifara, 2012)

- **data**: characteristics of buildings and their energy load
- **skill**: predict energy load of other buildings closely

4 Transportation (Stallkamp et al., 2012)

- **data**: some traffic sign images and meanings
- **skill**: recognize traffic signs accurately

ML is everywhere!

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

Entertainment: Recommender System (1/2)



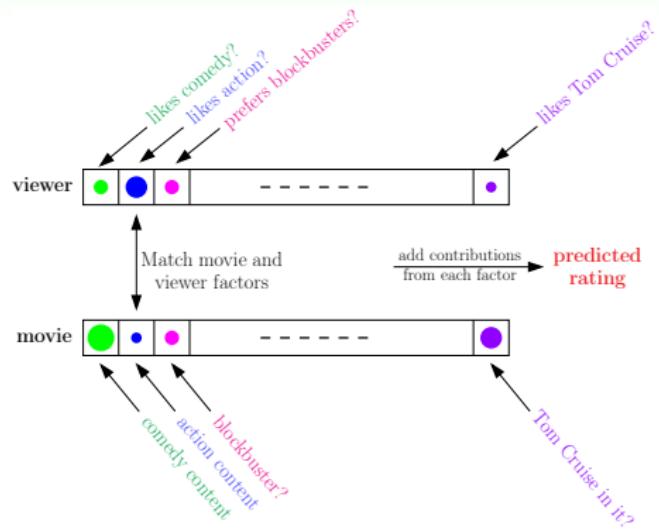
- **data**: how many users have rated some movies
- **skill**: predict how a user would rate an unrated movie

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 → 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?**

Entertainment: Recommender System (2/2)



A Possible ML Solution

- pattern:
 $\text{rating} \leftarrow \text{viewer/movie factors}$
- learning:
known rating
→ learned factors
→ unknown rating prediction

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

Fun Time

Which of the following field cannot use machine learning?

- ① Finance
- ② Medicine
- ③ Law
- ④ none of the above

Fun Time

Which of the following field cannot use machine learning?

- ① Finance
- ② Medicine
- ③ Law
- ④ none of the above

Reference Answer: ④

- ① predict stock price from data
 - ② predict medicine effect from data
 - ③ summarize legal documents from data
 - ④ :-)
- Welcome to study this hot topic!**

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?’

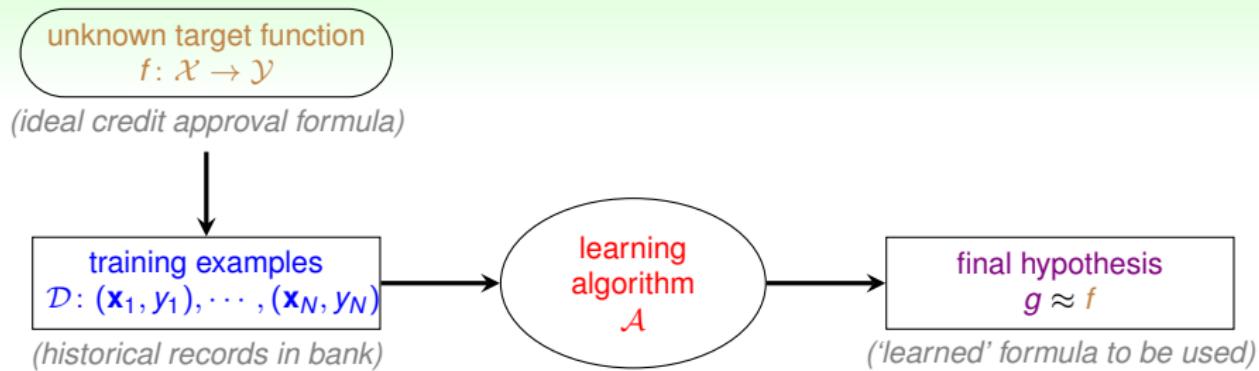
Formalize the Learning Problem

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:
 $f: \mathcal{X} \rightarrow \mathcal{Y}$ (ideal credit approval formula)
- data \Leftrightarrow training examples: $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$
(historical records in bank)
- hypothesis \Leftrightarrow skill with hopefully good performance:
 $g: \mathcal{X} \rightarrow \mathcal{Y}$ ('learned' formula to be used)

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

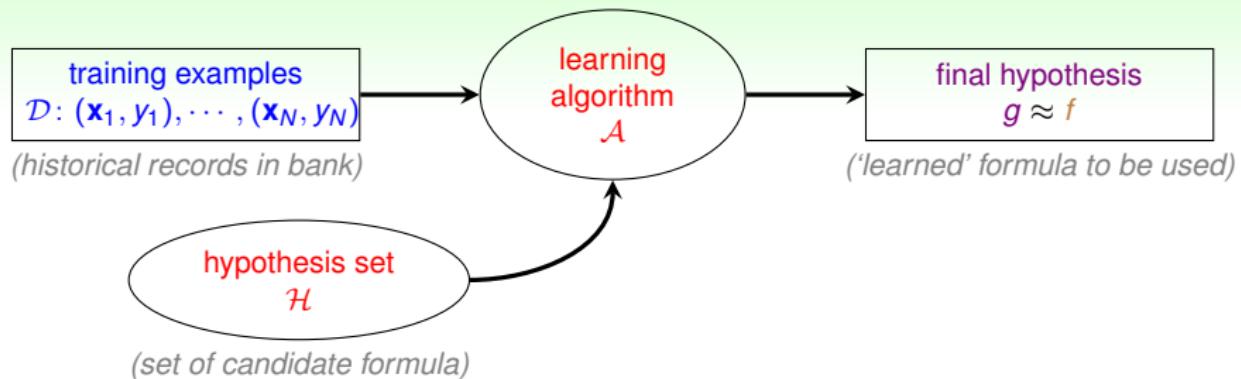
Learning Flow for Credit Approval



- 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?

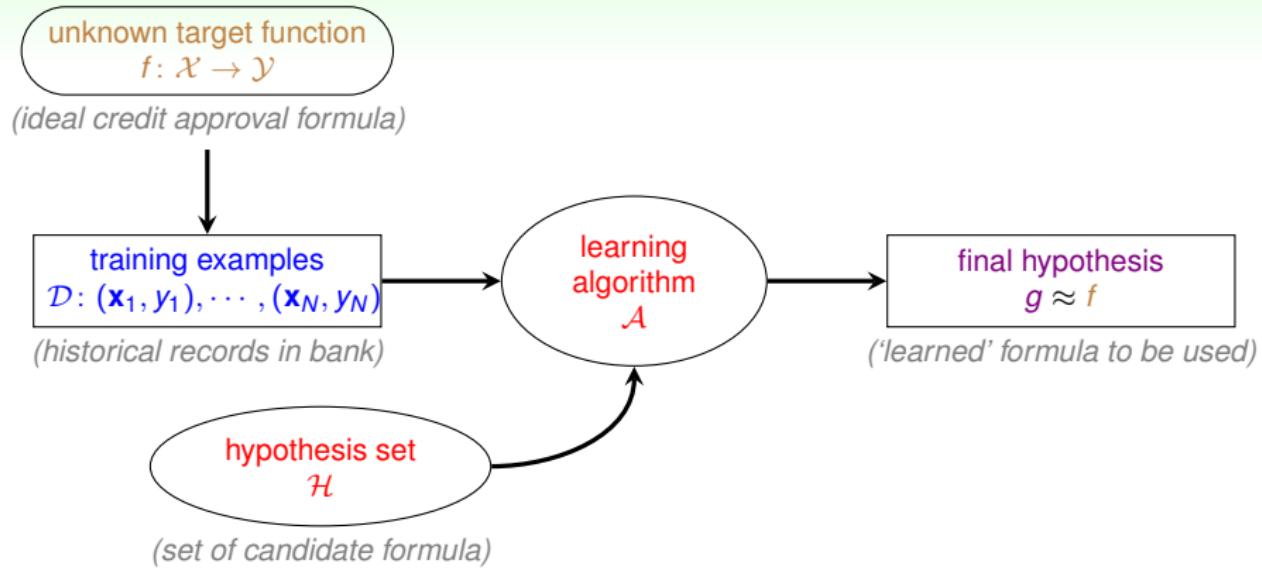
The Learning Model



- 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 ≤ 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



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

Fun Time

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

$$\mathcal{S}_1 = [0, 100]$$

\mathcal{S}_2 = all possible (userid, songid) pairs

\mathcal{S}_3 = all formula that ‘multiplies’ user factors & song factors, indexed by all possible combinations of such factors

\mathcal{S}_4 = 1,000,000 pairs of ((userid, songid), rating)

- 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 $\mathcal{S}_1 = \mathcal{X}, \mathcal{S}_2 = \mathcal{D}, \mathcal{S}_3 = \mathcal{Y}, \mathcal{S}_4 = \mathcal{H}$

Fun Time

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

$$\mathcal{S}_1 = [0, 100]$$

\mathcal{S}_2 = all possible (userid, songid) pairs

\mathcal{S}_3 = all formula that ‘multiplies’ user factors & song factors, indexed by all possible combinations of such factors

\mathcal{S}_4 = 1,000,000 pairs of ((userid, songid), rating)

- 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 $\mathcal{S}_1 = \mathcal{X}, \mathcal{S}_2 = \mathcal{D}, \mathcal{S}_3 = \mathcal{Y}, \mathcal{S}_4 = \mathcal{H}$

Reference Answer: ②

$$\mathcal{S}_4 \xrightarrow{\text{A on } \mathcal{S}_3} (g: \mathcal{S}_2 \rightarrow \mathcal{S}_1)$$

Machine Learning and Data Mining

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 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 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

Fun Time

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.

Summary

① When Can Machines Learn?

Lecture 1: The Learning Problem

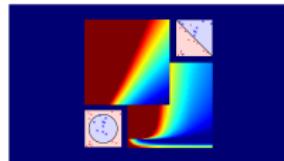
- Course Introduction
foundation oriented and story-like
- What is Machine Learning
use data to approximate target
- Applications of Machine Learning
almost everywhere
- Components of Machine Learning
 \mathcal{A} takes \mathcal{D} and \mathcal{H} to get g
- Machine Learning and Other Fields
related to DM, AI and Stats

- next: a simple and yet useful learning model (\mathcal{H} and \mathcal{A})

- 2 Why Can Machines Learn?
- 3 How Can Machines Learn?
- 4 How Can Machines Learn Better?

Machine Learning Foundations

(機器學習基石)



Lecture 2: Learning to Answer Yes/No

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: The Learning Problem

\mathcal{A} takes \mathcal{D} and \mathcal{H} to get g

Lecture 2: Learning to Answer Yes/No

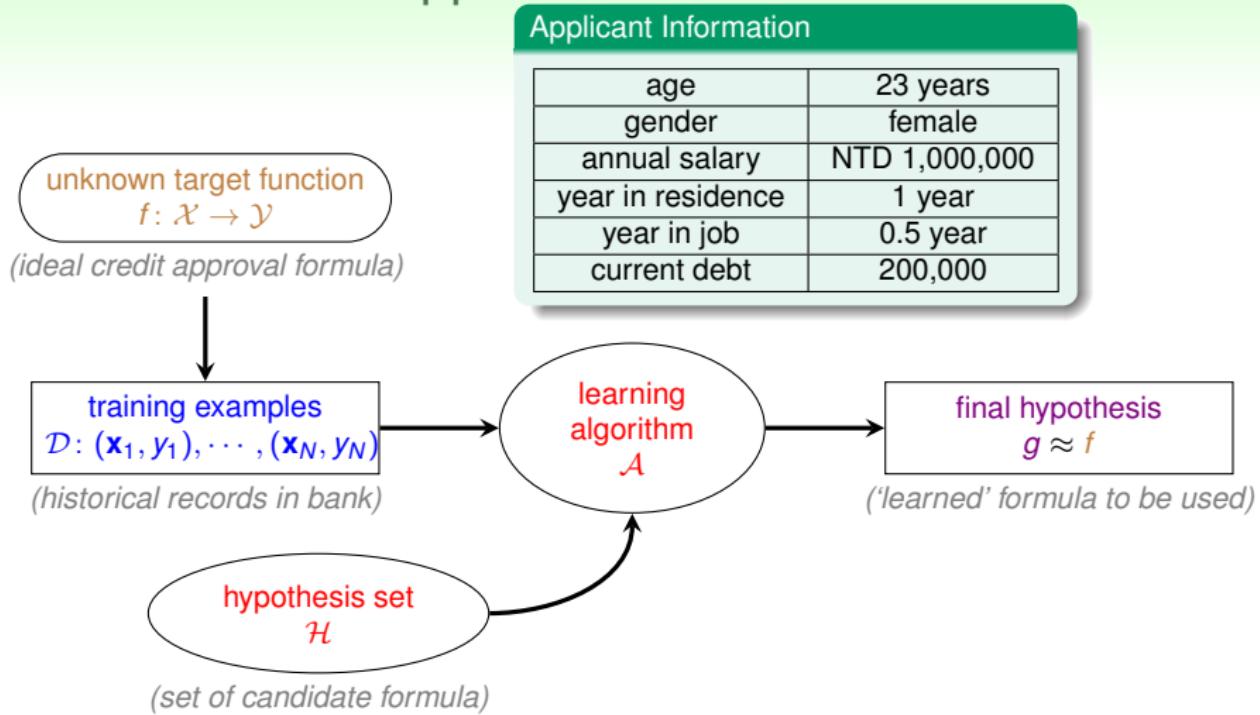
- Perceptron Hypothesis Set
- Perceptron Learning Algorithm (PLA)
- Guarantee of PLA
- Non-Separable Data

2 Why Can Machines Learn?

3 How Can Machines Learn?

4 How Can Machines Learn Better?

Credit Approval Problem Revisited



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

- For $\mathbf{x} = (x_1, x_2, \dots, x_d)$ ‘features of customer’, compute a weighted ‘score’ and

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

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

- $\mathcal{Y}: \{+1(\text{good}), -1(\text{bad})\}$, 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

Vector Form of Perceptron Hypothesis

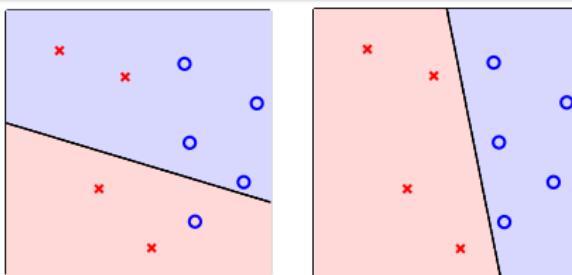
$$\begin{aligned}
 h(\mathbf{x}) &= \text{sign} \left(\left(\sum_{i=1}^d w_i x_i \right) - \text{threshold} \right) \\
 &= \text{sign} \left(\left(\sum_{i=1}^d w_i x_i \right) + \underbrace{(-\text{threshold})}_{w_0} \cdot \underbrace{(+1)}_{x_0} \right) \\
 &= \text{sign} \left(\sum_{i=0}^d w_i x_i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

- each ‘tall’ \mathbf{w} represents a hypothesis h & is multiplied with ‘tall’ \mathbf{x} —**will use tall versions to simplify notation**

what do perceptrons h ‘look like’?

Perceptrons in \mathbb{R}^2

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



- customer features \mathbf{x} : points on the plane (or points in \mathbb{R}^d)
- labels y :
 - (+1), × (-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**

Fun Time

Consider using a perceptron to detect spam messages.

Assume that each email is represented by the frequency of keyword occurrence, and output +1 indicates a spam. Which keywords below shall have large positive weights in a **good perceptron** for the task?

- ① coffee, tea, hamburger, steak
- ② free, drug, fantastic, deal
- ③ machine, learning, statistics, textbook
- ④ national, Taiwan, university, coursera

Fun Time

Consider using a perceptron to detect spam messages.

Assume that each email is represented by the frequency of keyword occurrence, and output +1 indicates a spam. Which keywords below shall have large positive weights in a **good perceptron** for the task?

- ① coffee, tea, hamburger, steak
- ② free, drug, fantastic, deal
- ③ machine, learning, statistics, textbook
- ④ national, Taiwan, university, coursera

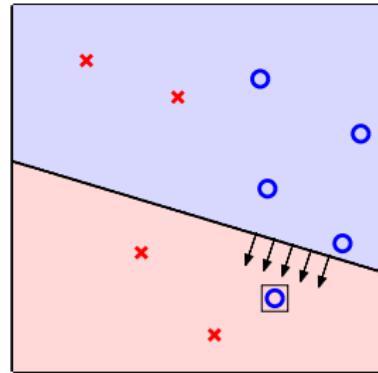
Reference Answer: ②

The occurrence of keywords with positive weights increase the 'spam score', and hence those keywords should often appear in spams.

Select g from \mathcal{H}

\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

Perceptron Learning Algorithm

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

For $t = 0, 1, \dots$

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

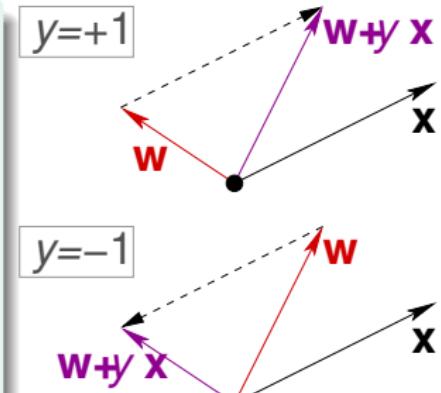
$$\text{sign}(\mathbf{w}_t^T \mathbf{x}_{n(t)}) \neq y_{n(t)}$$

- 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}

Cyclic PLA

For $t = 0, 1, \dots$

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

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

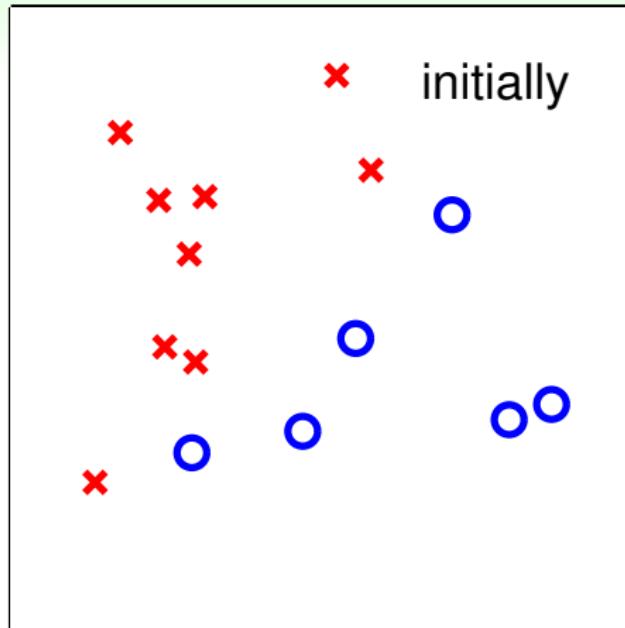
- ② correct the mistake by

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

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

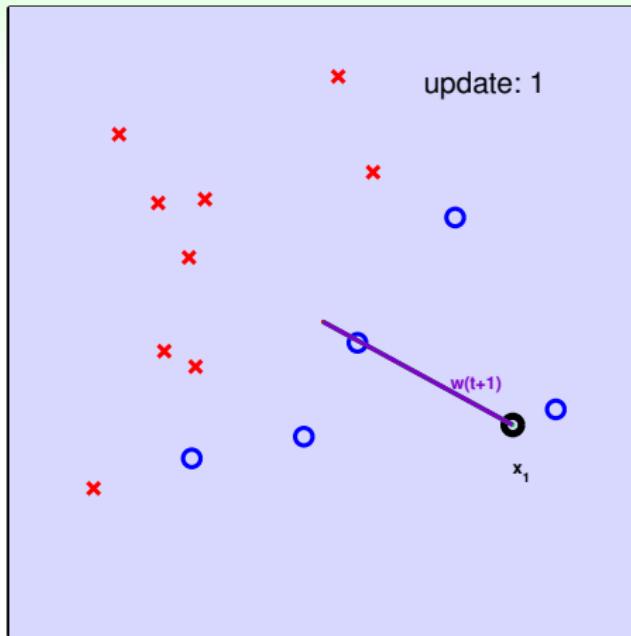
next can follow naïve cycle $(1, \dots, N)$
or precomputed random cycle

Seeing is Believing



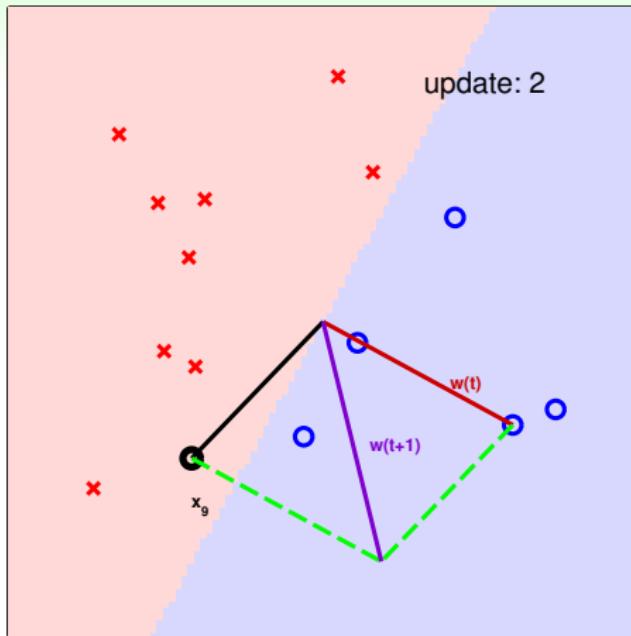
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



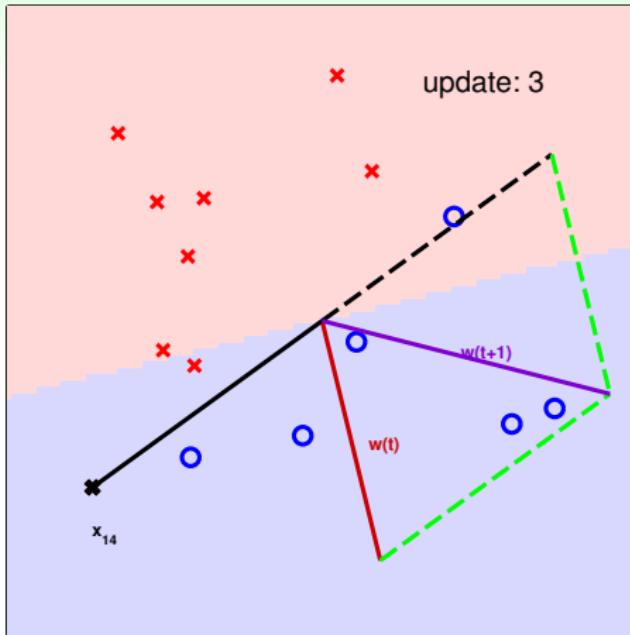
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



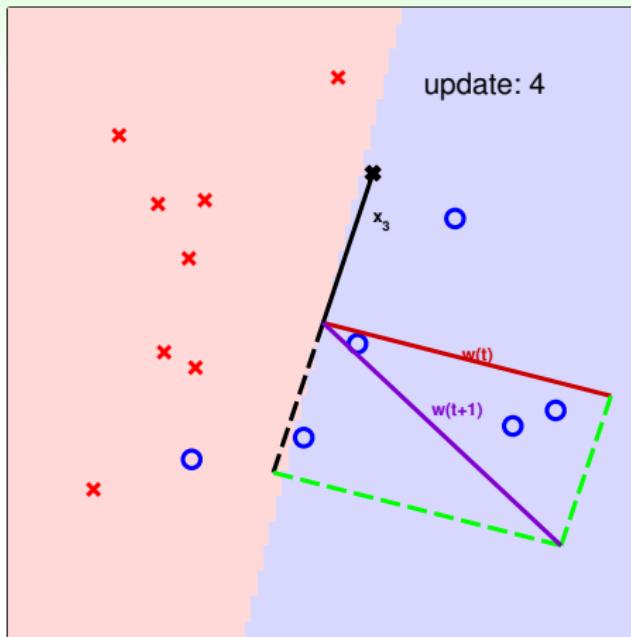
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



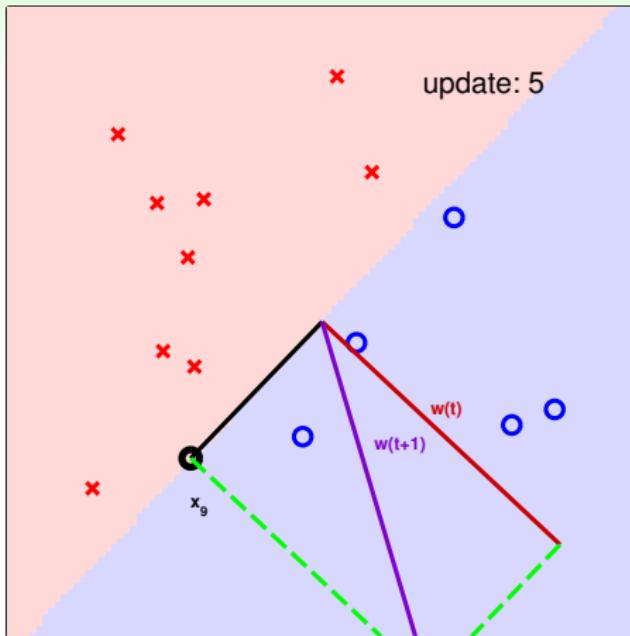
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



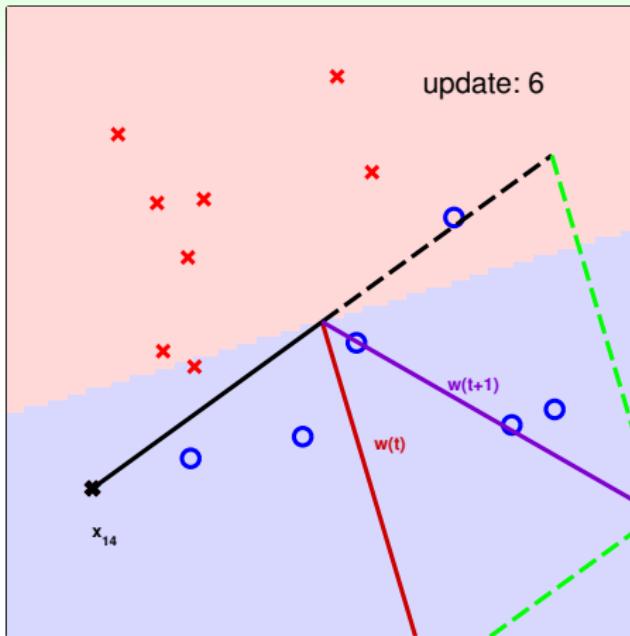
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



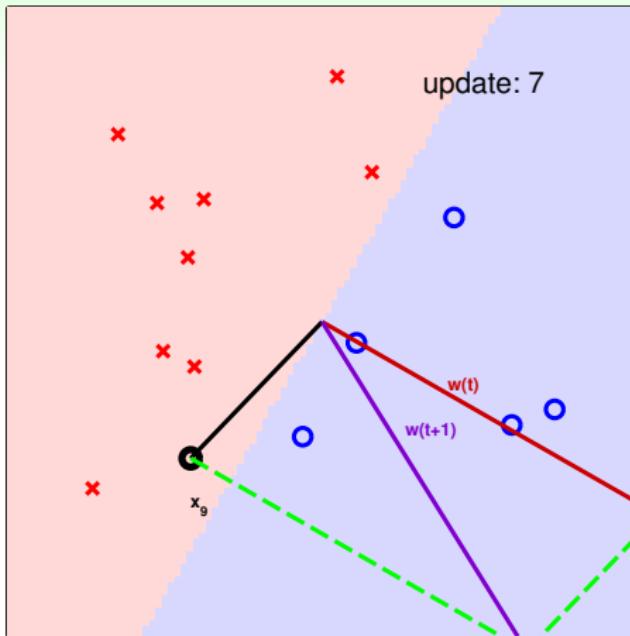
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



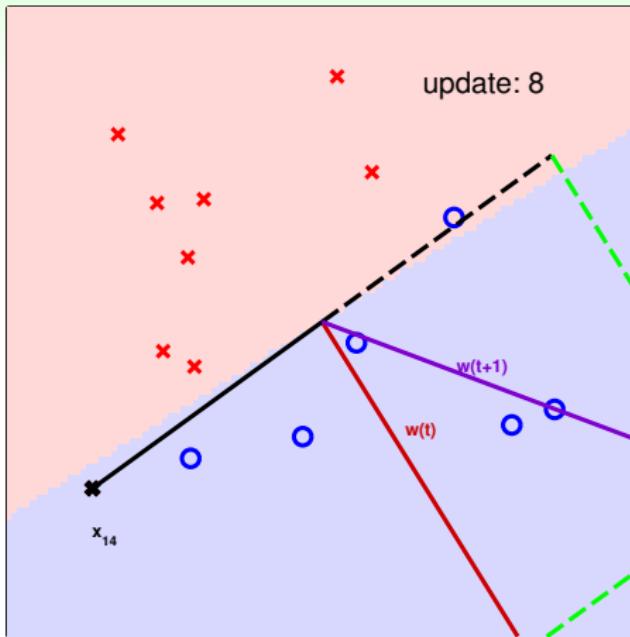
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



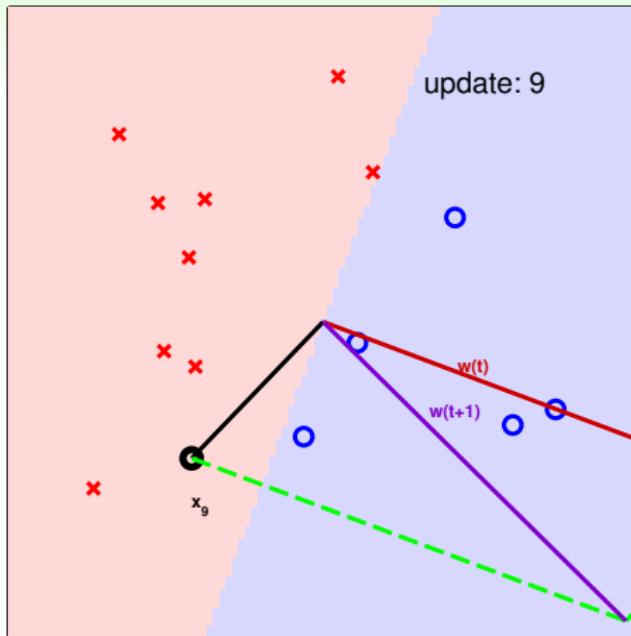
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



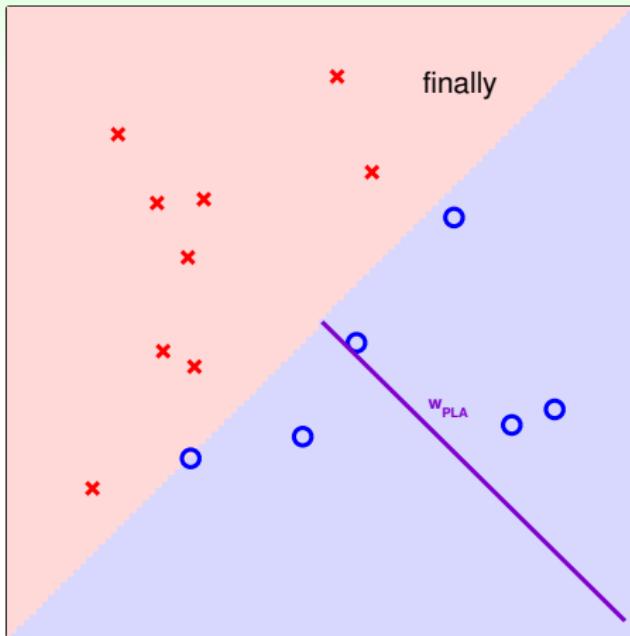
worked like a charm with < 20 lines!!
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



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

Seeing is Believing



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} : ??
- if not halting: ??

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

Fun Time

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?

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

- 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 $y_n \mathbf{w}_{t+1}^T \mathbf{x}_n < y_n \mathbf{w}_t^T \mathbf{x}_n$

Fun Time

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?

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

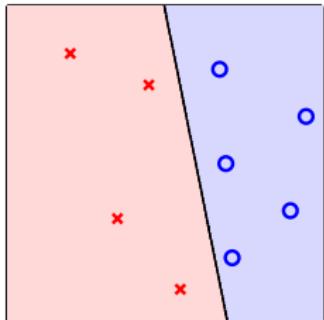
- ① $\mathbf{w}_{t+1}^T \mathbf{x}_n = y_n$
- ② $\text{sign}(\mathbf{w}_{t+1}^T \mathbf{x}_n) = y_n$
- ③ $y_n \mathbf{w}_{t+1}^T \mathbf{x}_n \geq y_n \mathbf{w}_t^T \mathbf{x}_n$
- ④ $y_n \mathbf{w}_{t+1}^T \mathbf{x}_n < y_n \mathbf{w}_t^T \mathbf{x}_n$

Reference Answer: ③

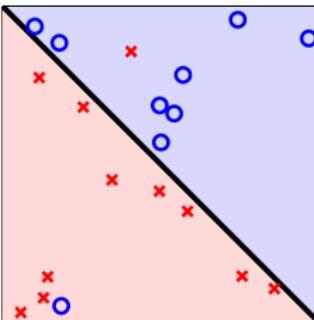
Simply multiply the second part of the rule by $y_n \mathbf{x}_n$. The result shows that **the rule somewhat ‘tries to correct the mistake.’**

Linear Separability

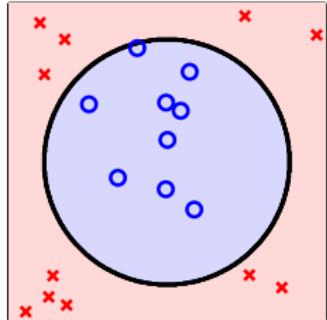
- if PLA halts (i.e. no more mistakes),
(necessary condition) \mathcal{D} allows some w to make no mistake
- call such \mathcal{D} **linear separable**



(linear separable)



(not linear separable)



(not linear separable)

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

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

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

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

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

\mathbf{w}_t appears more aligned with \mathbf{w}_f after update
(really?)

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

\mathbf{w}_t changed only when mistake

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

- 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,

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

Fun Time

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

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

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

Fun Time

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

The maximum value of $\frac{\mathbf{w}_f^T}{\|\mathbf{w}_f\|} \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|}$ is 1. Since T mistake corrections **increase the inner product by $\sqrt{T} \cdot \text{constant}$** , the maximum number of corrected mistakes is $1/\text{constant}^2$.

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

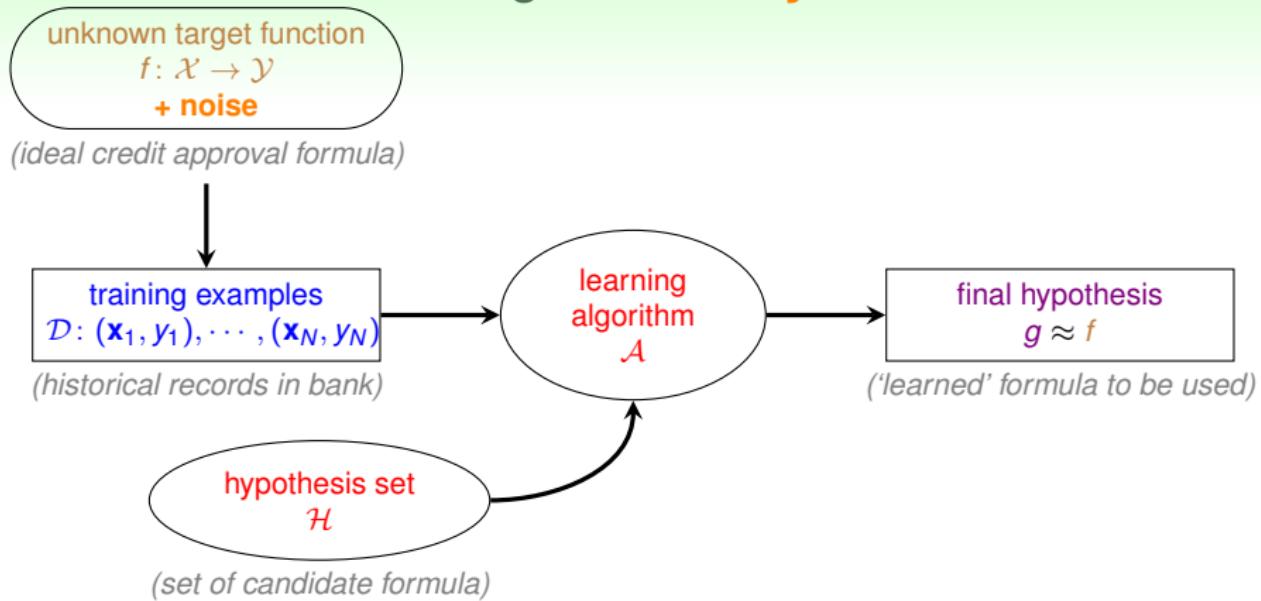
simple to implement, fast, works in any dimension d

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

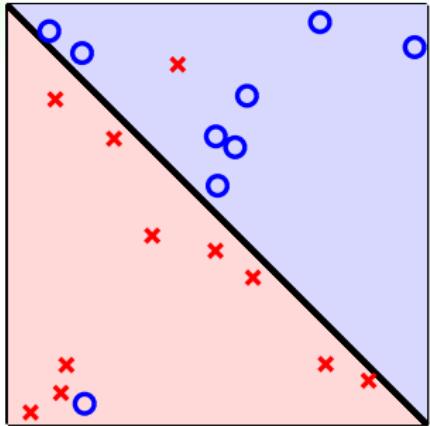
what if \mathcal{D} not linear separable?

Learning with **Noisy** Data



how to at least get $g \approx f$ on **noisy** \mathcal{D} ?

Line with Noise Tolerance



- 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

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

—NP-hard to solve, unfortunately

can we modify PLA to get
an ‘approximately good’ g ?

Pocket Algorithm

modify PLA algorithm (black lines) by **keeping best weights in pocket**

initialize pocket weights \hat{w}

For $t = 0, 1, \dots$

- 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{w} , replace \hat{w} by \mathbf{w}_{t+1}

...until enough iterations

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

a simple modification of PLA to find
(somewhat) ‘best’ weights

Fun Time

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

Fun Time

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

Summary

1 When Can Machines Learn?

Lecture 1: The Learning Problem

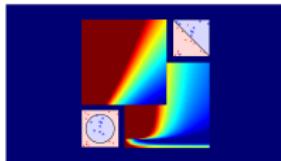
Lecture 2: Learning to Answer Yes/No

- 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

- next: the zoo of learning problems

- 2 Why Can Machines Learn?
- 3 How Can Machines Learn?
- 4 How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 3: Types of 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 2: Learning to Answer Yes/No

PLA \mathcal{A} takes linear separable \mathcal{D} and perceptrons \mathcal{H} to get hypothesis g

Lecture 3: Types of Learning

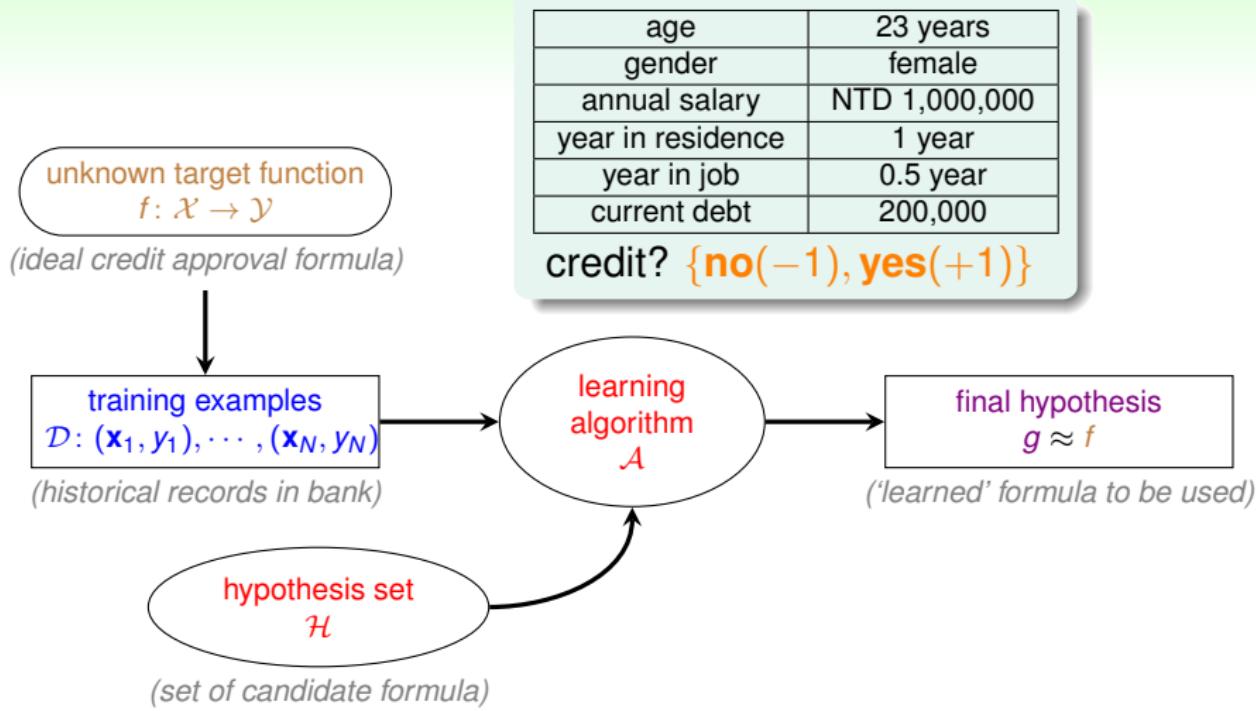
- Learning with Different Output Space \mathcal{Y}
- Learning with Different Data Label y_n
- Learning with Different Protocol $f \Rightarrow (\mathbf{x}_n, y_n)$
- Learning with Different Input Space \mathcal{X}

② Why Can Machines Learn?

③ How Can Machines Learn?

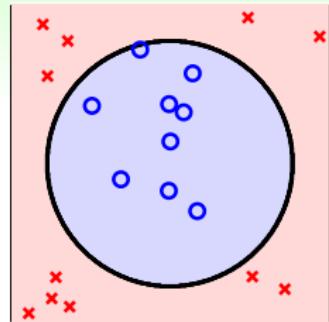
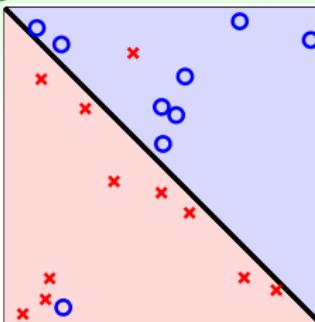
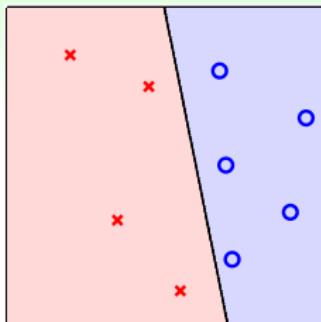
④ How Can Machines Learn Better?

Credit Approval Problem Revisited



$\mathcal{Y} = \{-1, +1\}$: **binary classification**

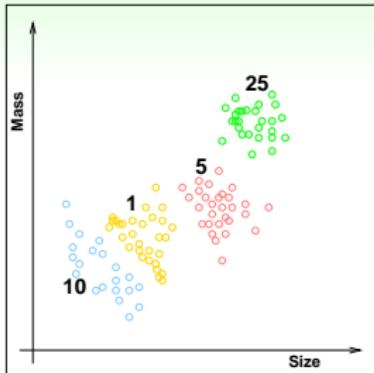
More Binary Classification Problems



- credit approve/disapprove
- email spam/non-spam
- patient sick/not sick
- ad profitable/not profitable
- answer correct/incorrect (KDDCup 2010)

core and important problem with
many tools as **building block of other tools**

Multiclass Classification: Coin Recognition Problem



- classify US coins (1c, 5c, 10c, 25c) by (size, mass)
- $\mathcal{Y} = \{1c, 5c, 10c, 25c\}$, or $\mathcal{Y} = \{1, 2, \dots, K\}$ (**abstractly**)
- binary classification: special case with $K = 2$

Other Multiclass Classification Problems

- written digits $\Rightarrow 0, 1, \dots, 9$
- pictures \Rightarrow apple, orange, strawberry
- emails \Rightarrow spam, primary, social, promotion, update (Google)

many applications in practice,
especially for 'recognition'

Regression: Patient Recovery Prediction Problem

- binary classification: patient features \Rightarrow sick or not
- multiclass classification: patient features \Rightarrow which type of cancer
- regression: patient features \Rightarrow **how many days before recovery**
- $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = [\text{lower}, \text{upper}] \subset \mathbb{R}$ (bounded regression)
 - deeply studied in statistics**

Other Regression Problems

- company data \Rightarrow stock price
- climate data \Rightarrow temperature

also core and important with many ‘statistical’ tools as **building block of other tools**

Structured Learning: Sequence Tagging Problem


I love ML
pronoun verb noun

- multiclass classification: word \Rightarrow word class
- structured learning:
sentence \Rightarrow structure (class of each word)
- $\mathcal{Y} = \{PVN, PVP, NVN, PV, \dots\}$, not including VVVVV
- huge multiclass classification problem
(structure \equiv hyperclass) **without 'explicit' class definition**

Other Structured Learning Problems

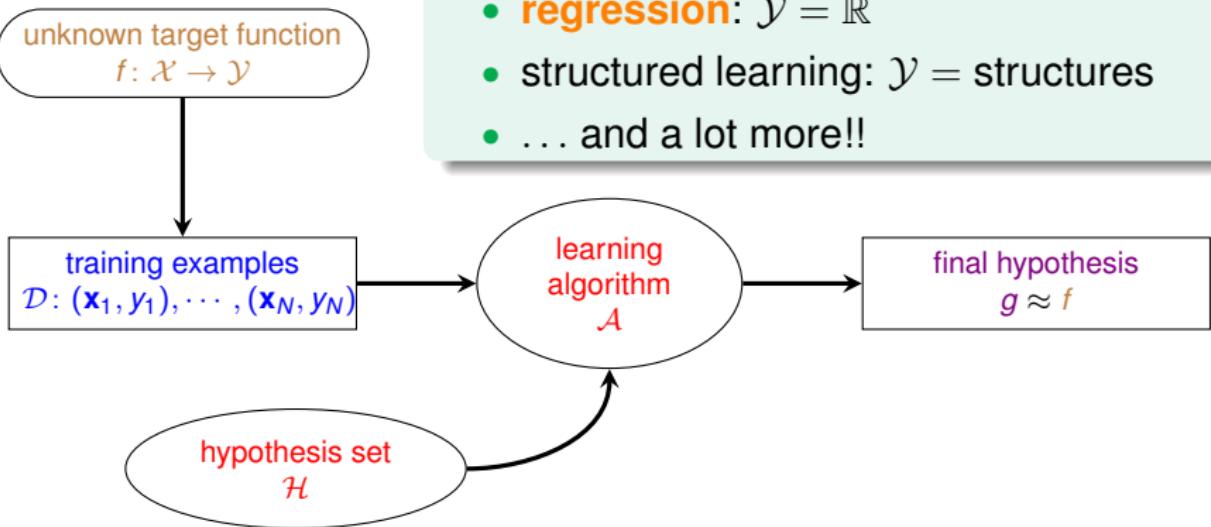
- protein data \Rightarrow protein folding
- speech data \Rightarrow speech parse tree

a fancy but complicated learning problem

Mini Summary

Learning with Different Output Space \mathcal{Y}

- **binary classification:** $\mathcal{Y} = \{-1, +1\}$
- multiclass classification: $\mathcal{Y} = \{1, 2, \dots, K\}$
- **regression:** $\mathcal{Y} = \mathbb{R}$
- structured learning: $\mathcal{Y} = \text{structures}$
- ... and a lot more!!



core tools: binary classification and regression

Fun Time

What is this learning problem?

The entrance system of the school gym, which does automatic face recognition based on machine learning, is built to charge four different groups of users differently: Staff, Student, Professor, Other. What type of learning problem best fits the need of the system?

- ① binary classification
- ② multiclass classification
- ③ regression
- ④ structured learning

Fun Time

What is this learning problem?

The entrance system of the school gym, which does automatic face recognition based on machine learning, is built to charge four different groups of users differently: Staff, Student, Professor, Other. What type of learning problem best fits the need of the system?

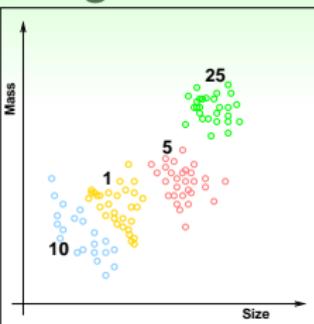
- ① binary classification
- ② multiclass classification
- ③ regression
- ④ structured learning

Reference Answer: ②

There is an ‘explicit’ \mathcal{Y} that contains four classes.

Supervised: Coin Recognition Revisited

unknown target function
 $f: \mathcal{X} \rightarrow \mathcal{Y}$



training examples
 $\mathcal{D}: (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

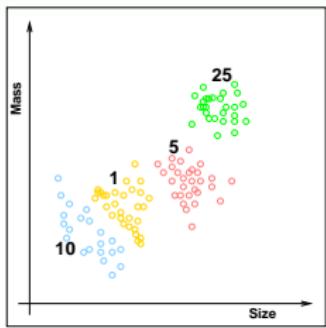
learning algorithm
 \mathcal{A}

final hypothesis
 $g \approx f$

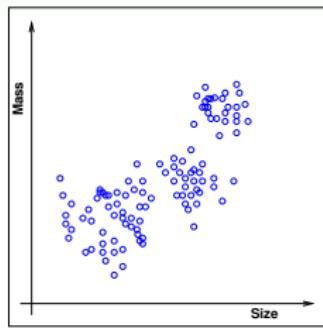
hypothesis set
 \mathcal{H}

supervised learning:
every \mathbf{x}_n comes with corresponding y_n

Unsupervised: Coin Recognition without y_n



supervised multiclass classification



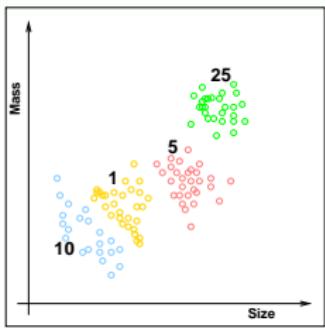
unsupervised multiclass classification
↔ ‘clustering’

Other Clustering Problems

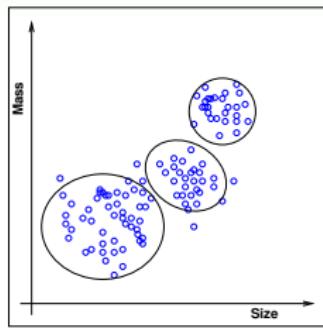
- articles \Rightarrow topics
- consumer profiles \Rightarrow consumer groups

clustering: a challenging but useful problem

Unsupervised: Coin Recognition without y_n



supervised multiclass classification



unsupervised multiclass classification
↔ ‘clustering’

Other Clustering Problems

- articles \Rightarrow topics
- consumer profiles \Rightarrow consumer groups

clustering: a challenging but useful problem

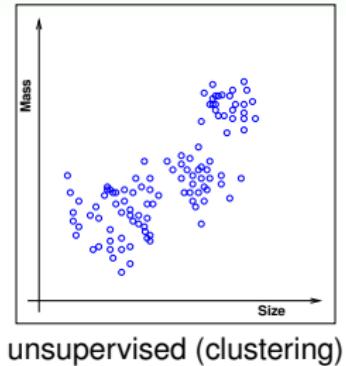
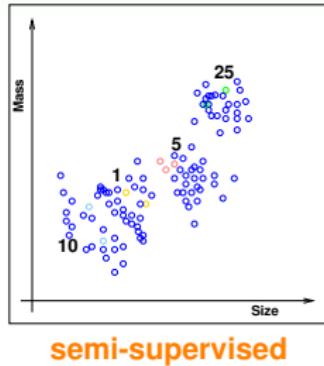
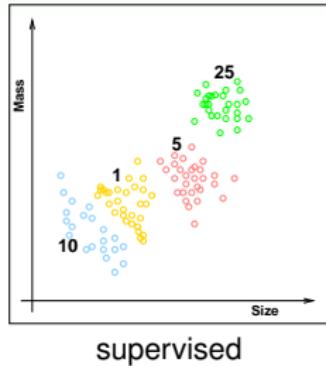
Unsupervised: Learning without y_n

Other Unsupervised Learning Problems

- clustering: $\{\mathbf{x}_n\} \Rightarrow \text{cluster}(\mathbf{x})$
(≈ ‘unsupervised multiclass classification’)
—i.e. articles ⇒ topics
- **density estimation**: $\{\mathbf{x}_n\} \Rightarrow \text{density}(\mathbf{x})$
(≈ ‘unsupervised bounded regression’)
—i.e. traffic reports with location ⇒ dangerous areas
- **outlier detection**: $\{\mathbf{x}_n\} \Rightarrow \text{unusual}(\mathbf{x})$
(≈ extreme ‘unsupervised binary classification’)
—i.e. Internet logs ⇒ intrusion alert
- ... and a lot more!!

unsupervised learning: diverse, with possibly
very different performance goals

Semi-supervised: Coin Recognition with Some y_n



Other Semi-supervised Learning Problems

- face images with a few labeled \Rightarrow face identifier (Facebook)
- medicine data with a few labeled \Rightarrow medicine effect predictor

semi-supervised learning: leverage
unlabeled data to avoid 'expensive' labeling

Reinforcement Learning

a ‘very different’ but natural way of learning

Teach Your Dog: Say ‘Sit Down’

The dog pees on the ground.

BAD DOG. THAT'S A VERY WRONG ACTION.

- cannot easily show the dog that $y_n = \text{sit}$ when $\mathbf{x}_n = \text{'sit down'}$
- but can ‘punish’ to say $\tilde{y}_n = \text{pee}$ is wrong



Other Reinforcement Learning Problems Using (\mathbf{x}, \tilde{y} , goodness)

- (customer, ad choice, ad click earning) \Rightarrow ad system
- (cards, strategy, winning amount) \Rightarrow black jack agent

reinforcement: learn with ‘**partial/implicit information**’ (often sequentially)

Reinforcement Learning

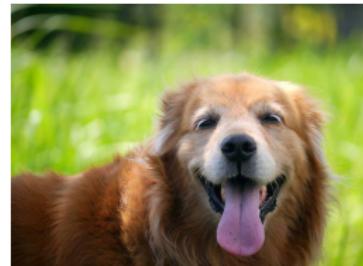
a ‘very different’ but natural way of learning

Teach Your Dog: Say ‘Sit Down’

The dog sits down.

Good Dog. Let me give you some cookies.

- still cannot show $y_n = \text{sit}$ when $\mathbf{x}_n = \text{'sit down'}$
- but can ‘reward’ to say $\tilde{y}_n = \text{sit}$ is good



Other Reinforcement Learning Problems Using (\mathbf{x}, \tilde{y} , goodness)

- (customer, ad choice, ad click earning) \Rightarrow ad system
- (cards, strategy, winning amount) \Rightarrow black jack agent

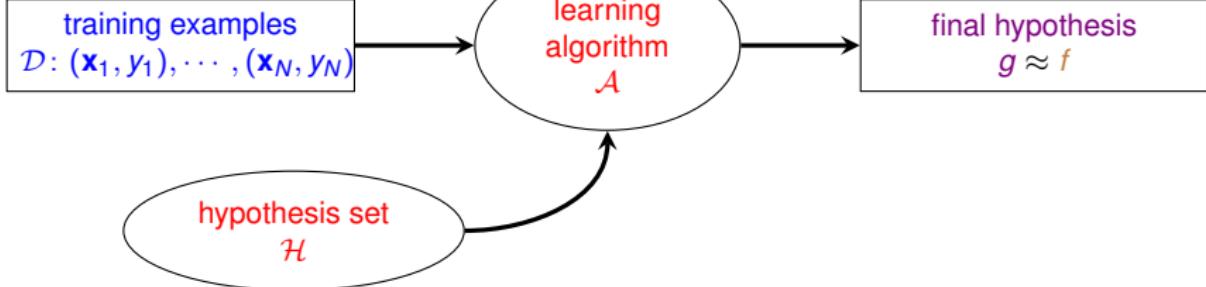
reinforcement: learn with ‘**partial/implicit information**’ (often sequentially)

Mini Summary

Learning with Different Data Label y_n

- **supervised**: all y_n
- unsupervised: no y_n
- semi-supervised: some y_n
- reinforcement: implicit y_n by goodness(\tilde{y}_n)
- ... and more!!

unknown target function
 $f: \mathcal{X} \rightarrow \mathcal{Y}$



core tool: supervised learning

Fun Time

What is this learning problem?

To build a tree recognition system, a company decides to gather one million of pictures on the Internet. Then, it asks each of the 10 company members to view 100 pictures and record whether each picture contains a tree. The pictures and records are then fed to a learning algorithm to build the system. What type of learning problem does the algorithm need to solve?

- ① supervised
- ② unsupervised
- ③ semi-supervised
- ④ reinforcement

Fun Time

What is this learning problem?

To build a tree recognition system, a company decides to gather one million of pictures on the Internet. Then, it asks each of the 10 company members to view 100 pictures and record whether each picture contains a tree. The pictures and records are then fed to a learning algorithm to build the system. What type of learning problem does the algorithm need to solve?

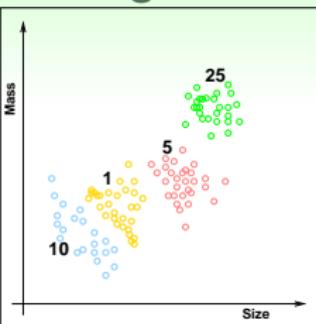
- ① supervised
- ② unsupervised
- ③ semi-supervised
- ④ reinforcement

Reference Answer: ③

The 1,000 records are the labeled (\mathbf{x}_n, y_n) ; the other 999,000 pictures are the unlabeled \mathbf{x}_n .

Batch Learning: Coin Recognition Revisited

unknown target function
 $f: \mathcal{X} \rightarrow \mathcal{Y}$



training examples
 $\mathcal{D}: (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

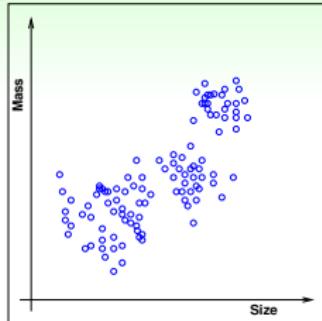
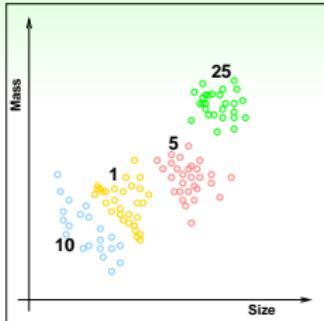
learning algorithm
 \mathcal{A}

final hypothesis
 $g \approx f$

hypothesis set
 \mathcal{H}

batch supervised multiclass classification:
learn from **all known** data

More Batch Learning Problems



- batch of (email, spam?) \Rightarrow spam filter
- batch of (patient, cancer) \Rightarrow cancer classifier
- batch of patient data \Rightarrow group of patients

batch learning: **a very common protocol**

Online: Spam Filter that ‘Improves’

- batch spam filter:
learn with known (email, spam?) pairs, and predict with fixed g
- **online** spam filter, which **sequentially**:
 - ① observe an email \mathbf{x}_t
 - ② predict spam status with current $g_t(\mathbf{x}_t)$
 - ③ receive ‘desired label’ y_t from user, and then update g_t with (\mathbf{x}_t, y_t)

Connection to What We Have Learned

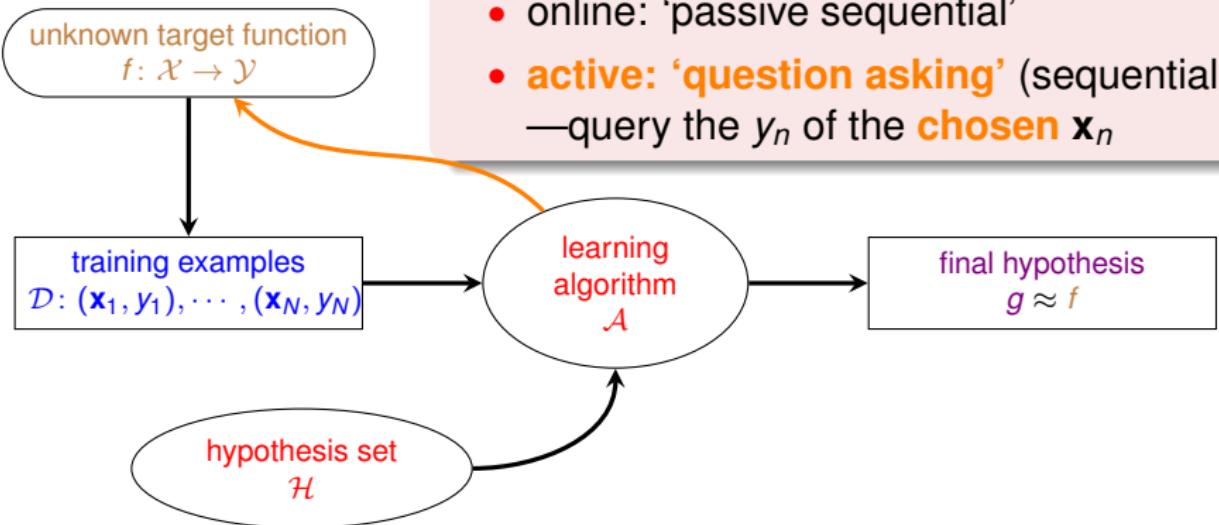
- PLA can be easily adapted to online protocol (how?)
- reinforcement learning is often done online (why?)

online: hypothesis ‘improves’ through receiving
data instances **sequentially**

Active Learning: Learning by ‘Asking’

Protocol \Leftrightarrow Learning Philosophy

- batch: ‘duck feeding’
- online: ‘passive sequential’
- **active: ‘question asking’** (sequentially)
 - query the y_n of the **chosen** \mathbf{x}_n

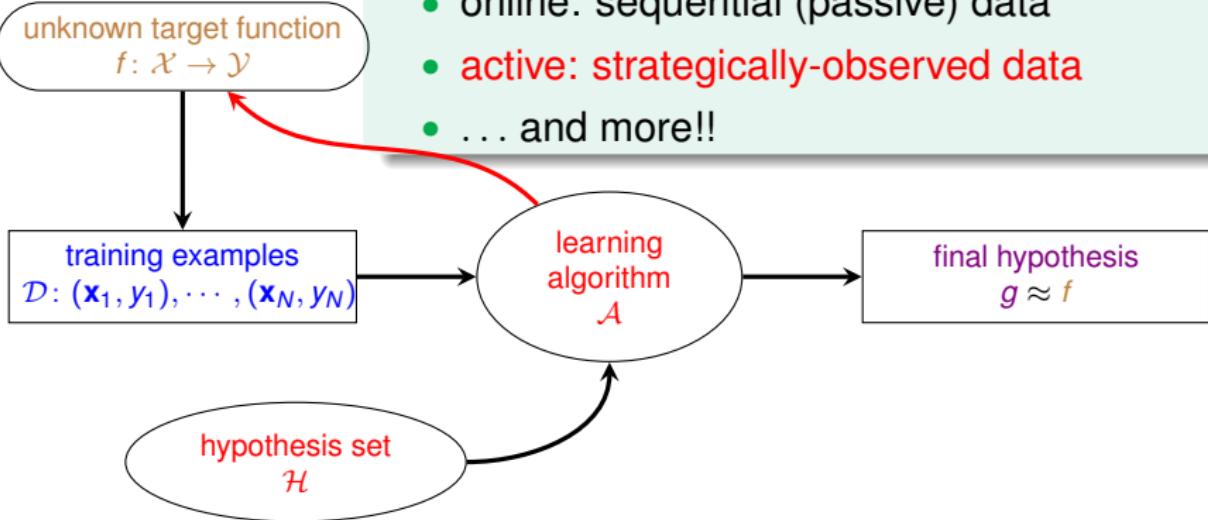


active: improve hypothesis with fewer labels
(hopefully) by asking questions **strategically**

Mini Summary

Learning with Different Protocol $f \Rightarrow (\mathbf{x}_n, y_n)$

- **batch**: all known data
- online: sequential (passive) data
- active: strategically-observed data
- ... and more!!



core protocol: batch

Fun Time

What is this learning problem?

A photographer has 100,000 pictures, each containing one baseball player. He wants to automatically categorize the pictures by its player inside. He starts by categorizing 1,000 pictures by himself, and then writes an algorithm that tries to categorize the other pictures if it is 'confident' on the category while pausing for (& learning from) human input if not. What protocol best describes the nature of the algorithm?

- 1 batch
- 2 online
- 3 active
- 4 random

Fun Time

What is this learning problem?

A photographer has 100,000 pictures, each containing one baseball player. He wants to automatically categorize the pictures by its player inside. He starts by categorizing 1,000 pictures by himself, and then writes an algorithm that tries to categorize the other pictures if it is 'confident' on the category while pausing for (& learning from) human input if not. What protocol best describes the nature of the algorithm?

- ① batch
- ② online
- ③ active
- ④ random

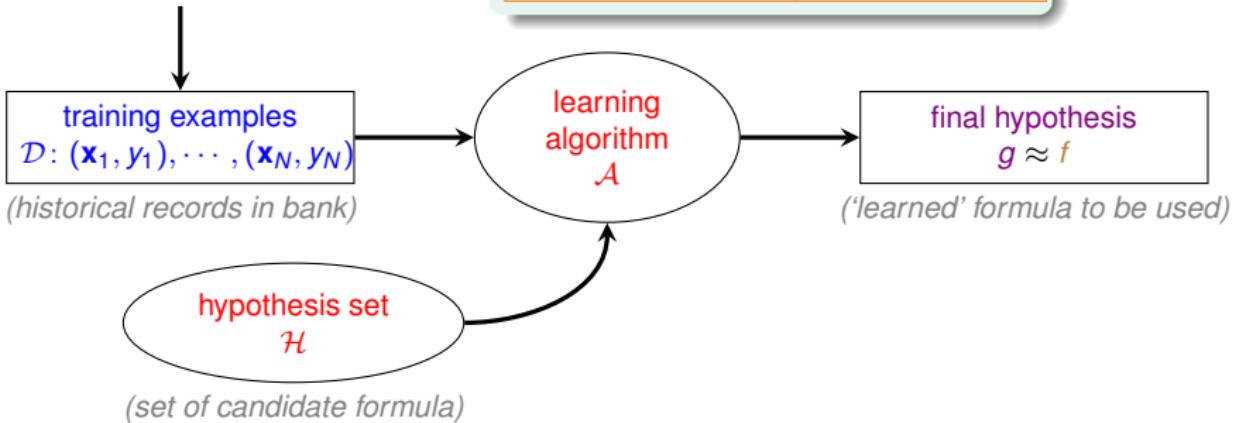
Reference Answer: ③

The algorithm takes an active but naïve strategy:
ask when 'confused'. **You should probably
do the same when taking a class. :-)**

Credit Approval Problem Revisited

unknown target function
 $f: \mathcal{X} \rightarrow \mathcal{Y}$
(ideal credit approval formula)

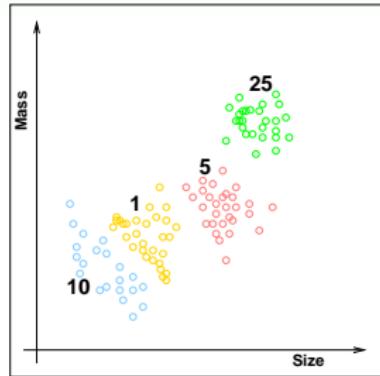
age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000



concrete features: each dimension of $\mathcal{X} \subseteq \mathbb{R}^d$
represents 'sophisticated physical meaning'

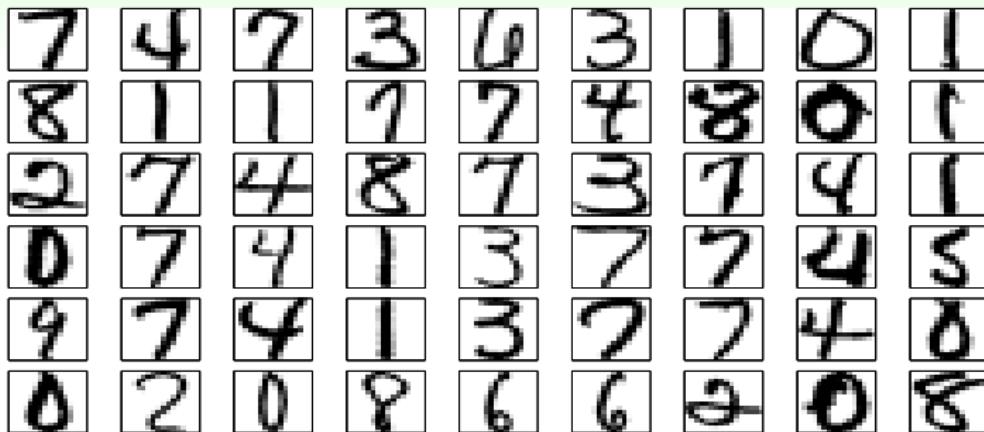
More on Concrete Features

- **(size, mass)** for coin classification
- **customer info** for credit approval
- **patient info** for cancer diagnosis
- often including ‘human intelligence’ on the learning task



concrete features: the ‘easy’ ones for ML

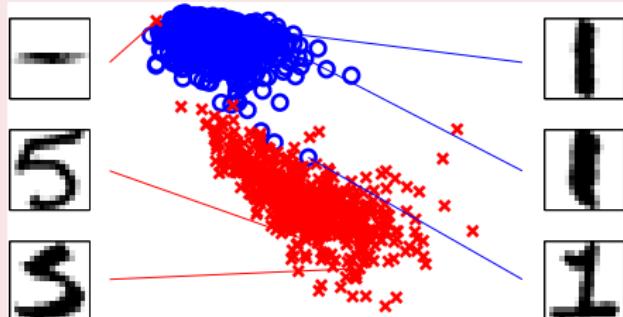
Raw Features: Digit Recognition Problem (1/2)



- digit recognition problem: features \Rightarrow meaning of digit
- a typical supervised multiclass classification problem

Raw Features: Digit Recognition Problem (2/2)

by Concrete Features



$$\mathbf{x} = (\text{symmetry}, \text{density})$$

by Raw Features

- 16 by 16 gray image $\mathbf{x} \equiv (0, 0, 0.9, 0.6, \dots) \in \mathbb{R}^{256}$
- '**simple** physical meaning'; thus more difficult for ML than concrete features

Other Problems with Raw Features

- image pixels, speech signal, etc.

raw features: often need human or machines
to **convert to concrete ones**

Abstract Features: Rating Prediction Problem

Rating Prediction Problem (KDDCup 2011)

- given previous (userid, itemid, rating) tuples, predict the rating that some userid would give to itemid?
- a regression problem with $\mathcal{Y} \subseteq \mathbb{R}$ as rating and $\mathcal{X} \subseteq \mathbb{N} \times \mathbb{N}$ as (userid, itemid)
- ‘no physical meaning’; thus even more difficult for ML

Other Problems with Abstract Features

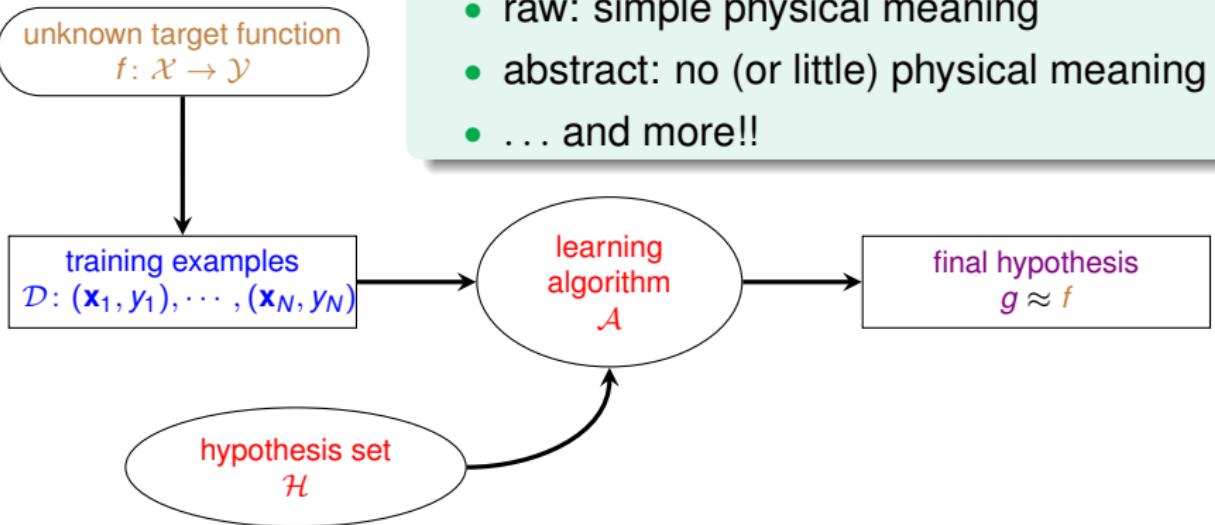
- student ID in online tutoring system (KDDCup 2010)
- advertisement ID in online ad system

abstract: again need ‘**feature conversion**/extraction/construction’

Mini Summary

Learning with Different Input Space \mathcal{X}

- **concrete**: sophisticated (and related) physical meaning
- raw: simple physical meaning
- abstract: no (or little) physical meaning
- ... and more!!



'easy' input: concrete

Fun Time

What features can be used?

Consider a problem of building an online image advertisement system that shows the users the most relevant images. What features can you choose to use?

- ① concrete
- ② concrete, raw
- ③ concrete, abstract
- ④ concrete, raw, abstract

Fun Time

What features can be used?

Consider a problem of building an online image advertisement system that shows the users the most relevant images. What features can you choose to use?

- ① concrete
- ② concrete, raw
- ③ concrete, abstract
- ④ concrete, raw, abstract

Reference Answer: ④

concrete user features, raw image features,
and maybe abstract user/image IDs

Summary

① When Can Machines Learn?

Lecture 2: Learning to Answer Yes/No

Lecture 3: Types of Learning

- Learning with Different Output Space \mathcal{Y}
[classification], [regression], structured
- Learning with Different Data Label y_n
[supervised], un/semi-supervised, reinforcement
- Learning with Different Protocol $f \Rightarrow (\mathbf{x}_n, y_n)$
[batch], online, active
- Learning with Different Input Space \mathcal{X}
[concrete], raw, abstract

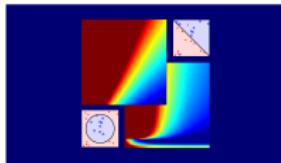
- next: learning is impossible?!

② Why Can Machines Learn?

③ How Can Machines Learn?

④ How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 4: Feasibility of 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 3: Types of Learning

focus: **binary classification** or **regression** from a
batch of **supervised** data with **concrete** features

Lecture 4: Feasibility of Learning

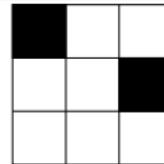
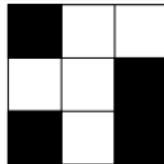
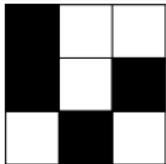
- Learning is Impossible?
- Probability to the Rescue
- Connection to Learning
- Connection to Real Learning

② Why Can Machines Learn?

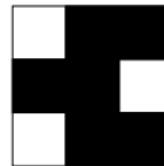
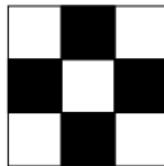
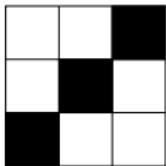
③ How Can Machines Learn?

④ How Can Machines Learn Better?

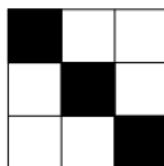
A Learning Puzzle



$$y_n = -1$$



$$y_n = +1$$



$$g(\mathbf{x}) = ?$$

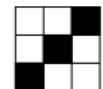
let's test your 'human learning'
with 6 examples :-)

Two Controversial Answers

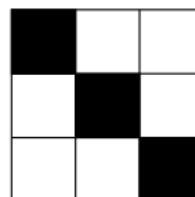
whatever you say about $g(\mathbf{x})$,



$$y_n = -1$$



$$y_n = +1$$



$$g(\mathbf{x}) = ?$$

truth $f(\mathbf{x}) = +1$ because ...

- symmetry $\Leftrightarrow +1$
- (black or white count = 3) or
(black count = 4 and
middle-top black) $\Leftrightarrow +1$

truth $f(\mathbf{x}) = -1$ because ...

- left-top black $\Leftrightarrow -1$
- middle column contains at
most 1 black and right-top
white $\Leftrightarrow -1$

all valid reasons, your **adversarial teacher**
can always call you '**didn't learn**'. :-(

A ‘Simple’ Binary Classification Problem

\mathbf{x}_n	$y_n = f(\mathbf{x}_n)$
0 0 0	o
0 0 1	x
0 1 0	x
0 1 1	o
1 0 0	x

- $\mathcal{X} = \{0, 1\}^3$, $\mathcal{Y} = \{\text{o}, \text{x}\}$, can enumerate all candidate f as \mathcal{H}

pick $g \in \mathcal{H}$ with all $g(\mathbf{x}_n) = y_n$ (like PLA),
does $g \approx f$?

No Free Lunch

 \mathcal{D}

x	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0 0 0	o	o	o	o	o	o	o	o	o	o
0 0 1	x	x	x	x	x	x	x	x	x	x
0 1 0	x	x	x	x	x	x	x	x	x	x
0 1 1	o	o	o	o	o	o	o	o	o	o
1 0 0	x	x	x	x	x	x	x	x	x	x
1 0 1		?	o	o	o	o	x	x	x	x
1 1 0		?	o	o	x	x	o	o	x	x
1 1 1		?	o	x	o	x	o	x	o	x

- $g \approx f$ inside \mathcal{D} : sure!
- $g \approx f$ outside \mathcal{D} : **No!** (but that's really what we want!)

learning from \mathcal{D} (to infer something outside \mathcal{D})
is doomed if **any 'unknown' f can happen.** :-)

Fun Time

This is a popular ‘brain-storming’ problem, with a claim that 2% of the world’s cleverest population can crack its ‘hidden pattern’.

$$(5, 3, 2) \rightarrow 151022, \quad (7, 2, 5) \rightarrow ?$$

It is like a ‘learning problem’ with $N = 1$, $\mathbf{x}_1 = (5, 3, 2)$, $y_1 = 151022$. Learn a hypothesis from the one example to predict on $\mathbf{x} = (7, 2, 5)$. What is your answer?

- ① 151026 ③ I need more examples to get the correct answer
- ② 143547 ④ there is no ‘correct’ answer

Fun Time

This is a popular ‘brain-storming’ problem, with a claim that 2% of the world’s cleverest population can crack its ‘hidden pattern’.

$$(5, 3, 2) \rightarrow 151022, \quad (7, 2, 5) \rightarrow ?$$

It is like a ‘learning problem’ with $N = 1$, $\mathbf{x}_1 = (5, 3, 2)$, $y_1 = 151022$. Learn a hypothesis from the one example to predict on $\mathbf{x} = (7, 2, 5)$. What is your answer?

- ① 151026 ③ I need more examples to get the correct answer
- ② 143547 ④ there is no ‘correct’ answer

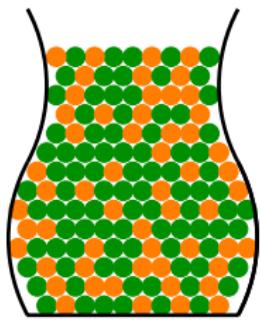
Reference Answer: ④

Following the same nature of the no-free-lunch problems discussed, we cannot hope to be correct under this ‘adversarial’ setting. BTW,

② is the designer’s answer: the first two digits = $x_1 \cdot x_2$; the next two digits = $x_1 \cdot x_3$; the last two digits = $(x_1 \cdot x_2 + x_1 \cdot x_3 - x_2)$.

Inferring Something Unknown

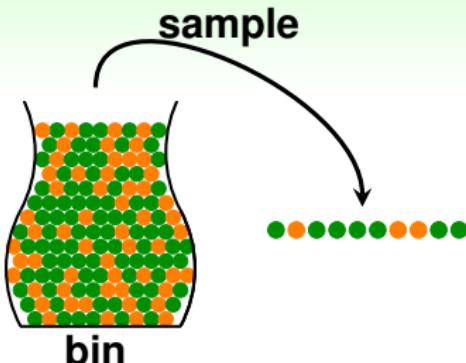
difficult to infer **unknown target f outside \mathcal{D}** in learning;
can we infer **something unknown** in **other scenarios?**



- consider a bin of many many **orange** and **green** marbles
- do we **know** the **orange** portion (probability)? **No!**

can you **infer** the **orange** probability?

Statistics 101: Inferring Orange Probability

**bin**

assume

orange probability = μ ,
green probability = $1 - \mu$,
with μ **unknown**

sample

N marbles sampled independently, with
orange fraction = ν ,
green fraction = $1 - \nu$,
now ν **known**

does **in-sample** ν say anything about
out-of-sample μ ?

Possible versus Probable

does **in-sample ν** say anything about out-of-sample μ ?

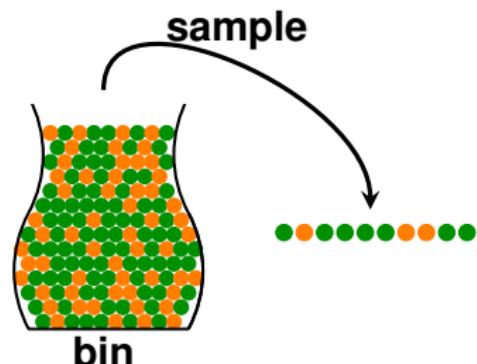
No!

possibly not: sample can be mostly green while bin is mostly orange

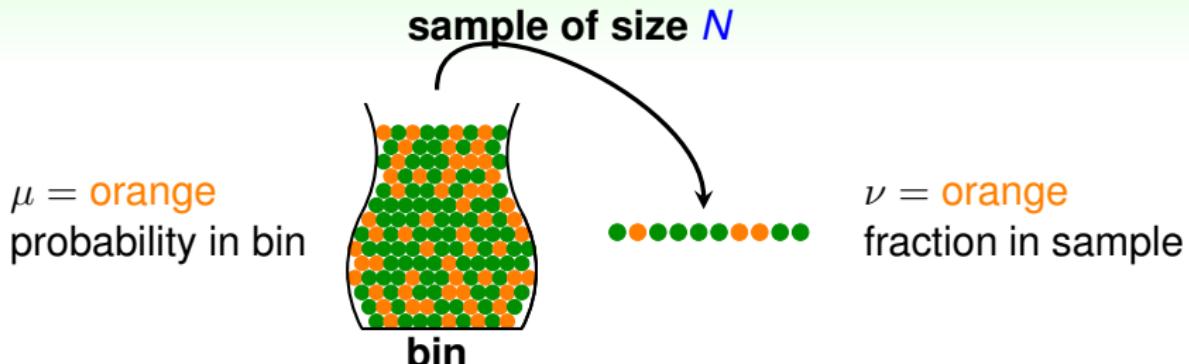
Yes!

probably yes: in-sample ν likely **close to** unknown μ

formally, **what does ν say about μ ?**



Hoeffding's Inequality (1/2)



- in big sample (N large), ν is probably close to μ (within ϵ)

$$\mathbb{P} [|\nu - \mu| > \epsilon] \leq 2 \exp(-2\epsilon^2 N)$$

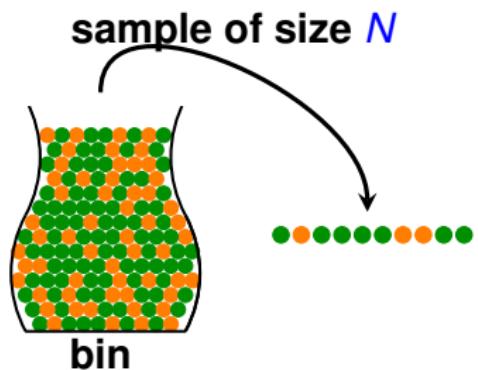
- called **Hoeffding's Inequality**, for marbles, coin, polling, ...

the statement ' $\nu = \mu$ ' is
probably approximately correct (PAC)

Hoeffding's Inequality (2/2)

$$\mathbb{P} [|\nu - \mu| > \epsilon] \leq 2 \exp(-2\epsilon^2 N)$$

- valid for all N and ϵ
- does not depend on μ ,
no need to 'know' μ
- larger sample size N or
looser gap ϵ
 \implies higher probability for ' $\nu \approx \mu$ '



if **large N** , can **probably** infer
unknown μ by known ν

Fun Time

Let $\mu = 0.4$. Use Hoeffding's Inequality

$$\mathbb{P} [|\nu - \mu| > \epsilon] \leq 2 \exp (-2\epsilon^2 N)$$

to bound the probability that a sample of 10 marbles will have $\nu \leq 0.1$. What bound do you get?

- 1 0.67
- 2 0.40
- 3 0.33
- 4 0.05

Fun Time

Let $\mu = 0.4$. Use Hoeffding's Inequality

$$\mathbb{P} [|\nu - \mu| > \epsilon] \leq 2 \exp (-2\epsilon^2 N)$$

to bound the probability that a sample of 10 marbles will have $\nu \leq 0.1$. What bound do you get?

- 1 0.67
- 2 0.40
- 3 0.33
- 4 0.05

Reference Answer: (3)

Set $N = 10$ and $\epsilon = 0.3$ and you get the answer. BTW, (4) is the actual probability and Hoeffding gives only an upper bound to that.

Connection to Learning

bin

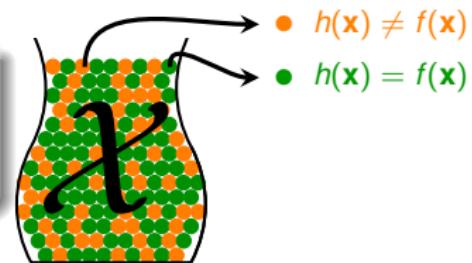
- unknown orange prob. μ
- marble $\bullet \in$ bin
- orange \bullet
- green \bullet
- size- N sample from bin

of i.i.d. marbles

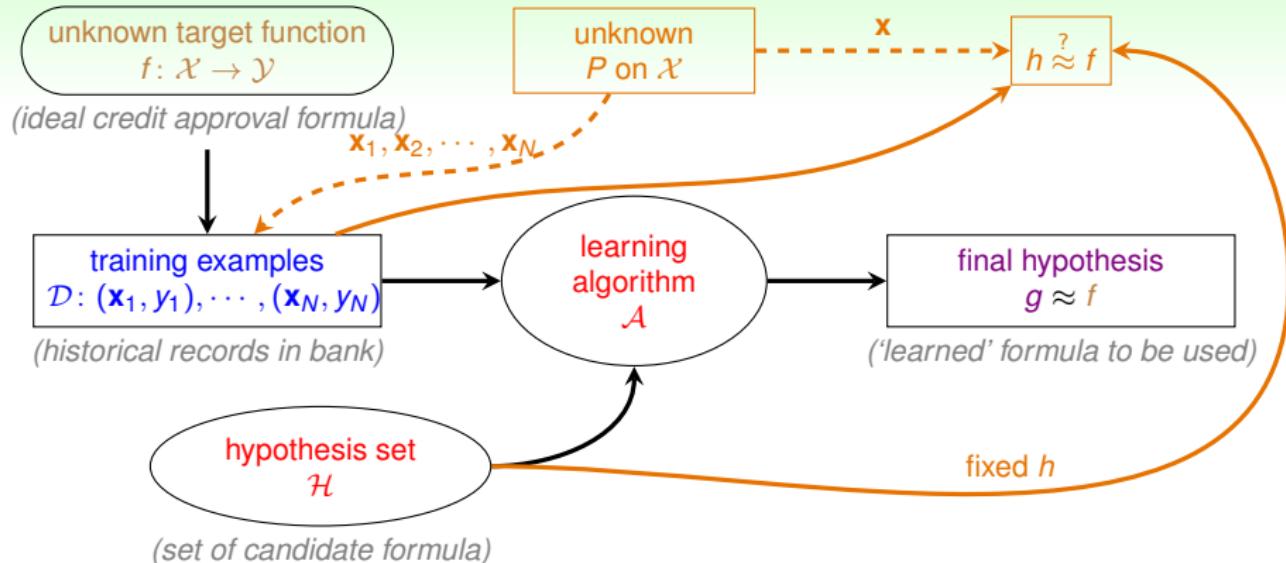
learning

- fixed hypothesis $h(\mathbf{x}) \stackrel{?}{=} \text{target } f(\mathbf{x})$
- $\mathbf{x} \in \mathcal{X}$
- h is wrong $\Leftrightarrow h(\mathbf{x}) \neq f(\mathbf{x})$
- h is right $\Leftrightarrow h(\mathbf{x}) = f(\mathbf{x})$
- check h on $\mathcal{D} = \{(\mathbf{x}_n, \underbrace{y_n}_{f(\mathbf{x}_n)})\}$
with i.i.d. \mathbf{x}_n

if **large N** & **i.i.d. \mathbf{x}_n** , can **probably** infer
unknown $[h(\mathbf{x}) \neq f(\mathbf{x})]$ probability
by known $[h(\mathbf{x}_n) \neq y_n]$ fraction



Added Components



for any fixed h , can probably infer

$$\text{unknown } E_{\text{out}}(h) = \mathbb{E}_{\mathbf{x} \sim P} [\llbracket h(\mathbf{x}) \neq f(\mathbf{x}) \rrbracket]$$

$$\text{by known } E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^N [\llbracket h(\mathbf{x}_n) \neq y_n \rrbracket].$$

The Formal Guarantee

for any fixed h , in ‘big’ data (N large),

in-sample error $E_{\text{in}}(h)$ is probably close to

out-of-sample error $E_{\text{out}}(h)$ (within ϵ)

$$\mathbb{P} [|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon] \leq 2 \exp(-2\epsilon^2 N)$$

same as the ‘bin’ analogy . . .

- valid for all N and ϵ
- does not depend on $E_{\text{out}}(h)$, no need to ‘know’ $E_{\text{out}}(h)$
— f and P can stay unknown
- ‘ $E_{\text{in}}(h) = E_{\text{out}}(h)$ ’ is probably approximately correct (PAC)

if ‘ $E_{\text{in}}(h) \approx E_{\text{out}}(h)$ ’ and ‘ $E_{\text{in}}(h)$ small’
 $\Rightarrow E_{\text{out}}(h)$ small $\Rightarrow h \approx f$ with respect to P

Verification of One h

for any fixed h , when data large enough,

$$E_{\text{in}}(h) \approx E_{\text{out}}(h)$$

Can we claim ‘good learning’ ($g \approx f$)?

Yes!

if $E_{\text{in}}(h)$ small for the fixed h
and \mathcal{A} pick the h as g
 $\Rightarrow 'g = f'$ PAC

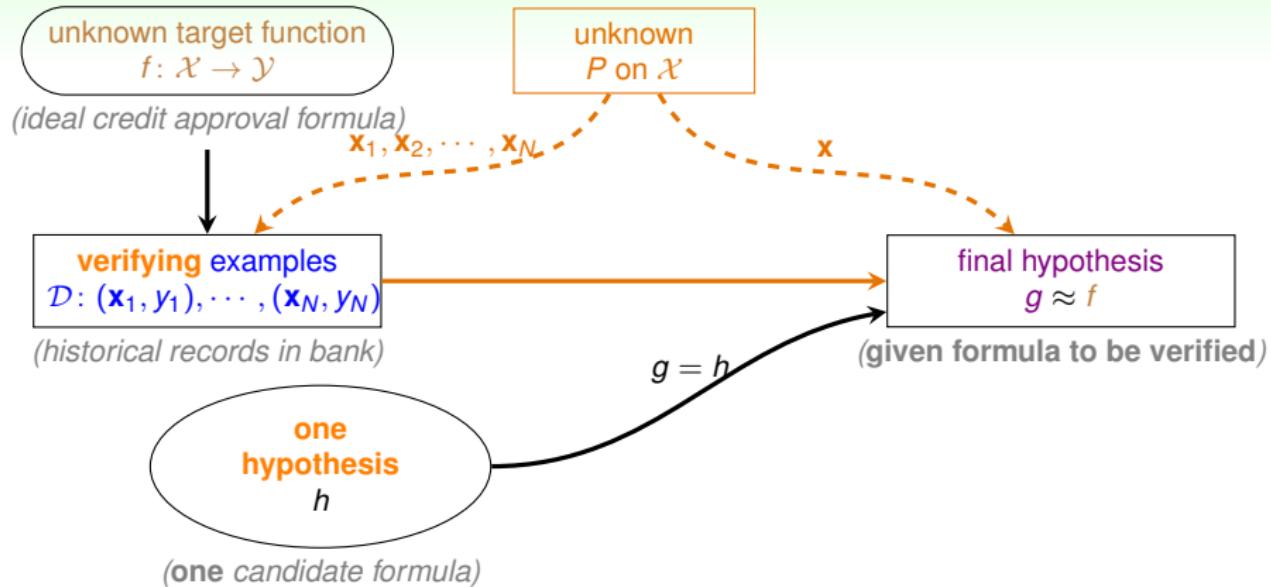
No!

if \mathcal{A} forced to pick THE h as g
 $\Rightarrow E_{\text{in}}(h)$ almost always not small
 $\Rightarrow 'g \neq f'$ PAC!

real learning:

\mathcal{A} shall make choices $\in \mathcal{H}$ (like PLA)
rather than being forced to pick one h . :-(

The ‘Verification’ Flow



can now use ‘historical records’ (data) to
verify ‘one candidate formula’ h

Fun Time

Your friend tells you her secret rule in investing in a particular stock: 'Whenever the stock goes down in the morning, it will go up in the afternoon; vice versa.' **To verify the rule, you chose 100 days uniformly at random from the past 10 years of stock data, and found that 80 of them satisfy the rule.** What is the best guarantee that you can get from the verification?

- ① You'll definitely be rich by exploiting the rule in the next 100 days.
- ② You'll likely be rich by exploiting the rule in the next 100 days, if the market behaves similarly to the last 10 years.
- ③ You'll likely be rich by exploiting the 'best rule' from 20 more friends in the next 100 days.
- ④ You'd definitely have been rich if you had exploited the rule in the past 10 years.

Fun Time

Your friend tells you her secret rule in investing in a particular stock: 'Whenever the stock goes down in the morning, it will go up in the afternoon; vice versa.' **To verify the rule, you chose 100 days uniformly at random from the past 10 years of stock data, and found that 80 of them satisfy the rule.** What is the best guarantee that you can get from the verification?

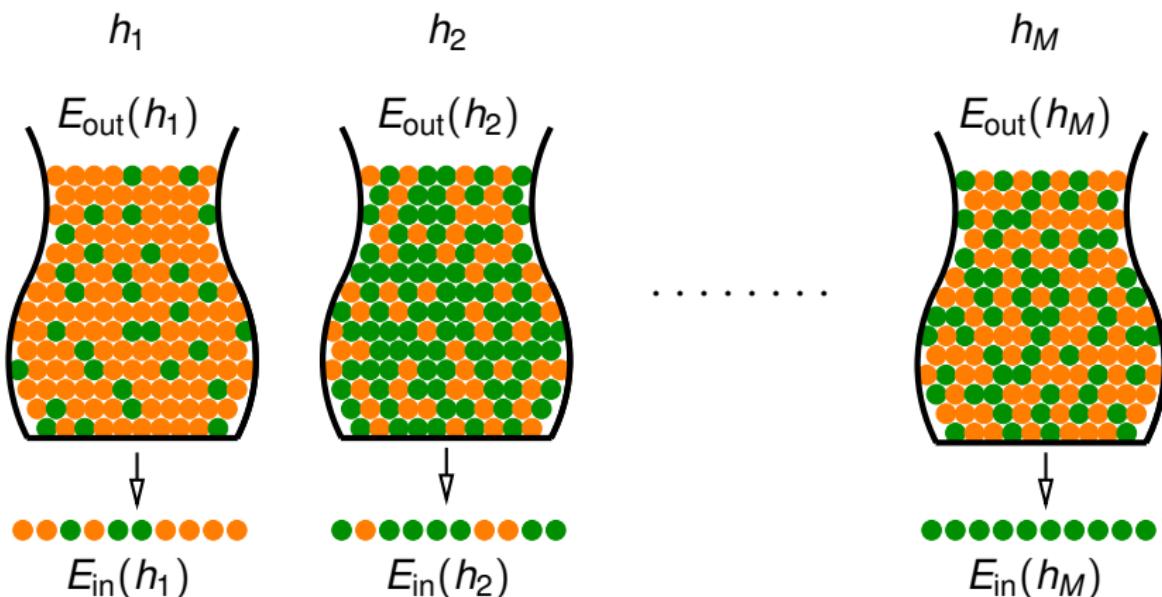
- ① You'll definitely be rich by exploiting the rule in the next 100 days.
- ② You'll likely be rich by exploiting the rule in the next 100 days, if the market behaves similarly to the last 10 years.
- ③ You'll likely be rich by exploiting the 'best rule' from 20 more friends in the next 100 days.
- ④ You'd definitely have been rich if you had exploited the rule in the past 10 years.

Reference Answer: ②

①: no free lunch; ③: no 'learning' guarantee in verification; ④: verifying with only 100 days, possible that the rule is mostly wrong for whole 10 years.

Multiple h

top

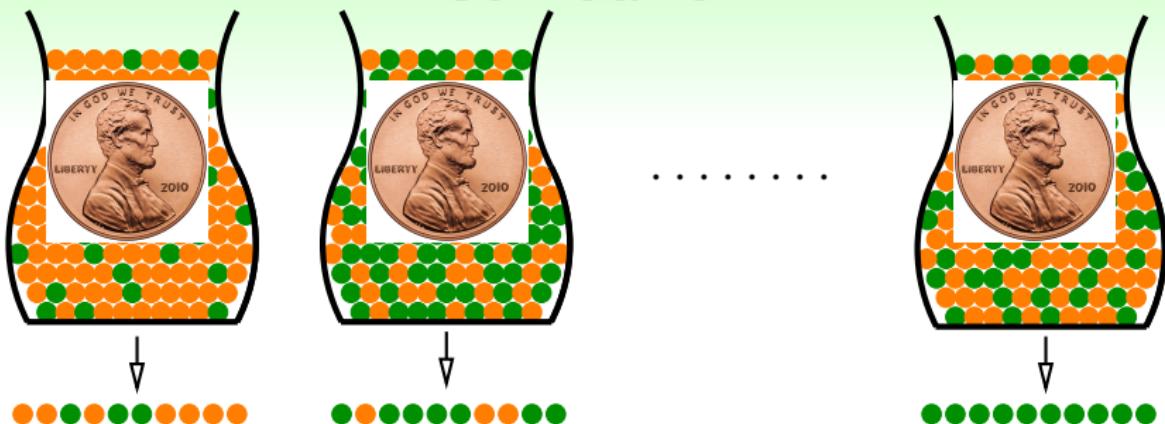


real learning (say like PLA):

BINGO when getting ?

bottom

Coin Game



Q: if everyone in size-150 NTU ML class flips a coin 5 times, and one of the students gets 5 heads for her coin 'g'. Is 'g' really magical?

A: No. Even if all coins are fair, the probability that one of the coins results in 5 heads is $1 - \left(\frac{31}{32}\right)^{150} > 99\%$.

BAD sample: E_{in} and E_{out} far away
—can get worse when involving 'choice'

BAD Sample and BAD Data

BAD Sample

e.g., $E_{\text{out}} = \frac{1}{2}$, but getting all heads ($E_{\text{in}} = 0$)!

BAD Data for One h

$E_{\text{out}}(h)$ and $E_{\text{in}}(h)$ far away:

e.g., E_{out} big (far from f), but E_{in} small (correct on most examples)

	\mathcal{D}_1	\mathcal{D}_2	...	\mathcal{D}_{1126}	...	\mathcal{D}_{5678}	...	Hoeffding
h	BAD					BAD		$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h] \leq \dots$

Hoeffding: small

$$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D}] = \sum_{\text{all possible } \mathcal{D}} \mathbb{P}(\mathcal{D}) \cdot [\text{BAD } \mathcal{D}]$$

BAD Data for Many h

BAD data for many h

\iff no 'freedom of choice' by \mathcal{A}

\iff there exists some h such that $E_{\text{out}}(h)$ and $E_{\text{in}}(h)$ far away

	\mathcal{D}_1	\mathcal{D}_2	...	\mathcal{D}_{1126}	...	\mathcal{D}_{5678}	Hoeffding
h_1	BAD					BAD	$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h_1] \leq \dots$
h_2		BAD					$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h_2] \leq \dots$
h_3	BAD	BAD				BAD	$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h_3] \leq \dots$
...							
h_M	BAD					BAD	$\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D} \text{ for } h_M] \leq \dots$
all	BAD	BAD				BAD	?

for M hypotheses, bound of $\mathbb{P}_{\mathcal{D}} [\text{BAD } \mathcal{D}]$?

Bound of BAD Data

$$\begin{aligned}
 & \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D}] \\
 = & \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D} \text{ for } h_1 \text{ or } \text{BAD } \mathcal{D} \text{ for } h_2 \text{ or } \dots \text{ or } \text{BAD } \mathcal{D} \text{ for } h_M] \\
 \leq & \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D} \text{ for } h_1] + \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D} \text{ for } h_2] + \dots + \mathbb{P}_{\mathcal{D}}[\text{BAD } \mathcal{D} \text{ for } h_M] \\
 & (\text{union bound}) \\
 \leq & 2 \exp(-2\epsilon^2 N) + 2 \exp(-2\epsilon^2 N) + \dots + 2 \exp(-2\epsilon^2 N) \\
 = & 2M \exp(-2\epsilon^2 N)
 \end{aligned}$$

- finite-bin version of Hoeffding, valid for all M , N and ϵ
- does not depend on any $E_{\text{out}}(h_m)$, no need to ‘know’ $E_{\text{out}}(h_m)$
— f and P can stay unknown
- ‘ $E_{\text{in}}(g) = E_{\text{out}}(g)$ ’ is PAC, regardless of \mathcal{A}

‘most reasonable’ \mathcal{A} (like PLA/pocket):
pick the h_m with lowest $E_{\text{in}}(h_m)$ as g

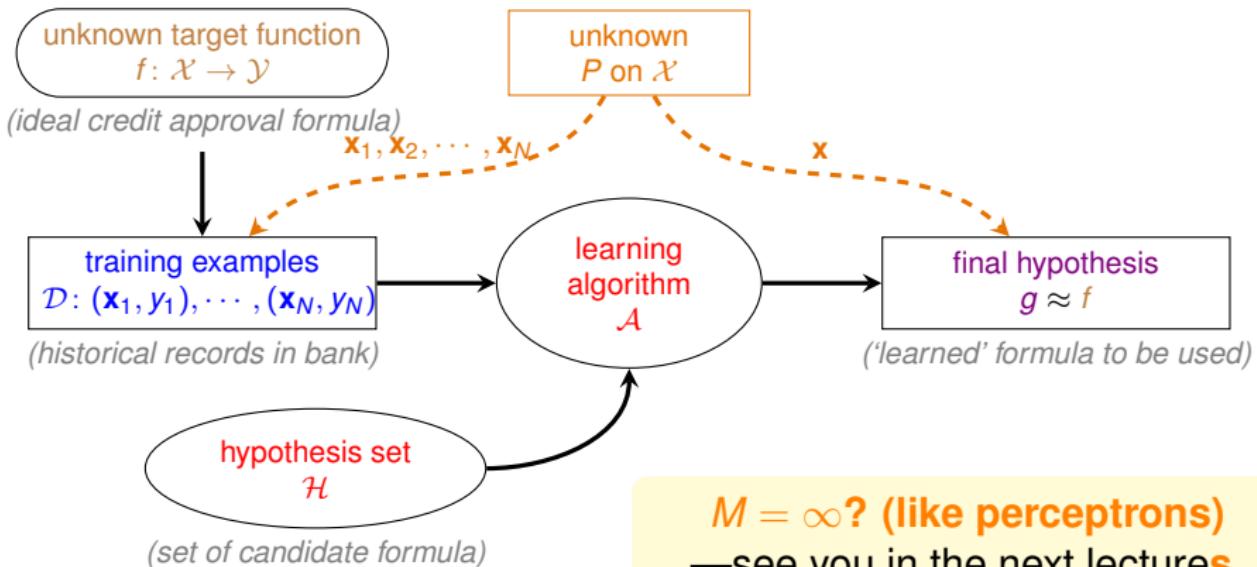
The ‘Statistical’ Learning Flow

if $|\mathcal{H}| = M$ finite, N large enough,

for whatever g picked by \mathcal{A} , $E_{\text{out}}(g) \approx E_{\text{in}}(g)$

if \mathcal{A} finds one g with $E_{\text{in}}(g) \approx 0$,

PAC guarantee for $E_{\text{out}}(g) \approx 0 \implies \text{learning possible :-)}$



Fun Time

Consider 4 hypotheses.

$$\begin{aligned} h_1(\mathbf{x}) &= \text{sign}(x_1), \quad h_2(\mathbf{x}) = \text{sign}(x_2), \\ h_3(\mathbf{x}) &= \text{sign}(-x_1), \quad h_4(\mathbf{x}) = \text{sign}(-x_2). \end{aligned}$$

For any N and ϵ , which of the following statement is not true?

- ① the **BAD** data of h_1 and the **BAD** data of h_2 are exactly the same
- ② the **BAD** data of h_1 and the **BAD** data of h_3 are exactly the same
- ③ $\mathbb{P}_{\mathcal{D}}[\text{BAD for some } h_k] \leq 8 \exp(-2\epsilon^2 N)$
- ④ $\mathbb{P}_{\mathcal{D}}[\text{BAD for some } h_k] \leq 4 \exp(-2\epsilon^2 N)$

Fun Time

Consider 4 hypotheses.

$$\begin{aligned} h_1(\mathbf{x}) &= \text{sign}(x_1), \quad h_2(\mathbf{x}) = \text{sign}(x_2), \\ h_3(\mathbf{x}) &= \text{sign}(-x_1), \quad h_4(\mathbf{x}) = \text{sign}(-x_2). \end{aligned}$$

For any N and ϵ , which of the following statement is not true?

- ① the **BAD** data of h_1 and the **BAD** data of h_2 are exactly the same
- ② the **BAD** data of h_1 and the **BAD** data of h_3 are exactly the same
- ③ $\mathbb{P}_{\mathcal{D}}[\text{BAD for some } h_k] \leq 8 \exp(-2\epsilon^2 N)$
- ④ $\mathbb{P}_{\mathcal{D}}[\text{BAD for some } h_k] \leq 4 \exp(-2\epsilon^2 N)$

Reference Answer: ①

The important thing is to note that ② is true, which implies that ④ is true if you revisit the union bound. Similar ideas will be used to conquer the $M = \infty$ case.

Summary

1 When Can Machines Learn?

Lecture 3: Types of Learning

Lecture 4: Feasibility of Learning

- Learning is Impossible?
absolutely no free lunch outside \mathcal{D}
- Probability to the Rescue
probably approximately correct outside \mathcal{D}
- Connection to Learning
verification possible if $E_{\text{in}}(h)$ small for fixed h
- Connection to Real Learning
learning possible if $|\mathcal{H}|$ finite and $E_{\text{in}}(g)$ small

2 Why Can Machines Learn?

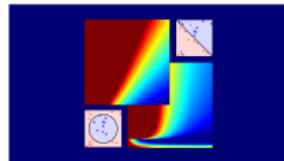
- next: what if $|\mathcal{H}| = \infty$?

3 How Can Machines Learn?

4 How Can Machines Learn Better?

Machine Learning Foundations

(機器學習基石)



Lecture 5: Training versus Testing

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 4: Feasibility of Learning

learning is **PAC**-possible
if enough **statistical data** and **finite** $|\mathcal{H}|$

2 Why Can Machines Learn?

Lecture 5: Training versus Testing

- Recap and Preview
- Effective Number of Lines
- Effective Number of Hypotheses
- Break Point

3 How Can Machines Learn?

4 How Can Machines Learn Better?

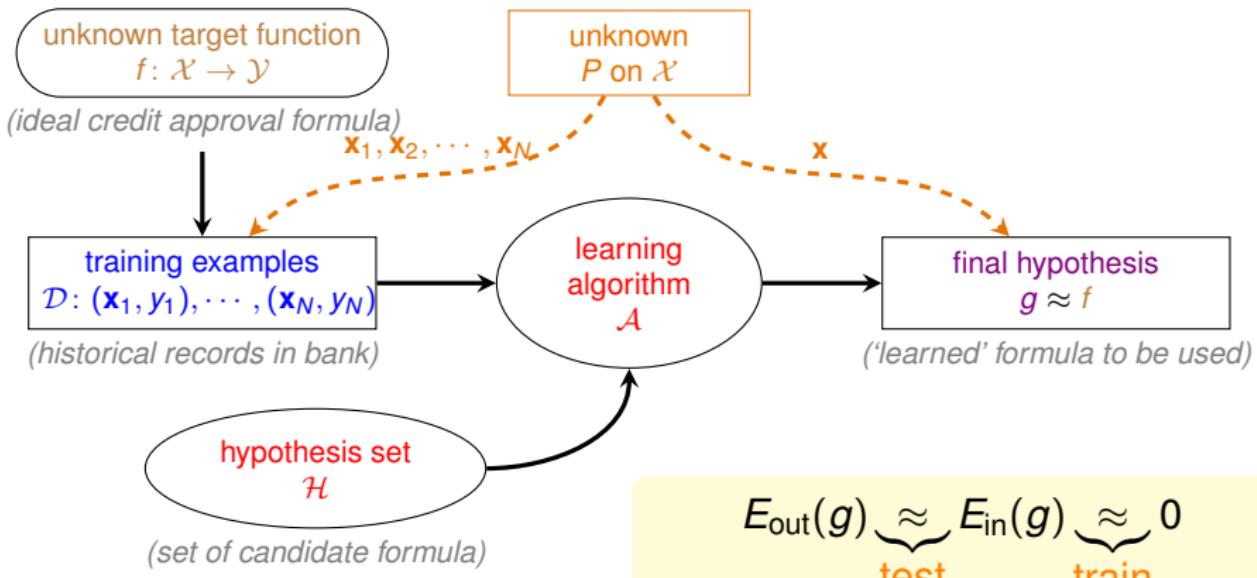
Recap: the ‘Statistical’ Learning Flow

if $|\mathcal{H}| = M$ finite, N large enough,

for whatever g picked by \mathcal{A} , $E_{\text{out}}(g) \approx E_{\text{in}}(g)$

if \mathcal{A} finds one g with $E_{\text{in}}(g) \approx 0$,

PAC guarantee for $E_{\text{out}}(g) \approx 0 \implies \text{learning possible :-)}$



Two Central Questions

for batch & supervised binary classification,

$\underbrace{g \approx f}_{\text{lecture 3}} \iff E_{\text{out}}(g) \approx 0$

achieved through $E_{\text{out}}(g) \approx E_{\text{in}}(g)$ and $E_{\text{in}}(g) \approx 0$

$\underbrace{E_{\text{out}}(g) \approx E_{\text{in}}(g)}_{\text{lecture 4}}$ $\underbrace{E_{\text{in}}(g) \approx 0}_{\text{lecture 2}}$

learning split to two central questions:

- ① can we make sure that $E_{\text{out}}(g)$ is close enough to $E_{\text{in}}(g)$?
- ② can we make $E_{\text{in}}(g)$ small enough?

what role does $\underbrace{M}_{|\mathcal{H}|}$ play for the two questions?

Trade-off on M

- ① can we make sure that $E_{\text{out}}(g)$ is close enough to $E_{\text{in}}(g)$?
- ② can we make $E_{\text{in}}(g)$ small enough?

small M

- ① Yes!,
 $\mathbb{P}[\text{BAD}] \leq 2 \cdot M \cdot \exp(\dots)$
- ② No!, too few choices

large M

- ① No!,
 $\mathbb{P}[\text{BAD}] \leq 2 \cdot M \cdot \exp(\dots)$
- ② Yes!, many choices

using the right M (or \mathcal{H}) is important

$M = \infty$ doomed?

Preview

Known

$$\mathbb{P} [|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \leq 2 \cdot M \cdot \exp(-2\epsilon^2 N)$$

Todo

- establish **a finite quantity** that replaces M

$$\mathbb{P} [|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \stackrel{?}{\leq} 2 \cdot m_{\mathcal{H}} \cdot \exp(-2\epsilon^2 N)$$

- justify the feasibility of learning for infinite M
- study $m_{\mathcal{H}}$ to understand its trade-off for ‘right’ \mathcal{H} , just like M

mysterious PLA to be fully resolved
after 3 more lectures :-)

Fun Time

Data size: how large do we need?

One way to use the inequality

$$\mathbb{P} [|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \leq \underbrace{2 \cdot M \cdot \exp(-2\epsilon^2 N)}_{\delta}$$

is to pick a tolerable difference ϵ as well as a tolerable **BAD** probability δ , and then gather data with size (N) large enough to achieve those tolerance criteria. Let $\epsilon = 0.1$, $\delta = 0.05$, and $M = 100$. What is the data size needed?

1 215

2 415

3 615

4 815

Reference Answer: 2

We can simply express N as a function of those 'known' variables. Then, the needed $N = \frac{1}{2\epsilon^2} \ln \frac{2M}{\delta}$.

Where Did M Come From?

$$\mathbb{P} [|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \leq 2 \cdot M \cdot \exp(-2\epsilon^2 N)$$

- **BAD events \mathcal{B}_m :** $|E_{\text{in}}(h_m) - E_{\text{out}}(h_m)| > \epsilon$
- to give \mathcal{A} freedom of choice: bound $\mathbb{P}[\mathcal{B}_1 \text{ or } \mathcal{B}_2 \text{ or } \dots \mathcal{B}_M]$
- worst case: all \mathcal{B}_m non-overlapping

$$\mathbb{P}[\mathcal{B}_1 \text{ or } \mathcal{B}_2 \text{ or } \dots \mathcal{B}_M] \quad \underbrace{\quad}_{\text{union bound}} \quad \mathbb{P}[\mathcal{B}_1] + \mathbb{P}[\mathcal{B}_2] + \dots + \mathbb{P}[\mathcal{B}_M]$$

where did **uniform bound fail**
to consider for $M = \infty$?

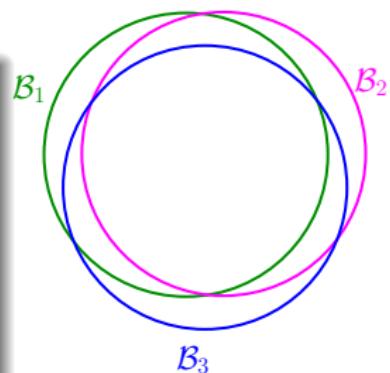
Where Did Uniform Bound Fail?

union bound $\mathbb{P}[\mathcal{B}_1] + \mathbb{P}[\mathcal{B}_2] + \dots + \mathbb{P}[\mathcal{B}_M]$

- **BAD events \mathcal{B}_m :** $|E_{\text{in}}(h_m) - E_{\text{out}}(h_m)| > \epsilon$

overlapping for similar hypotheses $h_1 \approx h_2$

- why?
 - 1 $E_{\text{out}}(h_1) \approx E_{\text{out}}(h_2)$
 - 2 for most \mathcal{D} , $E_{\text{in}}(h_1) = E_{\text{in}}(h_2)$
- union bound **over-estimating**

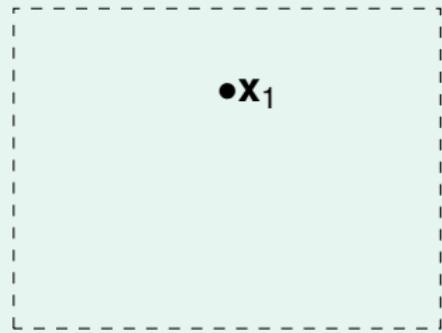


to account for overlap,
can we group similar hypotheses by **kind**?

How Many Lines Are There? (1/2)

$$\mathcal{H} = \left\{ \text{all lines in } \mathbb{R}^2 \right\}$$

- how many lines? ∞
- how many **kinds of** lines if viewed from one input vector \mathbf{x}_1 ?

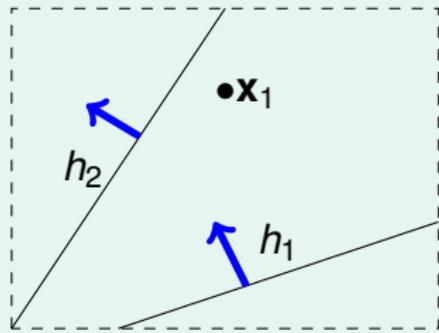


2 kinds: h_1 -like(\mathbf{x}_1) = o or h_2 -like(\mathbf{x}_1) = x

How Many Lines Are There? (1/2)

$$\mathcal{H} = \left\{ \text{all lines in } \mathbb{R}^2 \right\}$$

- how many lines? ∞
- how many **kinds of** lines if viewed from one input vector \mathbf{x}_1 ?

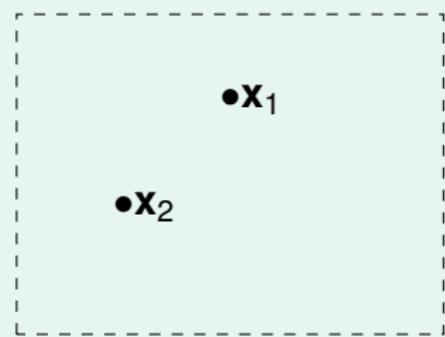


2 kinds: h_1 -like(\mathbf{x}_1) = o or h_2 -like(\mathbf{x}_1) = x

How Many Lines Are There? (2/2)

$$\mathcal{H} = \left\{ \text{all lines in } \mathbb{R}^2 \right\}$$

- how many **kinds of** lines if viewed from two inputs $\mathbf{x}_1, \mathbf{x}_2$?

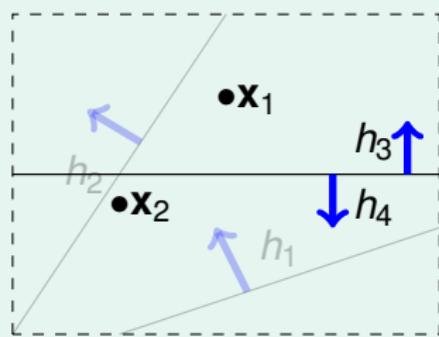


one input: 2; two inputs: 4; **three inputs?**

How Many Lines Are There? (2/2)

$$\mathcal{H} = \left\{ \text{all lines in } \mathbb{R}^2 \right\}$$

- how many **kinds of** lines if viewed from two inputs $\mathbf{x}_1, \mathbf{x}_2$?



4:

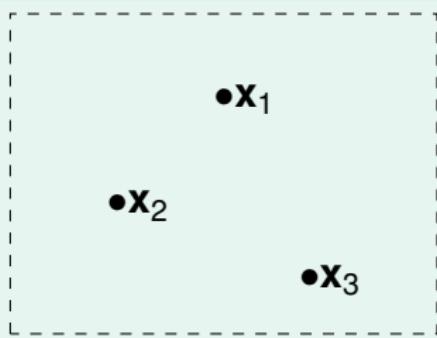


one input: 2; two inputs: 4; **three inputs?**

How Many Kinds of Lines for Three Inputs? (1/2)

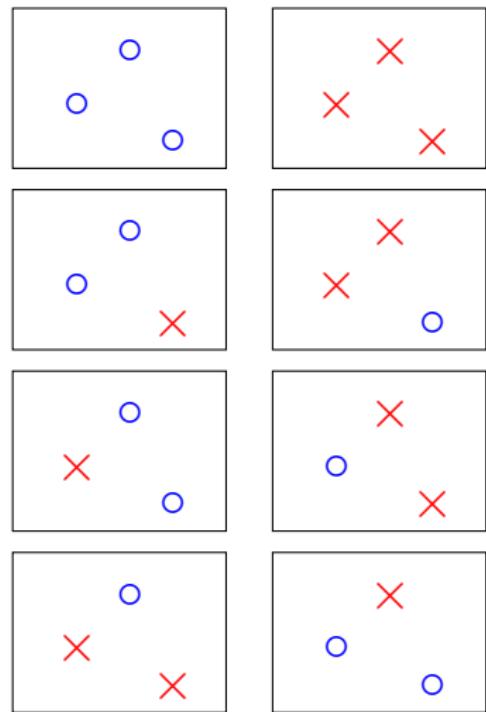
$$\mathcal{H} = \left\{ \text{all lines in } \mathbb{R}^2 \right\}$$

for three inputs $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$



always 8 for three inputs?

8:

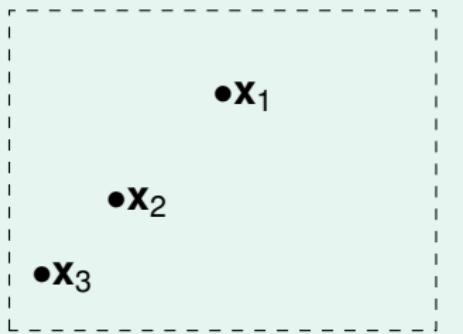


How Many Kinds of Lines for Three Inputs? (2/2)

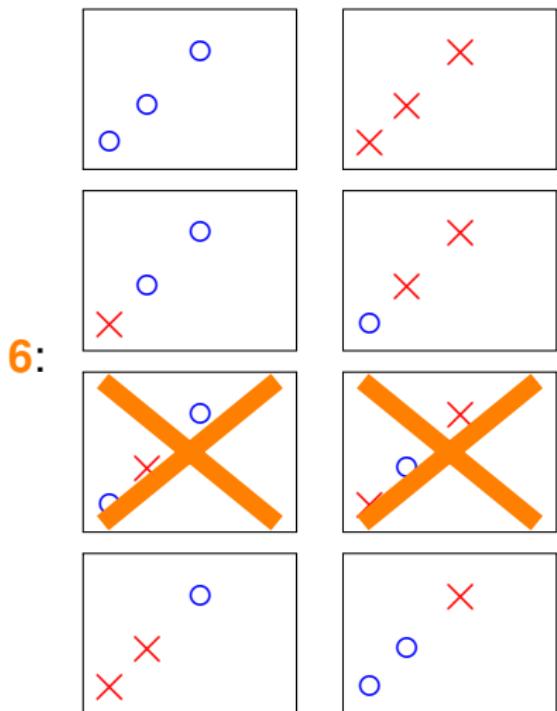
$$\mathcal{H} = \left\{ \text{all lines in } \mathbb{R}^2 \right\}$$

for **another** three inputs

$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$



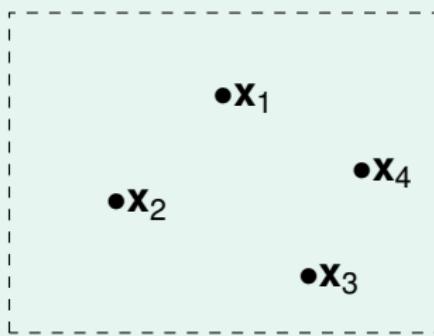
'fewer than 8' when degenerate
(e.g. collinear or same inputs)



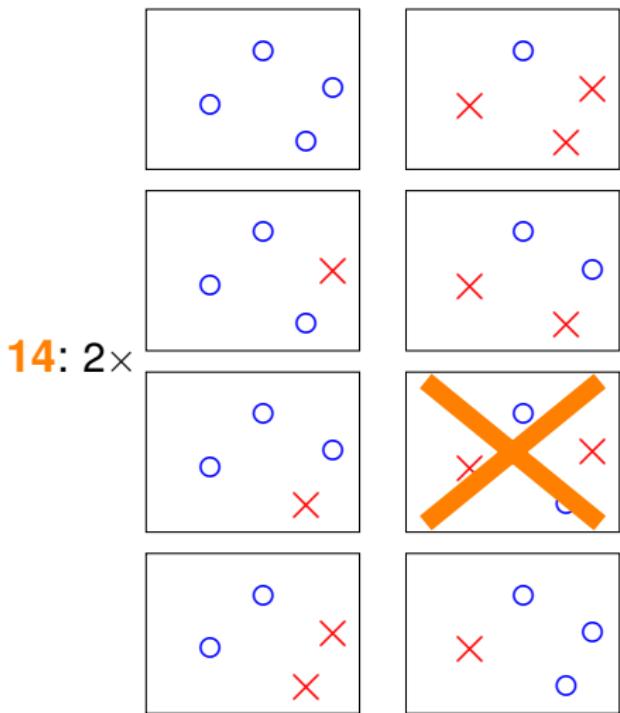
How Many Kinds of Lines for Four Inputs?

$$\mathcal{H} = \left\{ \text{all lines in } \mathbb{R}^2 \right\}$$

for four inputs $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$



for any four inputs
at most 14



Effective Number of Lines

maximum kinds of lines with respect to N inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
 \iff **effective number of lines**

- must be $\leq 2^N$ (why?)
- finite ‘grouping’ of infinitely-many lines $\in \mathcal{H}$
- wish:

$$\begin{aligned} & \mathbb{P} [|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \\ & \leq 2 \cdot \text{effective}(N) \cdot \exp(-2\epsilon^2 N) \end{aligned}$$

lines in 2D

N	$\text{effective}(N)$
1	2
2	4
3	8
4	14 $< 2^4$

- if **1** $\text{effective}(N)$ can replace M and
2 $\text{effective}(N) \ll 2^N$

learning possible with infinite lines :-)

Fun Time

What is the effective number of lines for five inputs $\in \mathbb{R}^2$?

1 14

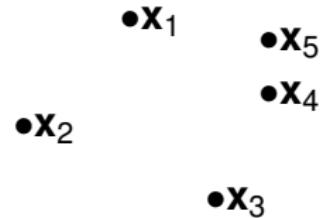
2 16

3 22

4 32

Reference Answer: 3

If you put the inputs roughly around a circle, you can then pick any consecutive inputs to be on one side of the line, and the other inputs to be on the other side. The procedure leads to effectively 22 kinds of lines, which is **much smaller than $2^5 = 32$** . You shall find it difficult to generate more kinds by varying the inputs, and we will give a formal proof in future lectures.



Dichotomies: Mini-hypotheses

$$\mathcal{H} = \{\text{hypothesis } h: \mathcal{X} \rightarrow \{\textcolor{red}{\times}, \textcolor{blue}{\circ}\}\}$$

- call

$$h(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = (h(\mathbf{x}_1), h(\mathbf{x}_2), \dots, h(\mathbf{x}_N)) \in \{\textcolor{red}{\times}, \textcolor{blue}{\circ}\}^N$$

a **dichotomy**: hypothesis ‘limited’ to the eyes of $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$

- $\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$:
- all dichotomies ‘implemented’ by \mathcal{H} on $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$**

	hypotheses \mathcal{H}	dichotomies $\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$
e.g.	all lines in \mathbb{R}^2	$\{\textcolor{blue}{oooo}, \textcolor{blue}{ooo}\textcolor{red}{x}, \textcolor{blue}{oo}\textcolor{red}{xx}, \dots\}$
size	possibly infinite	upper bounded by 2^N

$|\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)|$: candidate for **replacing M**

Growth Function

- $|\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)|$: depend on inputs $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$
- growth function:
remove dependence by **taking max of all possible** $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$

$$m_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in \mathcal{X}} |\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)|$$

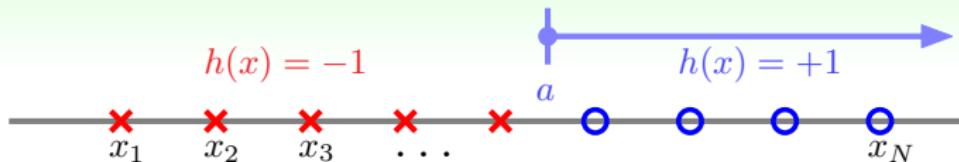
- finite, upper-bounded by 2^N

lines in 2D

N	$m_{\mathcal{H}}(N)$
1	2
2	4
3	$\max(\dots, 6, 8) = 8$
4	$14 < 2^4$

how to ‘calculate’ the growth function?

Growth Function for Positive Rays



- $\mathcal{X} = \mathbb{R}$ (one dimensional)
- \mathcal{H} contains h , where **each** $h(x) = \text{sign}(x - a)$ for threshold a
- ‘positive half’ of 1D perceptrons

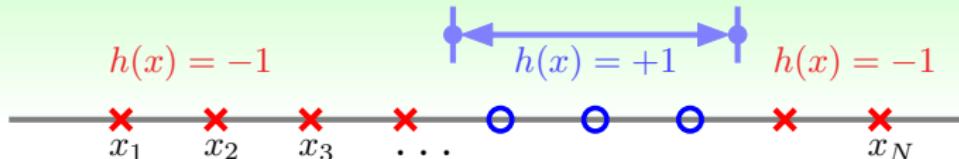
one dichotomy for $a \in$ each spot (x_n, x_{n+1}) :

$$m_{\mathcal{H}}(N) = N + 1$$

$(N + 1) \ll 2^N$ when N large!

x_1	x_2	x_3	x_4
○	○	○	○
×	○	○	○
×	×	○	○
×	×	×	○
×	×	×	×

Growth Function for Positive Intervals



- $\mathcal{X} = \mathbb{R}$ (one dimensional)
- \mathcal{H} contains h , where **each** $h(x) = +1$ iff $x \in [\ell, r]$, -1 otherwise

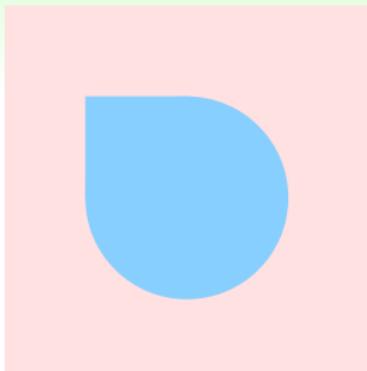
one dichotomy for each ‘interval kind’

$$\begin{aligned}
 m_{\mathcal{H}}(N) &= \underbrace{\binom{N+1}{2}}_{\text{interval ends in } N+1 \text{ spots}} + 1 \\
 &= \frac{1}{2}N^2 + \frac{1}{2}N + 1
 \end{aligned}$$

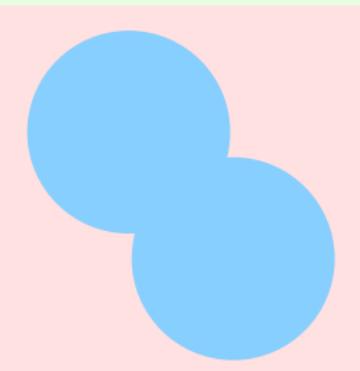
$(\frac{1}{2}N^2 + \frac{1}{2}N + 1) \ll 2^N$ when N large!

x_1	x_2	x_3	x_4
○	✗	✗	✗
○	○	✗	✗
○	○	○	✗
○	○	○	○
✗	○	✗	✗
✗	○	○	✗
✗	○	○	✗
✗	✗	○	○
✗	✗	○	○
✗	✗	✗	○
✗	✗	✗	✗

Growth Function for Convex Sets (1/2)



convex region in blue



non-convex region

- $\mathcal{X} = \mathbb{R}^2$ (two dimensional)
- \mathcal{H} contains h , where $h(\mathbf{x}) = +1$ iff \mathbf{x} in a **convex region**, -1 otherwise

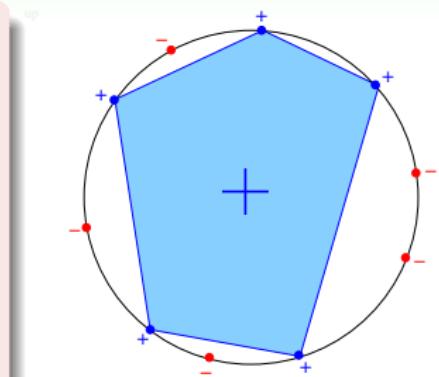
what is $m_{\mathcal{H}}(N)$?

Growth Function for Convex Sets (2/2)

- one possible set of N inputs:
 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ on a big circle
- **every dichotomy can be implemented** by \mathcal{H} using a convex region slightly extended from **contour of positive inputs**

$$m_{\mathcal{H}}(N) = 2^N$$

- call those N inputs '**shattered**' by \mathcal{H}



bottom

$m_{\mathcal{H}}(N) = 2^N \iff$
exists N inputs that can be shattered

Fun Time

Consider positive **and negative** rays as \mathcal{H} , which is equivalent to the perceptron hypothesis set in 1D. The hypothesis set is often called '**decision stump**' to describe the shape of its hypotheses. What is the growth function $m_{\mathcal{H}}(N)$?

① N

② $N + 1$

③ $2N$

④ 2^N

Reference Answer: ③

Two dichotomies when threshold in each of the $N - 1$ 'internal' spots; two dichotomies for the all- \circ and all- \times cases.

The Four Growth Functions

- positive rays:
- positive intervals:
- convex sets:
- 2D perceptrons:

$$m_{\mathcal{H}}(N) = N + 1$$

$$m_{\mathcal{H}}(N) = \frac{1}{2}N^2 + \frac{1}{2}N + 1$$

$$m_{\mathcal{H}}(N) = 2^N$$

$m_{\mathcal{H}}(N) < 2^N$ in some cases

what if $m_{\mathcal{H}}(N)$ replaces M ?

$$\mathbb{P} [|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \stackrel{?}{\leq} 2 \cdot m_{\mathcal{H}}(N) \cdot \exp(-2\epsilon^2 N)$$

polynomial: good; exponential: bad

for 2D or general perceptrons,
 $m_{\mathcal{H}}(N)$ polynomial?

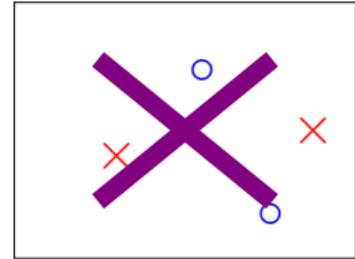
Break Point of \mathcal{H}

what do we know about 2D perceptrons now?

three inputs: 'exists' shatter;
four inputs, 'for all' no shatter

if no k inputs can be shattered by \mathcal{H} ,
call k a **break point** for \mathcal{H}

- $m_{\mathcal{H}}(k) < 2^k$
- $k + 1, k + 2, k + 3, \dots$ also break points!
- will study **minimum break point k**



2D perceptrons: **break point at 4**

The Four Break Points

- positive rays: $m_{\mathcal{H}}(N) = N + 1 = O(N)$
break point at 2
- positive intervals: $m_{\mathcal{H}}(N) = \frac{1}{2}N^2 + \frac{1}{2}N + 1 = O(N^2)$
break point at 3
- convex sets: $m_{\mathcal{H}}(N) = 2^N$
no break point
- 2D perceptrons: $m_{\mathcal{H}}(N) < 2^N$ in some cases
break point at 4

conjecture:

- no break point: $m_{\mathcal{H}}(N) = 2^N$ (sure!)
- break point k : $m_{\mathcal{H}}(N) = O(N^{k-1})$

excited? wait for next lecture :-)

Fun Time

Consider positive **and negative** rays as \mathcal{H} , which is equivalent to the perceptron hypothesis set in 1D. As discussed in an earlier quiz question, the growth function $m_{\mathcal{H}}(N) = 2N$. What is the minimum break point for \mathcal{H} ?

① 1

② 2

③ 3

④ 4

Reference Answer: ③

At $k = 3$, $m_{\mathcal{H}}(k) = 6$ while $2^k = 8$.

Summary

1 When Can Machines Learn?

Lecture 4: Feasibility of Learning

2 Why Can Machines Learn?

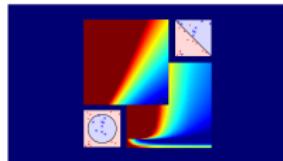
Lecture 5: Training versus Testing

- Recap and Preview
- two questions: $E_{\text{out}}(g) \approx E_{\text{in}}(g)$, and $E_{\text{in}}(g) \approx 0$
- Effective Number of Lines
 - at most 14 through the eye of 4 inputs
- Effective Number of Hypotheses
 - at most $m_{\mathcal{H}}(N)$ through the eye of N inputs
- Break Point
 - when $m_{\mathcal{H}}(N)$ becomes ‘non-exponential’
- next: $m_{\mathcal{H}}(N) = \text{poly}(N)?$

3 How Can Machines Learn?

4 How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 6: Theory of Generalization

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?

② Why Can Machines Learn?

Lecture 5: Training versus Testing

effective price of choice in training: (wishfully)
growth function $m_{\mathcal{H}}(N)$ with a break point

Lecture 6: Theory of Generalization

- Restriction of Break Point
- Bounding Function: Basic Cases
- Bounding Function: Inductive Cases
- A Pictorial Proof

③ How Can Machines Learn?

④ How Can Machines Learn Better?

The Four Break Points

growth function $m_{\mathcal{H}}(N)$: max number of dichotomies

- positive rays: $m_{\mathcal{H}}(N) = N + 1$
 $m_{\mathcal{H}}(2) = 3 < 2^2$: break point at 2
- positive intervals: $m_{\mathcal{H}}(N) = \frac{1}{2}N^2 + \frac{1}{2}N + 1$
 $m_{\mathcal{H}}(3) = 7 < 2^3$: break point at 3
- convex sets: $m_{\mathcal{H}}(N) = 2^N$

- 2D perceptrons: $m_{\mathcal{H}}(N) < 2^N$ in some cases


break point $k \implies$ break point $k + 1, \dots$
what else?

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

1 dichotomy , shatter any two points? **no**

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
○	○	○

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

2 dichotomies , shatter any two points? **no**

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
○	○	○
○	○	✗

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

3 dichotomies , shatter any two points? **no**

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
○	○	○
○	○	✗
○	✗	○

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

4 dichotomies , shatter any two points? **yes**

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
○	○	○
○	○	✗
○	✗	○
○	✗	✗

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

4 dichotomies , shatter any two points? **no**

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
○	○	○
○	○	✗
○	✗	○
✗	○	○

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

5 dichotomies , shatter any two points? **yes**

x_1	x_2	x_3
o	o	o
o	o	x
o	x	o
x	o	o
x	o	x

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

5 dichotomies , shatter any two points? **yes**

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
o	o	o
o	o	x
o	x	o
x	o	o
x	x	o

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

5 dichotomies , shatter any two points? **yes**

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
o	o	o
o	o	x
o	x	o
x	o	o
x	x	x

Restriction of Break Point (1/2)

what 'must be true' when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)

maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$ and $k = 2$?

maximum possible so far: **4 dichotomies**

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
○	○	○
○	○	✗
○	✗	○
✗	○	○
<hr/>		
:-)	:-)	:-)

Restriction of Break Point (2/2)

what ‘must be true’ when **minimum break point $k = 2$**

- $N = 1$: every $m_{\mathcal{H}}(N) = 2$ by definition
- $N = 2$: every $m_{\mathcal{H}}(N) < 4$ by definition
(so **maximum possible = 3**)
- $N = 3$: **maximum possible = 4** $\ll 2^3$

—break point k **restricts maximum possible $m_{\mathcal{H}}(N)$ a lot** for $N > k$

idea: $m_{\mathcal{H}}(N)$

\leq **maximum possible $m_{\mathcal{H}}(N)$ given k**

\leq **$poly(N)$**

Fun Time

When minimum break point $k = 1$, what is the maximum possible $m_{\mathcal{H}}(N)$ when $N = 3$?

1 1

2 2

3 4

4 8

Reference Answer: 1

Because $k = 1$, the hypothesis set cannot even shatter one point. Thus, every ‘column’ of the table cannot contain both \circ and \times . Then, after including the first dichotomy, it is not possible to include any other different dichotomy. Thus, the maximum possible $m_{\mathcal{H}}(N)$ is 1.

x_1	x_2	x_3
\circ	\times	\circ
\circ	\times	\times

Bounding Function

bounding function $B(N, k)$:

maximum possible $m_{\mathcal{H}}(N)$ when break point = k

- combinatorial quantity:
maximum number of length- N vectors with (o, x)
while '**no shatter**' any length- k subvectors
- irrelevant of the details of \mathcal{H}
e.g. $B(N, 3)$ bounds both
 - positive intervals ($k = 3$)
 - 1D perceptrons ($k = 3$)

new goal: $B(N, k) \leq \text{poly}(N) ?$

Table of Bounding Function (1/4)

		k						
		1	2	3	4	5	6	...
N	1							
	2		3					
	3		4					
	4							
	5							
	6							
	:							

Known

- $B(2, 2) = 3$ (maximum < 4)
- $B(3, 2) = 4$ ('pictorial' proof previously)

Table of Bounding Function (2/4)

		k						
		1	2	3	4	5	6	...
N	1	1						
	2	1	3					
	3	1	4					
	4	1						
	5	1						
	6	1						
	:	:						

Known

- $B(N, 1) = 1$ (see previous quiz)

Table of Bounding Function (3/4)

		k						
		1	2	3	4	5	6	...
N	1	1	2	2	2	2	2	...
	2	1	3	4	4	4	4	...
	3	1	4		8	8	8	...
	4	1				16	16	...
	5	1					32	...
	6	1						...
	:	:						

Known

- $B(N, k) = 2^N$ for $N < k$
—including all dichotomies not violating ‘breaking condition’

Table of Bounding Function (4/4)

		k						
		1	2	3	4	5	6	...
$B(N, k)$		1	2	2	2	2	2	...
N	1	1	2	2	2	2	2	...
	2	1	3	4	4	4	4	...
	3	1	4	7	8	8	8	...
	4	1			15	16	16	...
	5	1				31	32	...
	6	1					63	...
	:	:						...

Known

- $B(N, k) = 2^N - 1$ for $N = k$
 - removing a single dichotomy satisfies ‘breaking condition’

more than halfway done! :-)

Fun Time

For the 2D perceptrons, which of the following claim is true?

- ① minimum break point $k = 2$
- ② $m_{\mathcal{H}}(4) = 15$
- ③ $m_{\mathcal{H}}(N) < B(N, k)$ when $N = k = \text{minimum break point}$
- ④ $m_{\mathcal{H}}(N) > B(N, k)$ when $N = k = \text{minimum break point}$

Reference Answer: ③

As discussed previously, minimum break point for 2D perceptrons is 4, with $m_{\mathcal{H}}(4) = 14$. Also, note that $B(4, 4) = 15$. So bounding function $B(N, k)$ can be ‘loose’ in bounding $m_{\mathcal{H}}(N)$.

Estimating $B(4, 3)$

		k						
		1	2	3	4	5	6	...
1		1	2	2	2	2	2	...
2		1	3	4	4	4	4	...
3		1	4	7	8	8	8	...
N	4	1		?	15	16	16	...
	5	1			31	32	...	
	6	1				63	...	
	:	:					...	

Motivation

- $B(4, 3)$ shall be related to $B(3, ?)$
—‘adding’ one point from $B(3, ?)$

next: reduce $B(4, 3)$ to $B(3, ?)$

'Achieving' Dichotomies of $B(4, 3)$

after checking all 2^{2^4} sets of dichotomies, **the winner is ...**

	x_1	x_2	x_3	x_4
01	○	○	○	○
02	✗	○	○	○
03	○	✗	○	○
04	○	○	✗	○
05	○	○	○	✗
06	✗	✗	○	✗
07	✗	○	✗	○
08	✗	○	○	✗
09	○	✗	✗	○
10	○	✗	○	✗
11	○	○	✗	✗

	$B(N, k)$	1	2	3	4	5	6
	1	1	2	2	2	2	2
	2	1	3	4	4	4	4
	3	1	4	7	8	8	8
N	4	1		11	15	16	16
	5	1			31	32	
	6	1				63	

how to reduce $B(4, 3)$ to $B(3, ?)$ cases?

Reorganized Dichotomies of $B(4, 3)$

after checking all 2^{2^4} sets of dichotomies, **the winner is ...**

	x_1	x_2	x_3	x_4
01	o	o	o	o
02	x	o	o	o
03	o	x	o	o
04	o	o	x	o
05	o	o	o	x
06	x	x	o	x
07	x	o	x	o
08	x	o	o	x
09	o	x	x	o
10	o	x	o	x
11	o	o	x	x



	x_1	x_2	x_3	x_4
01	o	o	o	o
05	o	o	o	x
02	x	o	o	o
08	x	o	o	x
03	o	x	o	o
10	o	x	o	x
04	o	o	x	o
11	o	o	x	x
06	x	x	o	x
07	x	o	x	o
09	o	x	x	o

orange: pair; purple: single

Estimating Part of $B(4, 3)$ (1/2)

$$B(4, 3) = 11 = 2\alpha + \beta$$

	\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	
α	o	o	o	
	x	o	o	
	o	x	o	
	o	o	x	
β	x	x	o	
	x	o	x	
	o	x	x	

- $\alpha + \beta$: dichotomies on $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$
- $B(4, 3)$ ‘no shatter’ any 3 inputs
 $\implies \alpha + \beta$ ‘no shatter’ any 3

	\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	\mathbf{x}_4
2α	o	o	o	o
	o	o	o	x
	x	o	o	o
	x	o	o	x
β	o	x	o	o
	o	x	o	x
	o	o	x	o

	\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	\mathbf{x}_4
β	x	x	o	x
	x	o	x	o
	o	x	x	o

$$\alpha + \beta \leq B(3, 3)$$

Estimating Part of $B(4, 3)$ (2/2)

$$B(4, 3) = 11 = 2\alpha + \beta$$

	\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	
α	o	o	o	
	x	o	o	
	o	x	o	
	o	o	x	

- α : dichotomies on $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ with \mathbf{x}_4 paired
- $B(4, 3)$ ‘no shatter’ any 3 inputs
 $\implies \alpha$ ‘no shatter’ any 2

	\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	\mathbf{x}_4
2α	o	o	o	o
	o	o	o	x
	x	o	o	o
	x	o	o	x
	o	x	o	o
	o	x	o	x
	o	o	x	o
	o	o	x	x
β	x	x	o	x
	x	o	x	o
	o	x	x	o

$$\alpha \leq B(3, 2)$$

Putting It All Together

$$B(4, 3) = 2\alpha + \beta$$

$$\alpha + \beta \leq B(3, 3)$$

$$\alpha \leq B(3, 2)$$

$$\Rightarrow B(4, 3) \leq B(3, 3) + B(3, 2)$$

		k					
		1	2	3	4	5	6
B(N, k)		1	2	3	4	5	6
N	1	1	2	2	2	2	2
	2	1	3	4	4	4	4
	3	1	4	7	8	8	8
	4	1	≤ 5	11	15	16	16
	5	1	≤ 6	≤ 16	≤ 26	31	32
	6	1	≤ 7	≤ 22	≤ 42	≤ 57	63

now have **upper bound** of bounding function

Putting It All Together

$$B(N, k) = 2\alpha + \beta$$

$$\alpha + \beta \leq B(N - 1, k)$$

$$\alpha \leq B(N - 1, k - 1)$$

$$\Rightarrow B(N, k) \leq B(N - 1, k) + B(N - 1, k - 1)$$

		k					
		1	2	3	4	5	6
B(N, k)		1	2	3	4	5	6
N	1	1	2	2	2	2	2
	2	1	3	4	4	4	4
	3	1	4	7	8	8	8
	4	1	≤ 5	11	15	16	16
	5	1	≤ 6	≤ 16	≤ 26	31	32
	6	1	≤ 7	≤ 22	≤ 42	≤ 57	63

now have **upper bound** of bounding function

Bounding Function: The Theorem

$$B(N, k) \leq \underbrace{\sum_{i=0}^{k-1} \binom{N}{i}}_{\text{highest term } N^{k-1}}$$

- simple induction using **boundary and inductive formula**
- for fixed k , $B(N, k)$ upper bounded by $\text{poly}(N)$
 $\implies m_{\mathcal{H}}(N)$ is $\text{poly}(N)$ if break point exists

‘ \leq ’ can be ‘ $=$ ’ actually,

go play and prove it if math lover! :-)

The Three Break Points

$$B(N, k) \leq \underbrace{\sum_{i=0}^{k-1} \binom{N}{i}}_{\text{highest term } N^{k-1}}$$

- positive rays: $m_{\mathcal{H}}(N) = N + 1 \leq N + 1$
o x $m_{\mathcal{H}}(2) = 3 < 2^2$: break point at 2
- positive intervals: $m_{\mathcal{H}}(N) = \frac{1}{2}N^2 + \frac{1}{2}N + 1 \leq \frac{1}{2}N^2 + \frac{1}{2}N + 1$
o x o $m_{\mathcal{H}}(3) = 7 < 2^3$: break point at 3
- 2D perceptrons: $m_{\mathcal{H}}(N) = ? \leq \frac{1}{6}N^3 + \frac{5}{6}N + 1$
x o x o $m_{\mathcal{H}}(4) = 14 < 2^4$: break point at 4

can bound $m_{\mathcal{H}}(N)$ by only one break point

Fun Time

For 1D perceptrons (positive and negative rays), we know that $m_{\mathcal{H}}(N) = 2N$. Let k be the minimum break point. Which of the following is not true?

- ① $k = 3$
- ② for some integers $N > 0$, $m_{\mathcal{H}}(N) = \sum_{i=0}^{k-1} \binom{N}{i}$
- ③ for all integers $N > 0$, $m_{\mathcal{H}}(N) = \sum_{i=0}^{k-1} \binom{N}{i}$
- ④ for all integers $N > 2$, $m_{\mathcal{H}}(N) < \sum_{i=0}^{k-1} \binom{N}{i}$

Reference Answer: ③

The proof is generally trivial by listing the definitions. For ②, $N = 1$ or 2 gives the equality. One thing to notice is ④: the upper bound can be ‘loose’.

BAD Bound for General \mathcal{H}

want:

$$\mathbb{P}\left[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon\right] \leq 2 \cdot m_{\mathcal{H}}(\textcolor{orange}{N}) \cdot \exp\left(-2 \cdot \frac{\epsilon^2 N}{\epsilon^2 N}\right)$$

actually, when N large enough,

$$\mathbb{P}\left[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon\right] \leq 2 \cdot 2m_{\mathcal{H}}(2N) \cdot \exp\left(-2 \cdot \frac{1}{16} \epsilon^2 N\right)$$

next: **sketch** of proof

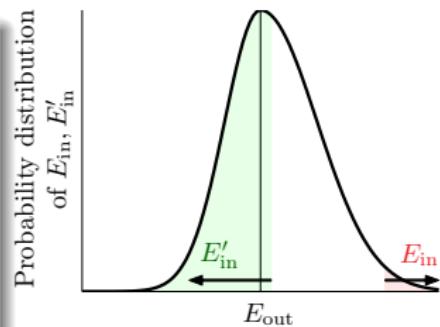
Step 1: Replace E_{out} by E'_{in}

$$\frac{1}{2} \mathbb{P} \left[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon \right]$$

\wedge

$$\mathbb{P} \left[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E'_{\text{in}}(h)| > \frac{\epsilon}{2} \right]$$

- $E_{\text{in}}(h)$ finitely many, $E_{\text{out}}(h)$ infinitely many
—replace the evil E_{out} first
- how? sample verification set \mathcal{D}' of size N to calculate E'_{in}
- BAD h of $E_{\text{in}} - E_{\text{out}}$
probably \Rightarrow BAD h of $E_{\text{in}} - E'_{\text{in}}$

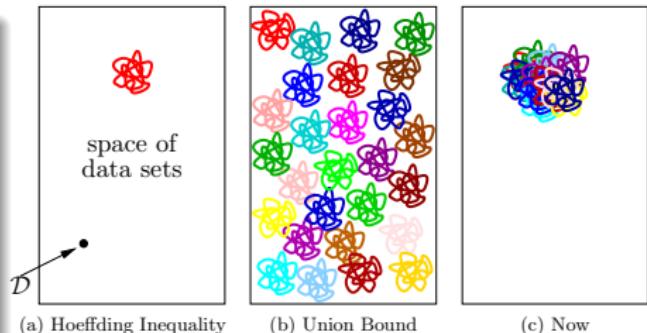


evil E_{out} removed by
 verification with ‘ghost data’

Step 2: Decompose \mathcal{H} by Kind

$$\begin{aligned} \text{BAD} &\leq 2\mathbb{P}\left[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E'_{\text{in}}(h)| > \frac{\epsilon}{2}\right] \\ &\leq 2m_{\mathcal{H}}(2N)\mathbb{P}\left[\text{fixed } h \text{ s.t. } |E_{\text{in}}(h) - E'_{\text{in}}(h)| > \frac{\epsilon}{2}\right] \end{aligned}$$

- E_{in} with \mathcal{D} , E'_{in} with \mathcal{D}'
—now $m_{\mathcal{H}}$ comes to play
- how? infinite \mathcal{H} becomes
 $|\mathcal{H}(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{x}'_1, \dots, \mathbf{x}'_N)|$ kinds
- union bound on $m_{\mathcal{H}}(2N)$ kinds

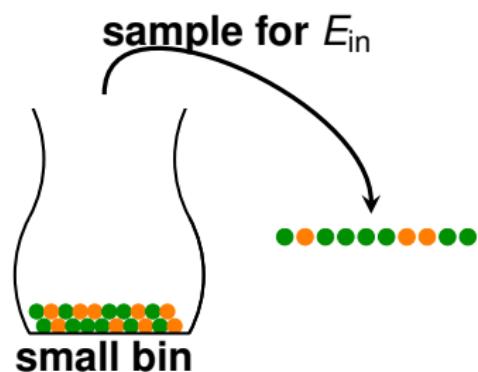


use $m_{\mathcal{H}}(2N)$ to calculate BAD-overlap properly

Step 3: Use Hoeffding without Replacement

$$\begin{aligned} \text{BAD} &\leq 2m_{\mathcal{H}}(2N)\mathbb{P}\left[\text{fixed } h \text{ s.t. } |E_{\text{in}}(h) - E'_{\text{in}}(h)| > \frac{\epsilon}{2}\right] \\ &\leq 2m_{\mathcal{H}}(2N) \cdot 2 \exp\left(-2\left(\frac{\epsilon}{4}\right)^2 N\right) \end{aligned}$$

- consider bin of $2N$ examples, choose N for E_{in} , leave others for E'_{in}
 $|E_{\text{in}} - E'_{\text{in}}| > \frac{\epsilon}{2} \Leftrightarrow \left|E_{\text{in}} - \frac{E_{\text{in}} + E'_{\text{in}}}{2}\right| > \frac{\epsilon}{4}$
- so? just ‘smaller bin’, ‘smaller ϵ ’, and Hoeffding without replacement



use Hoeffding after zooming to fixed h

That's All!

Vapnik-Chervonenkis (VC) bound:

$$\begin{aligned} & \mathbb{P}\left[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon\right] \\ & \leq 4m_{\mathcal{H}}(2N) \exp\left(-\frac{1}{8}\epsilon^2 N\right) \end{aligned}$$

- replace E_{out} by E'_{in}
- decompose \mathcal{H} by kind
- use Hoeffding without replacement

2D perceptrons:

- break point? 4
- $m_{\mathcal{H}}(N)$? $O(N^3)$

learning with 2D perceptrons feasible! :-)

Fun Time

For positive rays, $m_{\mathcal{H}}(N) = N + 1$. Plug it into the VC bound for $\epsilon = 0.1$ and $N = 10000$. What is VC bound of BAD events?

$$\mathbb{P} \left[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon \right] \leq 4m_{\mathcal{H}}(2N) \exp \left(-\frac{1}{8}\epsilon^2 N \right)$$

- ① 2.77×10^{-87}
- ② 5.54×10^{-83}
- ③ 2.98×10^{-1}
- ④ 2.29×10^2

Reference Answer: ③

Simple calculation. Note that the BAD probability bound is not very small even with 10000 examples.

Summary

① When Can Machines Learn?

② Why Can Machines Learn?

Lecture 5: Training versus Testing

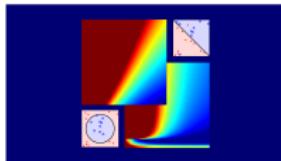
Lecture 6: Theory of Generalization

- Restriction of Break Point
break point ‘breaks’ consequent points
 - Bounding Function: Basic Cases
 $B(N, k)$ bounds $m_{\mathcal{H}}(N)$ with break point k
 - Bounding Function: Inductive Cases
 $B(N, k)$ is $\text{poly}(N)$
 - A Pictorial Proof
 $m_{\mathcal{H}}(N)$ can replace M with a few changes
- next: how to ‘use’ the break point?

③ How Can Machines Learn?

④ How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 7: The VC Dimension

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?

② Why Can Machines Learn?

Lecture 6: Theory of Generalization

$E_{\text{out}} \approx E_{\text{in}}$ possible

if $m_{\mathcal{H}}(N)$ breaks somewhere and N large enough

Lecture 7: The VC Dimension

- Definition of VC Dimension
- VC Dimension of Perceptrons
- Physical Intuition of VC Dimension
- Interpreting VC Dimension

③ How Can Machines Learn?

④ How Can Machines Learn Better?

Recap: More on Growth Function

$$m_{\mathcal{H}}(N) \text{ of break point } k \leq B(N, k) = \underbrace{\sum_{i=0}^{k-1} \binom{N}{i}}_{\text{highest term } N^{k-1}}$$

	k				
$B(N, k)$	1	2	3	4	5
N	1	1	2	2	2
	2	1	3	4	4
	3	1	4	7	8
	4	1	5	11	15
	5	1	6	16	26
	6	1	7	22	42

	k				
N^{k-1}	1	2	3	4	5
1	1	1	1	1	1
	2	1	2	4	8
	3	1	3	9	27
	4	1	4	16	64
	5	1	5	25	125
	6	1	6	36	216

provably & loosely, for $N \geq 2, k \geq 3$,

$$m_{\mathcal{H}}(N) \leq B(N, k) = \sum_{i=0}^{k-1} \binom{N}{i} \leq N^{k-1}$$

Recap: More on Vapnik-Chervonenkis (VC) Bound

For any $\mathbf{g} = \mathcal{A}(\mathcal{D}) \in \mathcal{H}$ and 'statistical' large \mathcal{D} , for $N \geq 2, k \geq 3$

$$\begin{aligned} & \mathbb{P}_{\mathcal{D}} \left[|E_{\text{in}}(\mathbf{g}) - E_{\text{out}}(\mathbf{g})| > \epsilon \right] \\ & \leq \mathbb{P}_{\mathcal{D}} \left[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon \right] \\ & \leq 4m_{\mathcal{H}}(2N) \exp \left(-\frac{1}{8}\epsilon^2 N \right) \\ & \underset{\text{if } k \text{ exists}}{\leq} 4(2N)^{k-1} \exp \left(-\frac{1}{8}\epsilon^2 N \right) \end{aligned}$$

- if (1) $m_{\mathcal{H}}(N)$ breaks at k (**good** \mathcal{H})
 - (2) N large enough (**good** \mathcal{D})
- \implies probably generalized ' $E_{\text{out}} \approx E_{\text{in}}$ ', and
- if (3) \mathcal{A} picks a g with small E_{in} (**good** \mathcal{A})
- \implies probably learned! ($\text{:-)} \text{ good luck}$)

VC Dimension

the formal name of **maximum non-break point**

Definition

VC dimension of \mathcal{H} , denoted $d_{\text{VC}}(\mathcal{H})$ is

largest N for which $m_{\mathcal{H}}(N) = 2^N$

- the **most** inputs \mathcal{H} that can shatter
- $d_{\text{VC}} = \text{'minimum } k\text{' - 1}$

$N \leq d_{\text{VC}}$ $\implies \mathcal{H}$ can shatter some N inputs

$k > d_{\text{VC}}$ $\implies k$ is a break point for \mathcal{H}

if $N \geq 2, d_{\text{VC}} \geq 2, m_{\mathcal{H}}(N) \leq N^{d_{\text{VC}}}$

The Four VC Dimensions

- positive rays:

$$d_{VC} = 1$$



$$m_{\mathcal{H}}(N) = N + 1$$

- positive intervals:

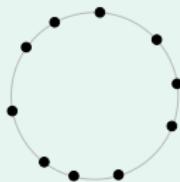
$$d_{VC} = 2$$



$$m_{\mathcal{H}}(N) = \frac{1}{2}N^2 + \frac{1}{2}N + 1$$

- convex sets:

$$d_{VC} = \infty$$



$$m_{\mathcal{H}}(N) = 2^N$$

- 2D perceptrons:

$$d_{VC} = 3$$



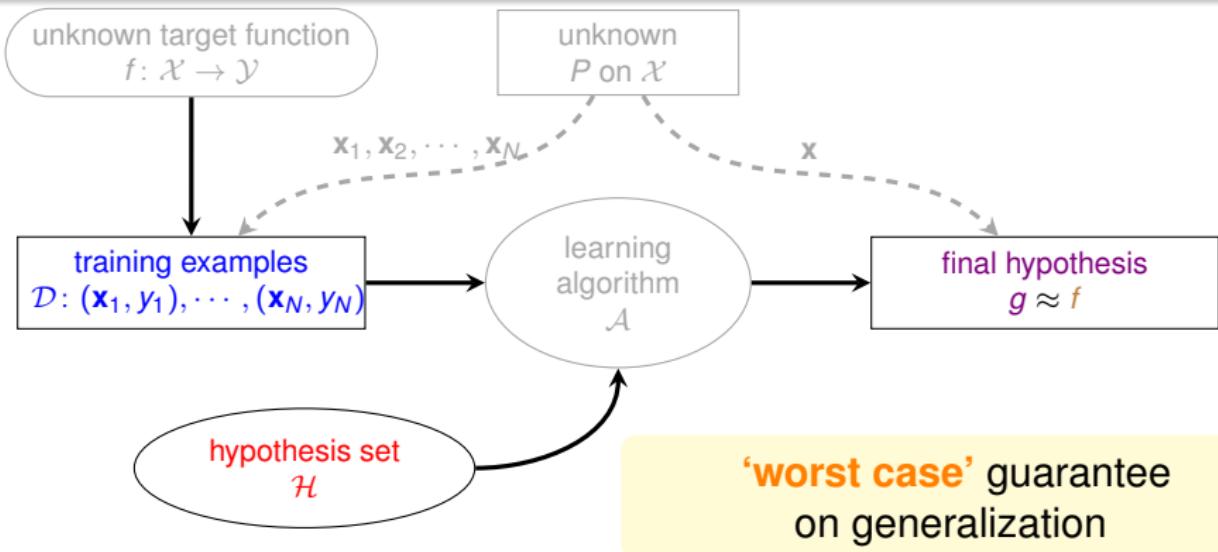
$$m_{\mathcal{H}}(N) \leq N^3 \text{ for } N \geq 2$$

good: **finite d_{VC}**

VC Dimension and Learning

finite $d_{VC} \Rightarrow g$ ‘will’ generalize ($E_{out}(g) \approx E_{in}(g)$)

- regardless of learning algorithm \mathcal{A}
- regardless of input distribution P
- regardless of target function f



Fun Time

If there is a set of N inputs that cannot be shattered by \mathcal{H} . Based only on this information, what can we conclude about $d_{\text{VC}}(\mathcal{H})$?

- ① $d_{\text{VC}}(\mathcal{H}) > N$
- ② $d_{\text{VC}}(\mathcal{H}) = N$
- ③ $d_{\text{VC}}(\mathcal{H}) < N$
- ④ no conclusion can be made

Reference Answer: ④

It is possible that there is another set of N inputs that can be shattered, which means $d_{\text{VC}} \geq N$. It is also possible that no set of N input can be shattered, which means $d_{\text{VC}} < N$. Neither cases can be ruled out by one non-shattering set.

2D PLA Revisited

linearly separable \mathcal{D}



PLA can converge

with $\mathbf{x}_n \sim P$ and $y_n = f(\mathbf{x}_n)$



$$\mathbb{P}[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \leq \dots \text{ by } d_{\text{VC}} = 3$$

T large

$$E_{\text{in}}(g) = 0$$

N large

$$E_{\text{out}}(g) \approx E_{\text{in}}(g)$$

$$E_{\text{out}}(g) \approx 0 \therefore$$

general PLA for \mathbf{x} with more than 2 features?

VC Dimension of Perceptrons

- 1D perceptron (pos/neg rays): $d_{VC} = 2$
- 2D perceptrons: $d_{VC} = 3$
 - $d_{VC} \geq 3$: 
 - $d_{VC} \leq 3$: 
- d -D perceptrons: $d_{VC} \stackrel{?}{=} d + 1$

two steps:

- $d_{VC} \geq d + 1$
- $d_{VC} \leq d + 1$

Extra Fun Time

What statement below shows that $d_{VC} \geq d + 1$?

- ① There are some $d + 1$ inputs we can shatter.
- ② We can shatter any set of $d + 1$ inputs.
- ③ There are some $d + 2$ inputs we cannot shatter.
- ④ We cannot shatter any set of $d + 2$ inputs.

Reference Answer: ①

d_{VC} is the maximum that $m_{\mathcal{H}}(N) = 2^N$, and $m_{\mathcal{H}}(N)$ is the most number of dichotomies of N inputs. So if we can find 2^{d+1} dichotomies on *some* $d + 1$ inputs, $m_{\mathcal{H}}(d + 1) = 2^{d+1}$ and hence $d_{VC} \geq d + 1$.

$$d_{\text{VC}} \geq d + 1$$

There are **some $d + 1$ inputs** we can shatter.

- some ‘trivial’ inputs:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \vdots \\ \mathbf{x}_{d+1}^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & 0 \\ 1 & 0 & \dots & 0 & 1 \end{bmatrix}$$

- visually in 2D:



note: **X invertible!**

Can We Shatter X?

$$X = \begin{bmatrix} -\mathbf{x}_1^T- \\ -\mathbf{x}_2^T- \\ \vdots \\ -\mathbf{x}_{d+1}^T- \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 1 & 0 & \dots & 0 & 1 \end{bmatrix} \text{ invertible}$$

to shatter ...

for any $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_{d+1} \end{bmatrix}$, find \mathbf{w} such that

$$\text{sign}(X\mathbf{w}) = \mathbf{y} \iff (X\mathbf{w}) = \mathbf{y} \xrightarrow{X \text{ invertible!}} \mathbf{w} = X^{-1}\mathbf{y}$$

'special' X can be shattered $\Rightarrow d_{VC} \geq d + 1$

Extra Fun Time

What statement below shows that $d_{VC} \leq d + 1$?

- ① There are some $d + 1$ inputs we can shatter.
- ② We can shatter any set of $d + 1$ inputs.
- ③ There are some $d + 2$ inputs we cannot shatter.
- ④ We cannot shatter any set of $d + 2$ inputs.

Reference Answer: ④

d_{VC} is the maximum that $m_{\mathcal{H}}(N) = 2^N$, and $m_{\mathcal{H}}(N)$ is the most number of dichotomies of N inputs. So if we cannot find 2^{d+2} dichotomies on *any* $d + 2$ inputs (i.e. break point), $m_{\mathcal{H}}(d + 2) < 2^{d+2}$ and hence $d_{VC} < d + 2$. That is, $d_{VC} \leq d + 1$.

$$d_{VC} \leq d + 1 \quad (1/2)$$

A 2D Special Case

$$\begin{matrix} \vdots & \vdots \\ \bullet & \bullet \end{matrix} \quad X = \left[\begin{array}{c} \text{--- } \mathbf{x}_1^T \text{ ---} \\ \text{--- } \mathbf{x}_2^T \text{ ---} \\ \text{--- } \mathbf{x}_3^T \text{ ---} \\ \text{--- } \mathbf{x}_4^T \text{ ---} \end{array} \right] = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array} \right]$$

○ ?
✗ ○

? cannot be ✗

$$\mathbf{w}^T \mathbf{x}_4 = \underbrace{\mathbf{w}^T \mathbf{x}_2}_{\textcircled{o}} + \underbrace{\mathbf{w}^T \mathbf{x}_3}_{\textcircled{o}} - \underbrace{\mathbf{w}^T \mathbf{x}_1}_{\texttimes} > 0$$

linear dependence **restricts dichotomy**

$$d_{VC} \leq d + 1 \quad (2/2)$$

d-D General Case

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_{d+1}^T \\ \mathbf{x}_{d+2}^T \end{bmatrix}$$

more rows than columns:

linear dependence (some a_i non-zero)

$$\mathbf{x}_{d+2} = a_1 \mathbf{x}_1 + a_2 \mathbf{x}_2 + \dots + a_{d+1} \mathbf{x}_{d+1}$$

- can you generate $(\text{sign}(a_1), \text{sign}(a_2), \dots, \text{sign}(a_{d+1}), \times)$? if so, what \mathbf{w} ?

$$\begin{aligned} \mathbf{w}^T \mathbf{x}_{d+2} &= a_1 \underbrace{\mathbf{w}^T \mathbf{x}_1}_\circ + a_2 \underbrace{\mathbf{w}^T \mathbf{x}_2}_\times + \dots + a_{d+1} \underbrace{\mathbf{w}^T \mathbf{x}_{d+1}}_\times \\ &> 0 \text{(contradiction!)} \end{aligned}$$

'general' \mathbf{X} no-shatter $\implies d_{VC} \leq d + 1$

Fun Time

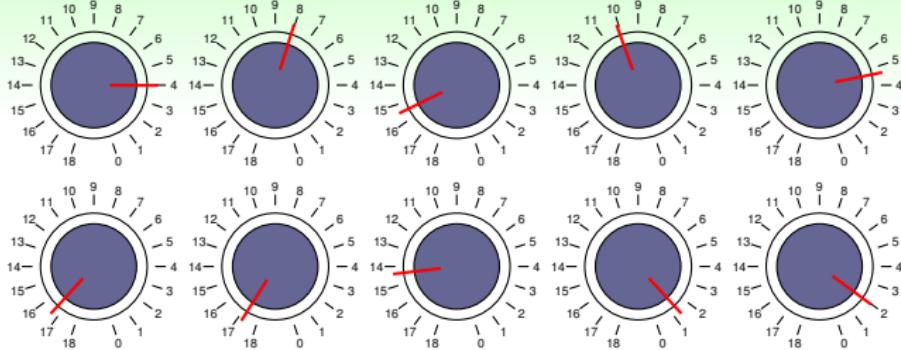
Based on the proof above, what is d_{VC} of 1126-D perceptrons?

- ① 1024
- ② 1126
- ③ 1127
- ④ 6211

Reference Answer: ③

Well, **too much fun for this section! :-)**

Degrees of Freedom



(modified from the work of Hugues Vermeiren on <http://www.texample.net>)

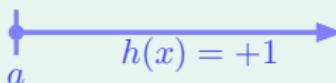
- hypothesis parameters $\mathbf{w} = (w_0, w_1, \dots, w_d)$: **creates degrees of freedom**
- hypothesis quantity $M = |\mathcal{H}|$: ‘analog’ degrees of freedom
- hypothesis ‘power’ $d_{VC} = d + 1$: **effective ‘binary’ degrees of freedom**

$d_{VC}(\mathcal{H})$: **powerfulness** of \mathcal{H}

Two Old Friends

Positive Rays ($d_{VC} = 1$)

$$h(x) = -1$$



free parameters: a

Positive Intervals ($d_{VC} = 2$)

$$h(x) = -1$$



free parameters: ℓ, r

practical rule of thumb:

$d_{VC} \approx \#\text{free parameters}$ (but not always)

M and d_{VC}

copied from Lecture 5 :-)

- ① can we make sure that $E_{out}(g)$ is close enough to $E_{in}(g)$?
- ② can we make $E_{in}(g)$ small enough?

small M

- ① Yes!,
 $\mathbb{P}[\text{BAD}] \leq 2 \cdot M \cdot \exp(\dots)$
- ② No!, too few choices

large M

- ① No!,
 $\mathbb{P}[\text{BAD}] \leq 2 \cdot M \cdot \exp(\dots)$
- ② Yes!, many choices

small d_{VC}

- ① Yes!, $\mathbb{P}[\text{BAD}] \leq 4 \cdot (2N)^{d_{VC}} \cdot \exp(\dots)$
- ② No!, too limited power

large d_{VC}

- ① No!, $\mathbb{P}[\text{BAD}] \leq 4 \cdot (2N)^{d_{VC}} \cdot \exp(\dots)$
- ② Yes!, lots of power

using the right d_{VC} (or \mathcal{H}) is important

Fun Time

Origin-crossing Hyperplanes are essentially perceptrons with w_0 fixed at 0. Make a guess about the d_{VC} of origin-crossing hyperplanes in \mathbb{R}^d .

- 1 1
- 2 d
- 3 $d + 1$
- 4 ∞

Reference Answer: (2)

The proof is almost the same as proving the d_{VC} for usual perceptrons, but it is the **intuition** ($d_{VC} \approx \# \text{free parameters}$) that you shall use to answer this quiz.

VC Bound Rephrase: Penalty for Model Complexity

For any $\mathbf{g} = \mathcal{A}(\mathcal{D}) \in \mathcal{H}$ and ‘statistical’ large \mathcal{D} , for $N \geq 2, d_{VC} \geq 2$

$$\mathbb{P}_{\mathcal{D}} \left[\underbrace{|E_{in}(\mathbf{g}) - E_{out}(\mathbf{g})| > \epsilon}_{\text{BAD}} \right] \leq \underbrace{4(2N)^{d_{VC}} \exp\left(-\frac{1}{8}\epsilon^2 N\right)}_{\delta}$$

Rephrase

..., with probability $\geq 1 - \delta$, **GOOD**: $|E_{in}(\mathbf{g}) - E_{out}(\mathbf{g})| \leq \epsilon$

$$\text{set } \delta = 4(2N)^{d_{VC}} \exp\left(-\frac{1}{8}\epsilon^2 N\right)$$

$$\frac{\delta}{4(2N)^{d_{VC}}} = \exp\left(-\frac{1}{8}\epsilon^2 N\right)$$

$$\ln\left(\frac{4(2N)^{d_{VC}}}{\delta}\right) = \frac{1}{8}\epsilon^2 N$$

$$\sqrt{\frac{8}{N} \ln\left(\frac{4(2N)^{d_{VC}}}{\delta}\right)} = \epsilon$$

VC Bound Rephrase: Penalty for Model Complexity

For any $\mathbf{g} = \mathcal{A}(\mathcal{D}) \in \mathcal{H}$ and ‘statistical’ large \mathcal{D} , for $N \geq 2, d_{\text{VC}} \geq 2$

$$\mathbb{P}_{\mathcal{D}} \left[\underbrace{|E_{\text{in}}(\mathbf{g}) - E_{\text{out}}(\mathbf{g})| > \epsilon}_{\text{BAD}} \right] \leq \underbrace{4(2N)^{d_{\text{VC}}} \exp\left(-\frac{1}{8}\epsilon^2 N\right)}_{\delta}$$

Rephrase

..., with probability $\geq 1 - \delta$, **GOOD!**

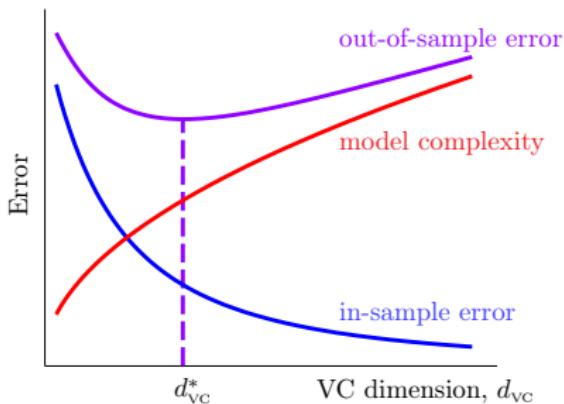
$$\begin{aligned} \text{gen. error } |E_{\text{in}}(\mathbf{g}) - E_{\text{out}}(\mathbf{g})| &\leq \sqrt{\frac{8}{N} \ln \left(\frac{4(2N)^{d_{\text{VC}}}}{\delta} \right)} \\ E_{\text{in}}(\mathbf{g}) - \sqrt{\frac{8}{N} \ln \left(\frac{4(2N)^{d_{\text{VC}}}}{\delta} \right)} &\leq E_{\text{out}}(\mathbf{g}) \leq E_{\text{in}}(\mathbf{g}) + \sqrt{\frac{8}{N} \ln \left(\frac{4(2N)^{d_{\text{VC}}}}{\delta} \right)} \end{aligned}$$

$\underbrace{\sqrt{\dots}}_{\Omega(N, \mathcal{H}, \delta)}$: penalty for **model complexity**

THE VC Message

with a high probability,

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \underbrace{\sqrt{\frac{8}{N} \ln \left(\frac{4(2N)^{d_{\text{VC}}}}{\delta} \right)}}_{\Omega(N, \mathcal{H}, \delta)}$$



- $d_{\text{VC}} \uparrow$: $E_{\text{in}} \downarrow$ but $\Omega \uparrow$
- $d_{\text{VC}} \downarrow$: $\Omega \downarrow$ but $E_{\text{in}} \uparrow$
- best d_{VC}^* in the middle

powerful \mathcal{H} not always good!

VC Bound Rephrase: Sample Complexity

For any $\mathbf{g} = \mathcal{A}(\mathcal{D}) \in \mathcal{H}$ and ‘statistical’ large \mathcal{D} , for $N \geq 2$, $d_{\text{VC}} \geq 2$

$$\mathbb{P}_{\mathcal{D}} \left[\underbrace{|E_{\text{in}}(\mathbf{g}) - E_{\text{out}}(\mathbf{g})| > \epsilon}_{\text{BAD}} \right] \leq \underbrace{4(2N)^{d_{\text{VC}}} \exp\left(-\frac{1}{8}\epsilon^2 N\right)}_{\delta}$$

given **specs** $\epsilon = 0.1$, $\delta = 0.1$, $d_{\text{VC}} = 3$, want $4(2N)^{d_{\text{VC}}} \exp\left(-\frac{1}{8}\epsilon^2 N\right) \leq \delta$

N	bound
100	2.82×10^7
1,000	9.17×10^9
10,000	1.19×10^8
100,000	1.65×10^{-38}
29,300	9.99×10^{-2}

sample complexity:
need $N \approx 10,000d_{\text{VC}}$ in theory

practical rule of thumb:

$N \approx 10d_{\text{VC}}$ often enough!

Looseness of VC Bound

$$\mathbb{P}_{\mathcal{D}} \left[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon \right] \leq 4(2N)^{d_{\text{VC}}} \exp \left(-\frac{1}{8} \epsilon^2 N \right)$$

theory: $N \approx 10,000d_{vc}$; practice: $N \approx 10d_{vc}$

Why?

- Hoeffding for unknown E_{out} any distribution, any target
 - $m_{\mathcal{H}}(N)$ instead of $|\mathcal{H}(\mathbf{x}_1, \dots, \mathbf{x}_N)|$ ‘any’ data
 - $N^{d_{vc}}$ instead of $m_{\mathcal{H}}(N)$ ‘any’ \mathcal{H} of same d_{vc}
 - union bound on worst cases any choice made by \mathcal{A}

—but hardly better, and ‘similarly loose for all models’

philosophical message of VC bound important for improving ML

Fun Time

Consider the VC Bound below. How can we decrease the probability of getting **BAD** data?

$$\mathbb{P}_{\mathcal{D}} \left[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon \right] \leq 4(2N)^{d_{\text{VC}}} \exp \left(-\frac{1}{8}\epsilon^2 N \right)$$

- ① decrease model complexity d_{VC}
- ② increase data size N a lot
- ③ increase generalization error tolerance ϵ
- ④ all of the above

Reference Answer: ④

Congratulations on being
Master of VC bound! :-)

Summary

1 When Can Machines Learn?

2 Why Can Machines Learn?

Lecture 6: Theory of Generalization

Lecture 7: The VC Dimension

- Definition of VC Dimension

maximum non-break point

- VC Dimension of Perceptrons

$$d_{VC}(\mathcal{H}) = d + 1$$

- Physical Intuition of VC Dimension

$d_{VC} \approx \#$ free parameters

- Interpreting VC Dimension

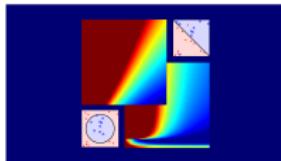
loosely: model complexity & sample complexity

- next: more than noiseless binary classification?

3 How Can Machines Learn?

4 How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 8: Noise and Error

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?

② Why Can Machines Learn?

Lecture 7: The VC Dimension

learning happens
if **finite d_{VC} , large N , and low E_{in}**

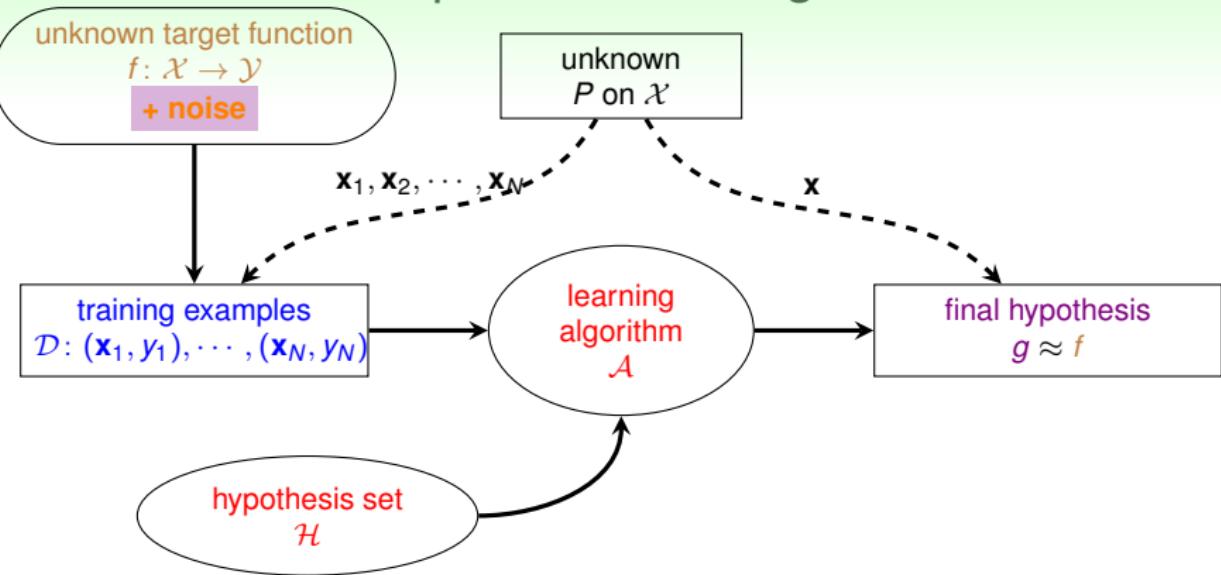
Lecture 8: Noise and Error

- Noise and Probabilistic Target
- Error Measure
- Algorithmic Error Measure
- Weighted Classification

③ How Can Machines Learn?

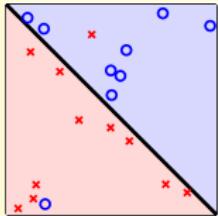
④ How Can Machines Learn Better?

Recap: The Learning Flow



what if there is **noise**?

Noise



briefly introduced **noise** before **pocket** algorithm

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

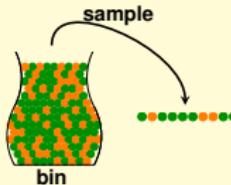
credit? {no(-1), yes($+1$)}
but more!

- **noise in y :** good customer, ‘mislabeled’ as bad?
- **noise in y :** same customers, different labels?
- **noise in x :** inaccurate customer information?

does VC bound work under **noise**?

Probabilistic Marbles

one key of VC bound: **marbles!**



'deterministic' marbles

- marble $\mathbf{x} \sim P(\mathbf{x})$
- deterministic color
 $\llbracket f(\mathbf{x}) \neq h(\mathbf{x}) \rrbracket$

'probabilistic' (noisy) marbles

- marble $\mathbf{x} \sim P(\mathbf{x})$
- probabilistic color
 $\llbracket y \neq h(\mathbf{x}) \rrbracket$ with $y \sim P(y|\mathbf{x})$

same nature: can estimate $\mathbb{P}[\text{orange}]$ if $\stackrel{i.i.d.}{\sim}$

VC holds for $\underbrace{\mathbf{x} \stackrel{i.i.d.}{\sim} P(\mathbf{x}), y \stackrel{i.i.d.}{\sim} P(y|\mathbf{x})}_{(\mathbf{x},y) \stackrel{i.i.d.}{\sim} P(\mathbf{x},y)}$

Target Distribution $P(y|\mathbf{x})$

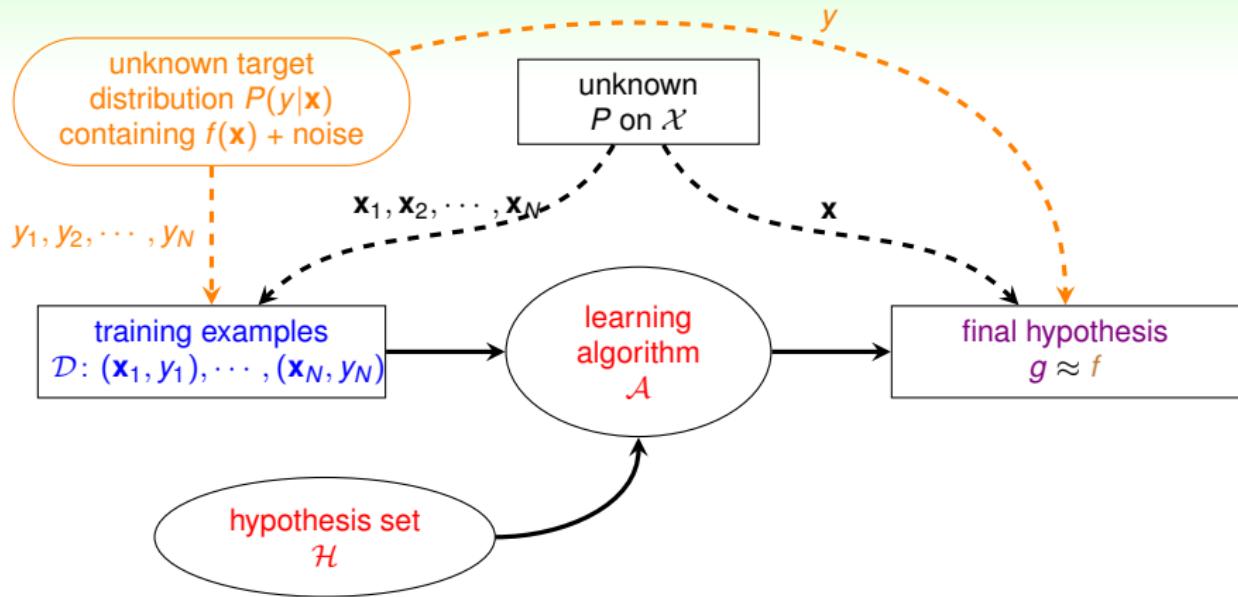
characterizes behavior of '**mini-target**' on one \mathbf{x}

- can be viewed as 'ideal mini-target' + noise, e.g.
 - $P(\circ|\mathbf{x}) = 0.7$, $P(\times|\mathbf{x}) = 0.3$
 - ideal mini-target $f(\mathbf{x}) = \circ$
 - 'flipping' noise level = 0.3
- deterministic target f : **special case of target distribution**
 - $P(y|\mathbf{x}) = 1$ for $y = f(\mathbf{x})$
 - $P(y|\mathbf{x}) = 0$ for $y \neq f(\mathbf{x})$

goal of learning:

predict **ideal mini-target (w.r.t. $P(y|\mathbf{x})$)**
on **often-seen inputs (w.r.t. $P(\mathbf{x})$)**

The New Learning Flow



VC still works, pocket algorithm explained :-)

Fun Time

Let's revisit PLA/pocket. Which of the following claim is true?

- ① In practice, we should try to compute if \mathcal{D} is linear separable before deciding to use PLA.
- ② If we know that \mathcal{D} is not linear separable, then the target function f must not be a linear function.
- ③ If we know that \mathcal{D} is linear separable, then the target function f must be a linear function.
- ④ None of the above

Reference Answer: ④

- ① After computing if \mathcal{D} is linear separable, we shall know \mathbf{w}^* and then there is no need to use PLA. ② What about noise? ③ What about 'sampling luck'? :-)

Error Measure

final hypothesis
 $g \approx f$

- how well? previously, considered out-of-sample measure

$$E_{\text{out}}(g) = \mathbb{E}_{\mathbf{x} \sim P} [\mathbb{I}[g(\mathbf{x}) \neq f(\mathbf{x})]]$$

- more generally, **error measure** $E(g, f)$
- naturally considered
 - **out-of-sample**: averaged over unknown \mathbf{x}
 - **pointwise**: evaluated on one \mathbf{x}
 - **classification**: $[\mathbb{I}[\text{prediction} \neq \text{target}]]$

classification error $[\dots]$:
often also called '**0/1 error**'

Pointwise Error Measure

can often express $E(g, f) = \text{averaged } \text{err}(g(\mathbf{x}), f(\mathbf{x}))$, like

$$E_{\text{out}}(g) = \mathcal{E}_{\mathbf{x} \sim P} \underbrace{\llbracket g(\mathbf{x}) \neq f(\mathbf{x}) \rrbracket}_{\text{err}(g(\mathbf{x}), f(\mathbf{x}))}$$

—err: called **pointwise error measure**

in-sample

$$E_{\text{in}}(g) = \frac{1}{N} \sum_{n=1}^N \text{err}(g(\mathbf{x}_n), f(\mathbf{x}_n))$$

out-of-sample

$$E_{\text{out}}(g) = \mathcal{E}_{\mathbf{x} \sim P} \text{err}(g(\mathbf{x}), f(\mathbf{x}))$$

will mainly consider pointwise err for simplicity

Two Important Pointwise Error Measures

$$\text{err} \left(\underbrace{g(\mathbf{x})}_{\tilde{y}}, \underbrace{f(\mathbf{x})}_{y} \right)$$

0/1 error

$$\text{err}(\tilde{y}, y) = \llbracket \tilde{y} \neq y \rrbracket$$

- correct or incorrect?
- often for classification

squared error

$$\text{err}(\tilde{y}, y) = (\tilde{y} - y)^2$$

- how far is \tilde{y} from y ?
- often for regression

how does err '**guide**' learning?

Ideal Mini-Target

interplay between **noise** and **error**:

$P(y|\mathbf{x})$ and err define **ideal mini-target** $f(\mathbf{x})$

$$P(y = 1|\mathbf{x}) = 0.2, P(y = 2|\mathbf{x}) = 0.7, P(y = 3|\mathbf{x}) = 0.1$$

$$\text{err}(\tilde{y}, y) = \llbracket \tilde{y} \neq y \rrbracket$$

$$\tilde{y} = \begin{cases} 1 & \text{avg. err 0.8} \\ 2 & \text{avg. err 0.3(*)} \\ 3 & \text{avg. err 0.9} \\ 1.9 & \text{avg. err 1.0(Really? :-))} \end{cases}$$

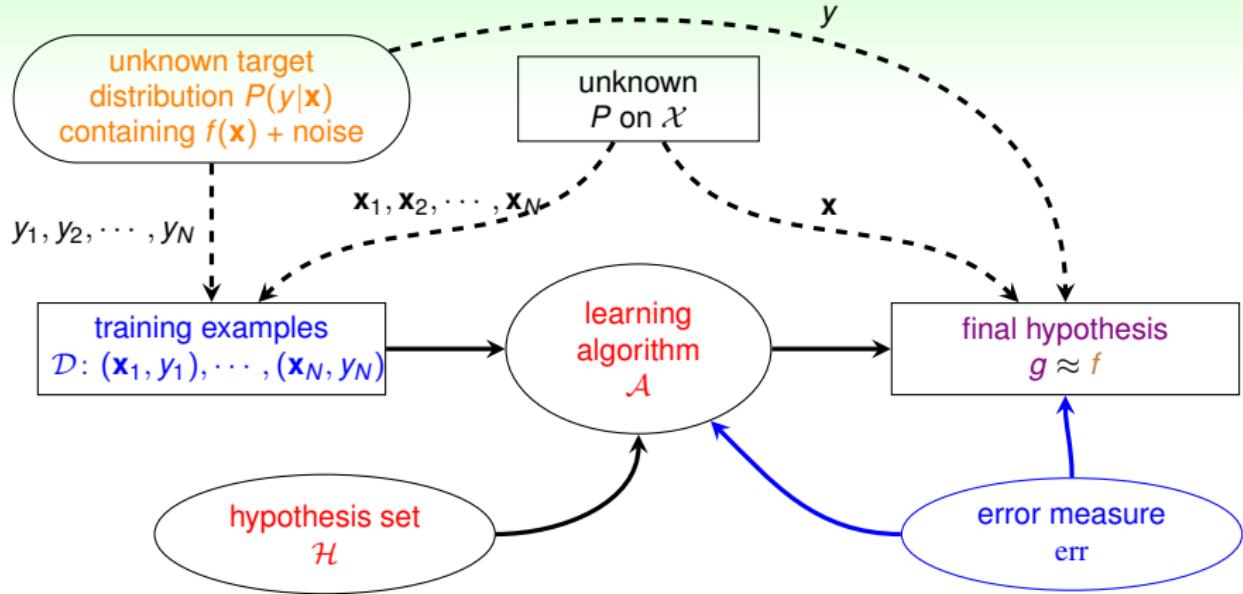
$$f(\mathbf{x}) = \operatorname{argmax}_{y \in \mathcal{Y}} P(y|\mathbf{x})$$

$$\text{err}(\tilde{y}, y) = (\tilde{y} - y)^2$$

$$\begin{cases} 1 & \text{avg. err 1.1} \\ 2 & \text{avg. err 0.3} \\ 3 & \text{avg. err 1.5} \\ 1.9 & \text{avg. err 0.29(*)} \end{cases}$$

$$f(\mathbf{x}) = \sum_{y \in \mathcal{Y}} y \cdot P(y|\mathbf{x})$$

Learning Flow with Error Measure



extended VC theory/'philosophy'
works for most \mathcal{H} and err

Fun Time

Consider the following $P(y|\mathbf{x})$ and $\text{err}(\tilde{y}, y) = |\tilde{y} - y|$. Which of the following is the ideal mini-target $f(\mathbf{x})$?

$$\begin{aligned}P(y = 1|\mathbf{x}) &= 0.10, P(y = 2|\mathbf{x}) = 0.35, \\P(y = 3|\mathbf{x}) &= 0.15, P(y = 4|\mathbf{x}) = 0.40.\end{aligned}$$

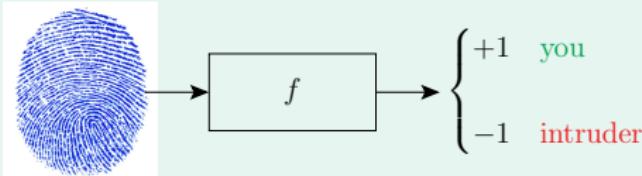
- ① $2.5 = \text{average within } \mathcal{Y} = \{1, 2, 3, 4\}$
- ② $2.85 = \text{weighted mean from } P(y|\mathbf{x})$
- ③ $3 = \text{weighted median from } P(y|\mathbf{x})$
- ④ $4 = \text{argmax } P(y|\mathbf{x})$

Reference Answer: ③

For the ‘absolute error’, the weighted median provably results in the minimum average err.

Choice of Error Measure

Fingerprint Verification



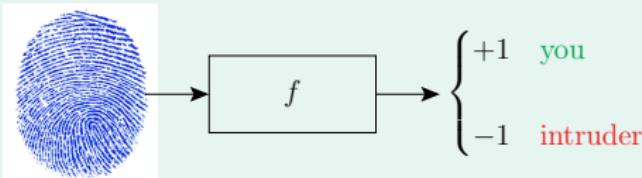
two types of error: **false accept** and **false reject**

		g	
		+1	-1
f	+1	no error	false reject
	-1	false accept	no error

0/1 error penalizes both types **equally**

Fingerprint Verification for Supermarket

Fingerprint Verification



two types of error: **false accept** and **false reject**

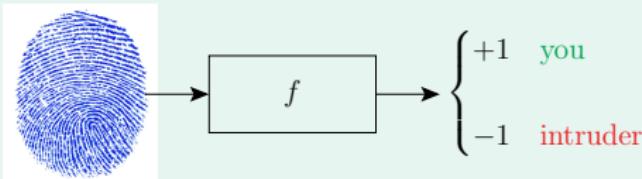
		g	
		+1	-1
f	+1	no error	false reject
	-1	false accept	no error

		g	
		+1	-1
f	+1	0	10
	-1	1	0

- supermarket: fingerprint for discount
- false reject:** **very unhappy customer, lose future business**
- false accept:** give away a minor discount, intruder left fingerprint :-)

Fingerprint Verification for CIA

Fingerprint Verification



two types of error: **false accept** and **false reject**

		g	
		+1	-1
f	+1	no error	false reject
	-1	false accept	no error

		g	
		+1	-1
f	+1	0	1
	-1	1000	0

- CIA: fingerprint for entrance
- **false accept:** **very serious consequences!**
- **false reject:** unhappy employee, but so what? :-)

Take-home Message for Now

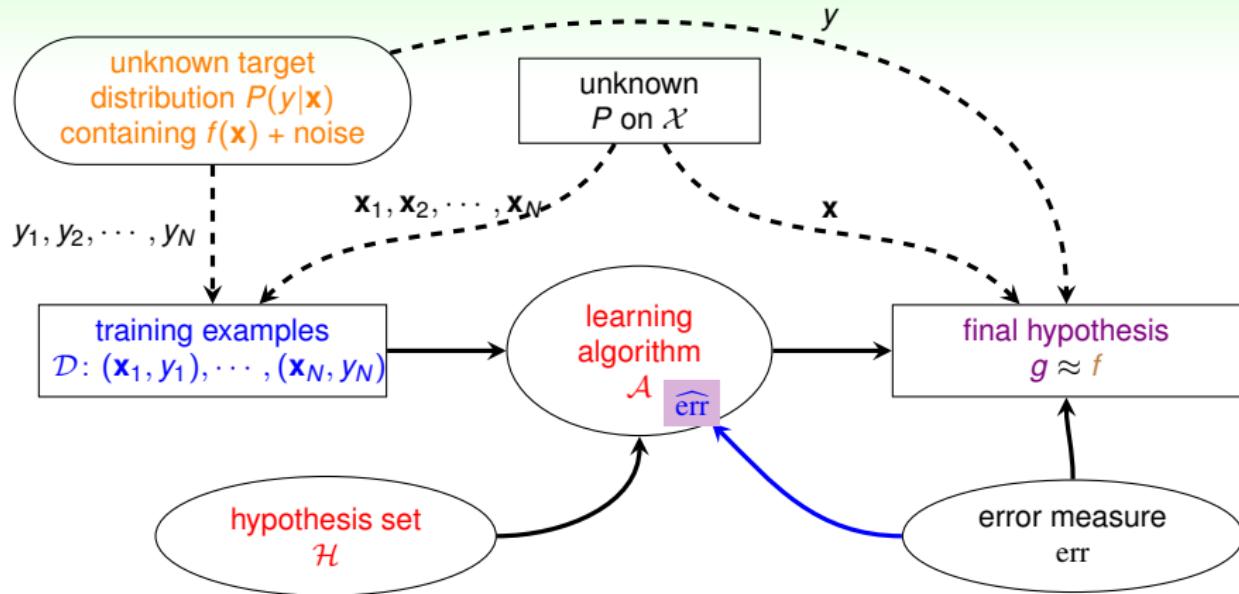
err is application/user-dependent

Algorithmic Error Measures $\widehat{\text{err}}$

- true: just **err**
- plausible:
 - 0/1: minimum 'flipping noise'—NP-hard to optimize, **remember? :-)**
 - squared: minimum **Gaussian noise**
- friendly: easy to optimize for \mathcal{A}
 - closed-form solution
 - convex objective function

$\widehat{\text{err}}$: more in next lectures

Learning Flow with Algorithmic Error Measure



err: application goal;
 $\widehat{\text{err}}$: a key part of many \mathcal{A}

Fun Time

Consider err below for CIA. What is $E_{\text{in}}(g)$ when using this err?

		g			
	+1	0	-1		
f	+1	1000	0		
	-1				

1 $\frac{1}{N} \sum_{n=1}^N \llbracket y_n \neq g(\mathbf{x}_n) \rrbracket$
2 $\frac{1}{N} \left(\sum_{y_n=+1} \llbracket y_n \neq g(\mathbf{x}_n) \rrbracket + 1000 \sum_{y_n=-1} \llbracket y_n \neq g(\mathbf{x}_n) \rrbracket \right)$
3 $\frac{1}{N} \left(\sum_{y_n=+1} \llbracket y_n \neq g(\mathbf{x}_n) \rrbracket - 1000 \sum_{y_n=-1} \llbracket y_n \neq g(\mathbf{x}_n) \rrbracket \right)$
4 $\frac{1}{N} \left(1000 \sum_{y_n=+1} \llbracket y_n \neq g(\mathbf{x}_n) \rrbracket + \sum_{y_n=-1} \llbracket y_n \neq g(\mathbf{x}_n) \rrbracket \right)$

Reference Answer: (2)

When $y_n = -1$, the **false positive** made on such (\mathbf{x}_n, y_n) is penalized **1000 times more!**

Weighted Classification

CIA Cost (Error, Loss, ...) Matrix

		$h(\mathbf{x})$	
		+1	-1
y	+1	0	1
	-1	1000	0

out-of-sample

$$E_{\text{out}}(h) = \mathbb{E}_{(\mathbf{x}, y) \sim P} \left\{ \begin{array}{ll} 1 & \text{if } y = +1 \\ 1000 & \text{if } y = -1 \end{array} \right\} \cdot \mathbb{I}[y \neq h(\mathbf{x})]$$

in-sample

$$E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^N \left\{ \begin{array}{ll} 1 & \text{if } y_n = +1 \\ 1000 & \text{if } y_n = -1 \end{array} \right\} \cdot \mathbb{I}[y_n \neq h(\mathbf{x}_n)]$$

weighted classification:
different 'weight' for different (\mathbf{x}, y)

Minimizing E_{in} for Weighted Classification

$$E_{\text{in}}^w(h) = \frac{1}{N} \sum_{n=1}^N \left\{ \begin{array}{ll} 1 & \text{if } y_n = +1 \\ 1000 & \text{if } y_n = -1 \end{array} \right\} \cdot \llbracket y_n \neq h(\mathbf{x}_n) \rrbracket$$

Naïve Thoughts

- PLA: **doesn't matter if linear separable. :-)**
- pocket: modify **pocket-replacement rule**
 - if \mathbf{w}_{t+1} reaches smaller E_{in}^w than $\hat{\mathbf{w}}$, replace $\hat{\mathbf{w}}$ by \mathbf{w}_{t+1}

pocket: some guarantee on $E_{\text{in}}^{0/1}$;
modified pocket: similar guarantee on E_{in}^w ?

Systematic Route: Connect E_{in}^w and $E_{\text{in}}^{0/1}$

original problem

		$h(\mathbf{x})$	
		+1	-1
y	+1	0	1
	-1	1000	0

- $\mathcal{D}:$
- $(\mathbf{x}_1, +1)$
 - $(\mathbf{x}_2, -1)$
 - $(\mathbf{x}_3, -1)$
 - ...
 - $(\mathbf{x}_{N-1}, +1)$
 - $(\mathbf{x}_N, +1)$

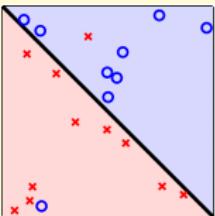
equivalent problem

		$h(\mathbf{x})$	
		+1	-1
y	+1	0	1
	-1	1	0

- $(\mathbf{x}_1, +1)$
- $(\mathbf{x}_2, -1), (\mathbf{x}_2, -1), \dots, (\mathbf{x}_2, -1)$
- $(\mathbf{x}_3, -1), (\mathbf{x}_3, -1), \dots, (\mathbf{x}_3, -1)$
- ...
- $(\mathbf{x}_{N-1}, +1)$
- $(\mathbf{x}_N, +1)$

after **copying -1 examples 1000 times**,
 E_{in}^w for LHS $\equiv E_{\text{in}}^{0/1}$ for RHS!

Weighted Pocket Algorithm



		$h(\mathbf{x})$	
		+1	-1
y	+1	0	1
	-1	1000	0

using ‘virtual copying’, weighted pocket algorithm include:

- weighted PLA:
randomly check **-1 example** mistakes with **1000** times more probability
- weighted pocket replacement:
if \mathbf{w}_{t+1} reaches smaller E_{in}^w than $\hat{\mathbf{w}}$, replace $\hat{\mathbf{w}}$ by \mathbf{w}_{t+1}

systematic route (called ‘reduction’):
can be applied to many other algorithms!

Fun Time

Consider the CIA cost matrix. If there are 10 examples with $y_n = -1$ (intruder) and 999,990 examples with $y_n = +1$ (you). What would $E_{in}^w(h)$ be for a constant $h(\mathbf{x})$ that always returns +1?

		$h(\mathbf{x})$	
		+1	-1
y	+1	0	1
	-1	1000	0

- 1 0.001
- 2 0.01
- 3 0.1
- 4 1

Reference Answer: (2)

While the quiz is a simple evaluation, it is not uncommon that the data is very **unbalanced** for such an application. Properly ‘setting’ the weights can be used to avoid the lazy constant prediction.

Summary

① When Can Machines Learn?

② Why Can Machines Learn?

Lecture 7: The VC Dimension

Lecture 8: Noise and Error

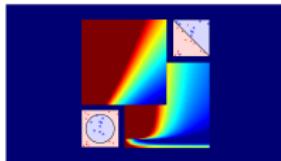
- Noise and Probabilistic Target
can replace $f(x)$ by $P(y|x)$
- Error Measure
affect ‘ideal’ target
- Algorithmic Error Measure
user-dependent \Rightarrow plausible or friendly
- Weighted Classification
easily done by virtual ‘example copying’

• next: more algorithms, please? :-)

③ How Can Machines Learn?

④ How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 9: Linear Regression

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?
- ② Why Can Machines Learn?

Lecture 8: Noise and Error

learning can happen
with **target distribution $P(y|x)$** and **low E_{in} w.r.t. err**

- ③ **How** Can Machines Learn?

Lecture 9: Linear Regression

- Linear Regression Problem
- Linear Regression Algorithm
- Generalization Issue
- Linear Regression for Binary Classification

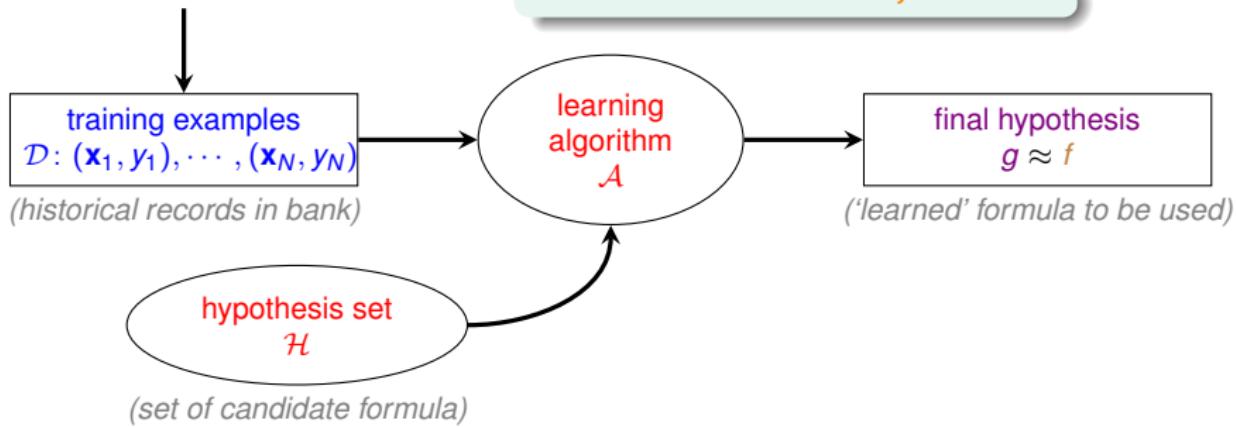
- ④ How Can Machines Learn Better?

Credit Limit Problem

unknown target function
 $f: \mathcal{X} \rightarrow \mathcal{Y}$
 (ideal credit **limit** formula)

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

credit limit? **100,000**



$\mathcal{Y} = \mathbb{R}$: regression

Linear Regression Hypothesis

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

- For $\mathbf{x} = (x_0, x_1, x_2, \dots, x_d)$ ‘features of customer’, approximate the **desired credit limit** with a **weighted sum**:

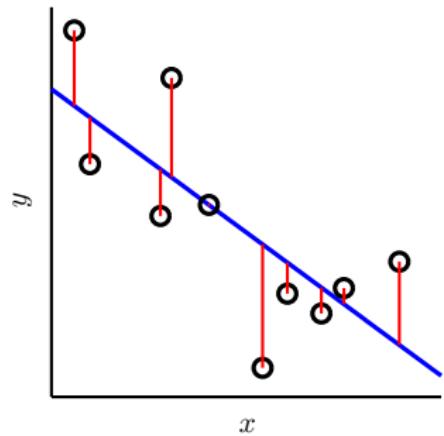
$$y \approx \sum_{i=0}^d w_i x_i$$

- linear regression hypothesis: $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

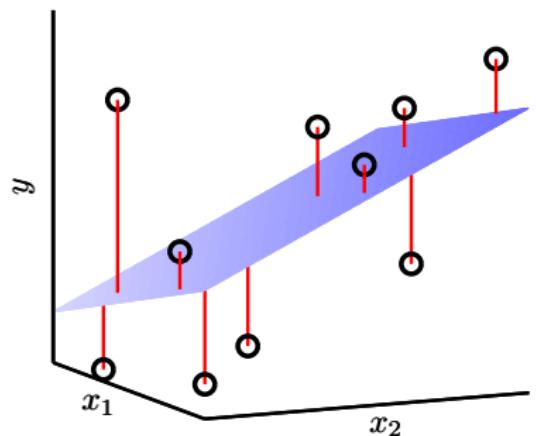
$h(\mathbf{x})$: like **perceptron**, but without the **sign**

Illustration of Linear Regression

$$\mathbf{x} = (x) \in \mathbb{R}$$



$$\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$$



linear regression:
find **lines/hyperplanes** with small **residuals**

The Error Measure

popular/historical error measure:

$$\text{squared error } \text{err}(\hat{y}, y) = (\hat{y} - y)^2$$

in-sample

$$E_{\text{in}}(\mathbf{h}\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \underbrace{(\mathbf{h}(\mathbf{x}_n) - y_n)^2}_{\mathbf{w}^T \mathbf{x}_n}$$

out-of-sample

$$E_{\text{out}}(\mathbf{w}) = \mathcal{E}_{(\mathbf{x}, \mathbf{y}) \sim P} (\mathbf{w}^T \mathbf{x} - y)^2$$

next: how to minimize $E_{\text{in}}(\mathbf{w})$?

Fun Time

Consider using linear regression hypothesis $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ to predict the credit limit of customers \mathbf{x} . Which feature below shall have a positive weight in a **good hypothesis** for the task?

- ① birth month
- ② monthly income
- ③ current debt
- ④ number of credit cards owned

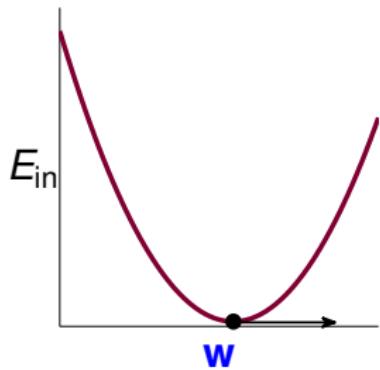
Reference Answer: ②

Customers with higher monthly income should naturally be given a higher credit limit, which is captured by the positive weight on the 'monthly income' feature.

Matrix Form of $E_{\text{in}}(\mathbf{w})$

$$\begin{aligned}
 E_{\text{in}}(\mathbf{w}) &= \frac{1}{N} \sum_{n=1}^N (\mathbf{w}^T \mathbf{x}_n - y_n)^2 = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n^T \mathbf{w} - y_n)^2 \\
 &= \frac{1}{N} \left\| \begin{array}{c} \mathbf{x}_1^T \mathbf{w} - y_1 \\ \mathbf{x}_2^T \mathbf{w} - y_2 \\ \dots \\ \mathbf{x}_N^T \mathbf{w} - y_N \end{array} \right\|^2 \\
 &= \frac{1}{N} \left\| \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \dots \\ \mathbf{x}_N^T \end{bmatrix} \mathbf{w} - \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} \right\|^2 \\
 &= \frac{1}{N} \| \underbrace{\mathbf{X}}_{N \times d+1} \underbrace{\mathbf{w}}_{d+1 \times 1} - \underbrace{\mathbf{y}}_{N \times 1} \|^2
 \end{aligned}$$

$$\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$



- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, **convex**
- necessary condition of ‘best’ \mathbf{w}

$$\nabla E_{\text{in}}(\mathbf{w}) \equiv \begin{bmatrix} \frac{\partial E_{\text{in}}}{\partial w_0}(\mathbf{w}) \\ \frac{\partial E_{\text{in}}}{\partial w_1}(\mathbf{w}) \\ \vdots \\ \frac{\partial E_{\text{in}}}{\partial w_d}(\mathbf{w}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

—not possible to ‘roll down’

task: find \mathbf{w}_{LIN} such that $\nabla E_{\text{in}}(\mathbf{w}_{\text{LIN}}) = \mathbf{0}$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 = \frac{1}{N} \left(\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{w}^T \mathbf{X}^T \mathbf{y} + \mathbf{y}^T \mathbf{y} \right)$$

A b c

one \mathbf{w} only

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} (a\mathbf{w}^2 - 2b\mathbf{w} + c)$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N} (2a\mathbf{w} - 2b)$$

simple! :-)

vector \mathbf{w}

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} (\mathbf{w}^T \mathbf{A} \mathbf{w} - 2\mathbf{w}^T \mathbf{b} + c)$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N} (2\mathbf{A}\mathbf{w} - 2\mathbf{b})$$

similar (derived by definition)

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N} (\mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y})$$

Optimal Linear Regression Weights

task: find \mathbf{w}_{LIN} such that $\frac{2}{N} (\mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y}) = \nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$

invertible $\mathbf{X}^T \mathbf{X}$

- **easy!** unique solution

$$\mathbf{w}_{\text{LIN}} = \underbrace{(\mathbf{X}^T \mathbf{X})^{-1}}_{\text{pseudo-inverse}} \mathbf{X}^T \mathbf{y}$$

- often the case because $N \gg d + 1$

singular $\mathbf{X}^T \mathbf{X}$

- **many** optimal solutions
- one of the solutions

$$\mathbf{w}_{\text{LIN}} = \mathbf{X}^\dagger \mathbf{y}$$

by defining \mathbf{X}^\dagger in other ways

practical suggestion:

use **well-implemented** \dagger routine

instead of $(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$

for numerical stability when **almost-singular**

Linear Regression Algorithm

- ① from \mathcal{D} , construct input matrix \mathbf{X} and output vector \mathbf{y} by

$$\mathbf{X} = \underbrace{\begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}}_{N \times (d+1)} \quad \mathbf{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{N \times 1}$$

- ② calculate pseudo-inverse

$$\underbrace{\mathbf{X}^\dagger}_{(d+1) \times N}$$

- ③ return

$$\underbrace{\mathbf{w}_{\text{LIN}}}_{(d+1) \times 1} = \mathbf{X}^\dagger \mathbf{y}$$

simple and efficient
with **good** † routine

Fun Time

After getting \mathbf{w}_{LIN} , we can calculate the predictions $\hat{\mathbf{y}}_n = \mathbf{w}_{\text{LIN}}^T \mathbf{x}_n$. If all $\hat{\mathbf{y}}_n$ are collected in a vector $\hat{\mathbf{y}}$ similar to how we form \mathbf{y} , what is the matrix formula of $\hat{\mathbf{y}}$?

- ① \mathbf{y}
- ② $\mathbf{X}\mathbf{X}^T \mathbf{y}$
- ③ $\mathbf{X}\mathbf{X}^\dagger \mathbf{y}$
- ④ $\mathbf{X}\mathbf{X}^\dagger \mathbf{X}\mathbf{X}^T \mathbf{y}$

Reference Answer: ③

Note that $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}_{\text{LIN}}$. Then, a simple substitution of \mathbf{w}_{LIN} reveals the answer.

Is Linear Regression a ‘Learning Algorithm’?

$$\mathbf{w}_{\text{LIN}} = \mathbf{X}^\dagger \mathbf{y}$$

No!

- analytic (**closed-form**) solution, ‘instantaneous’
- not improving E_{in} nor E_{out} iteratively

Yes!

- good E_{in} ?
yes, optimal!
- good E_{out} ?
yes, finite d_{VC} like perceptrons
- improving iteratively?
somewhat, within an iterative pseudo-inverse routine

if $E_{\text{out}}(\mathbf{w}_{\text{LIN}})$ is good, **learning ‘happened’!**

Benefit of Analytic Solution: 'Simpler-than-VC' Guarantee

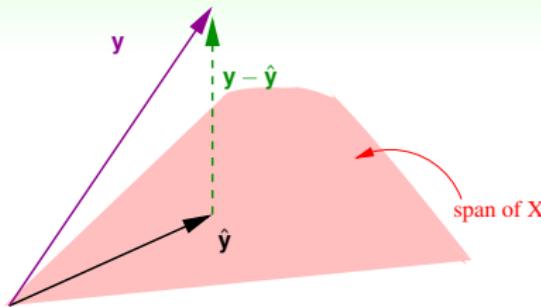
$$\overline{E_{\text{in}}} = \mathcal{E}_{\mathcal{D} \sim P^N} \left\{ E_{\text{in}}(\mathbf{w}_{\text{LIN}} \text{ w.r.t. } \mathcal{D}) \right\} \xrightarrow{\text{to be shown}} \text{noise level} \cdot \left(1 - \frac{d+1}{N}\right)$$



$$\begin{aligned}
 E_{\text{in}}(\mathbf{w}_{\text{LIN}}) &= \frac{1}{N} \|\mathbf{y} - \underbrace{\hat{\mathbf{y}}}_{\text{predictions}}\|^2 = \frac{1}{N} \|\mathbf{y} - \underbrace{\mathbf{X} \mathbf{X}^\dagger \mathbf{y}}_{\mathbf{w}_{\text{LIN}}}\|^2 \\
 &= \frac{1}{N} \|(\underbrace{\mathbf{I}}_{\text{identity}} - \mathbf{X} \mathbf{X}^\dagger) \mathbf{y}\|^2
 \end{aligned}$$

call $\mathbf{X} \mathbf{X}^\dagger$ the hat matrix \mathbf{H}
because it puts \wedge on \mathbf{y}

Geometric View of Hat Matrix

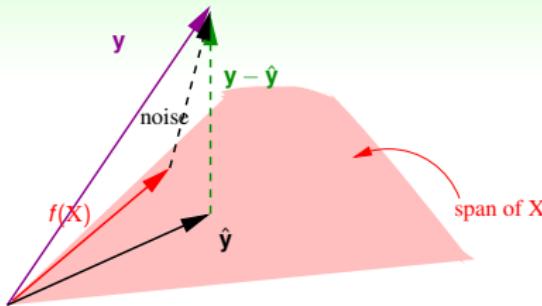


in \mathbb{R}^N

- $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}_{\text{LIN}}$ within the span of X columns
- $\mathbf{y} - \hat{\mathbf{y}}$ smallest: $\mathbf{y} - \hat{\mathbf{y}} \perp \text{span}$
- H: project \mathbf{y} to $\hat{\mathbf{y}} \in \text{span}$
- $\mathbf{I} - \mathbf{H}$: transform \mathbf{y} to $\mathbf{y} - \hat{\mathbf{y}} \perp \text{span}$

claim: $\text{trace}(\mathbf{I} - \mathbf{H}) = N - (d + 1)$. Why? :-)

An Illustrative ‘Proof’



- if \mathbf{y} comes from some ideal $f(\mathbf{X}) \in \text{span}$ plus **noise**
- **noise** transformed by $\mathbf{I} - \mathbf{H}$ to be $\mathbf{y} - \hat{\mathbf{y}}$

$$\begin{aligned} E_{\text{in}}(\mathbf{w}_{\text{LIN}}) &= \frac{1}{N} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \frac{1}{N} \|(\mathbf{I} - \mathbf{H})\mathbf{noise}\|^2 \\ &= \frac{1}{N} (N - (d + 1)) \|\mathbf{noise}\|^2 \end{aligned}$$

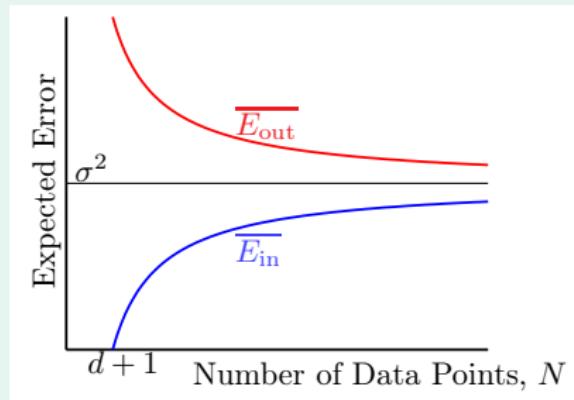
$$\overline{E_{\text{in}}} = \mathbf{noise} \text{ level} \cdot \left(1 - \frac{d+1}{N}\right)$$

$$\overline{E_{\text{out}}} = \mathbf{noise} \text{ level} \cdot \left(1 + \frac{d+1}{N}\right) \text{ (complicated!)}$$

The Learning Curve

$$\overline{E}_{\text{out}} = \text{noise level} \cdot \left(1 + \frac{d+1}{N}\right)$$

$$\overline{E}_{\text{in}} = \text{noise level} \cdot \left(1 - \frac{d+1}{N}\right)$$



- both converge to σ^2 (**noise level**) for $N \rightarrow \infty$
- expected generalization error: $\frac{2(d+1)}{N}$
—similar to worst-case guarantee from VC

linear regression (LinReg):
learning 'happened'!

Fun Time

Which of the following property about H is not true?

- ① H is symmetric
- ② $H^2 = H$ (double projection = single one)
- ③ $(I - H)^2 = I - H$ (double residual transform = single one)
- ④ none of the above

Reference Answer: ④

You can conclude that ② and ③ are true by their physical meanings! :-)

Linear Classification vs. Linear Regression

Linear Classification

$$\mathcal{Y} = \{-1, +1\}$$

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

$$\text{err}(\hat{y}, y) = \|\hat{y} \neq y\|$$

NP-hard to solve in general

Linear Regression

$$\mathcal{Y} = \mathbb{R}$$

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

$$\text{err}(\hat{y}, y) = (\hat{y} - y)^2$$

efficient analytic solution

$\{-1, +1\} \subset \mathbb{R}$: linear regression for classification?

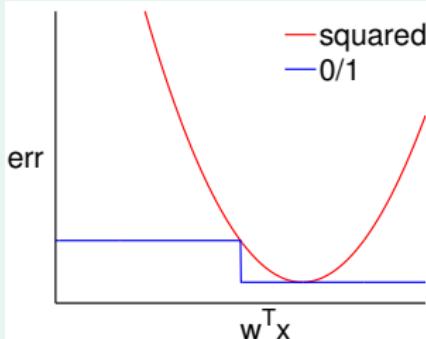
- ① run LinReg on binary classification data \mathcal{D} (efficient)
- ② return $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{LIN}}^T \mathbf{x})$

but explanation of this heuristic?

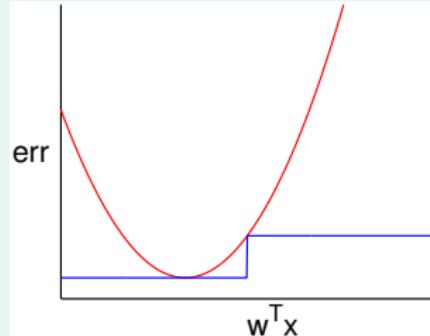
Relation of Two Errors

$$\text{err}_{0/1} = [\text{sign}(\mathbf{w}^T \mathbf{x}) \neq y] \quad \text{err}_{\text{sqr}} = (\mathbf{w}^T \mathbf{x} - y)^2$$

desired $y = 1$



desired $y = -1$



$$\text{err}_{0/1} \leq \text{err}_{\text{sqr}}$$

Linear Regression for Binary Classification

$$\text{err}_{0/1} \leq \text{err}_{\text{sqr}}$$

$$\begin{aligned}\text{classification } E_{\text{out}}(\mathbf{w}) &\stackrel{\text{VC}}{\leq} \text{classification } E_{\text{in}}(\mathbf{w}) + \sqrt{\dots} \\ &\leq \text{regression } E_{\text{in}}(\mathbf{w}) + \sqrt{\dots}\end{aligned}$$

- (loose) upper bound err_{sqr} as $\widehat{\text{err}}$ to approximate $\text{err}_{0/1}$
- trade **bound tightness** for **efficiency**

\mathbf{w}_{LIN} : useful baseline classifier,
or as **initial PLA/pocket vector**

Fun Time

Which of the following functions are upper bounds of the pointwise 0/1 error $\llbracket \text{sign}(\mathbf{w}^T \mathbf{x}) \neq y \rrbracket$ for $y \in \{-1, +1\}$?

- ① $\exp(-y\mathbf{w}^T \mathbf{x})$
- ② $\max(0, 1 - y\mathbf{w}^T \mathbf{x})$
- ③ $\log_2(1 + \exp(-y\mathbf{w}^T \mathbf{x}))$
- ④ all of the above

Reference Answer: ④

Plot the curves and you'll see. Thus, all three can be used for binary classification. In fact, all three functions connect to very important algorithms in machine learning and we will discuss one of them soon in the next lecture.

Stay tuned. :-)

Summary

- ① When Can Machines Learn?
- ② Why Can Machines Learn?

Lecture 8: Noise and Error

- ③ How Can Machines Learn?

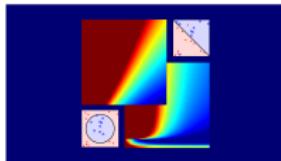
Lecture 9: Linear Regression

- Linear Regression Problem
use hyperplanes to approximate real values
- Linear Regression Algorithm
analytic solution with pseudo-inverse
- Generalization Issue
 $E_{\text{out}} - E_{\text{in}} \approx \frac{2(d+1)}{N}$ **on average**
- Linear Regression for Binary Classification
0/1 error \leq squared error

- next: **binary classification, regression, and then?**

- ④ How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 10: Logistic Regression

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?
- ② Why Can Machines Learn?
- ③ **How** Can Machines Learn?

Lecture 9: Linear Regression

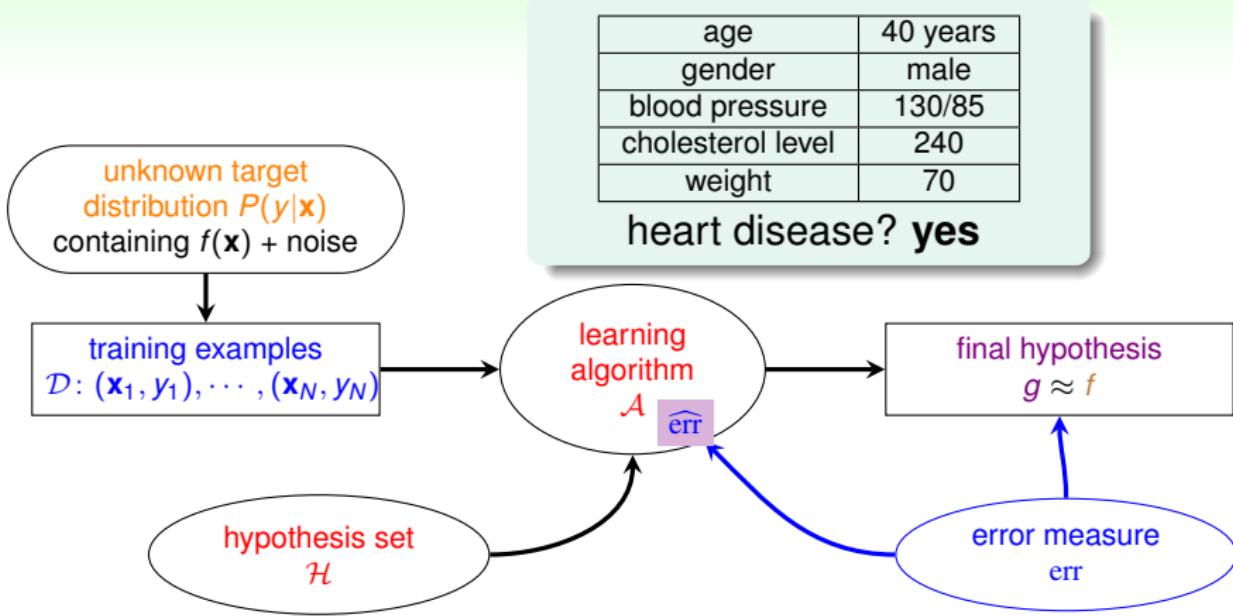
analytic solution $\mathbf{w}_{\text{LIN}} = \mathbf{X}^\dagger \mathbf{y}$ with
linear regression hypotheses and **squared error**

Lecture 10: Logistic Regression

- Logistic Regression Problem
- Logistic Regression Error
- Gradient of Logistic Regression Error
- Gradient Descent

- ④ How Can Machines Learn Better?

Heart Attack Prediction Problem (1/2)



age	40 years
gender	male
blood pressure	130/85
cholesterol level	240
weight	70

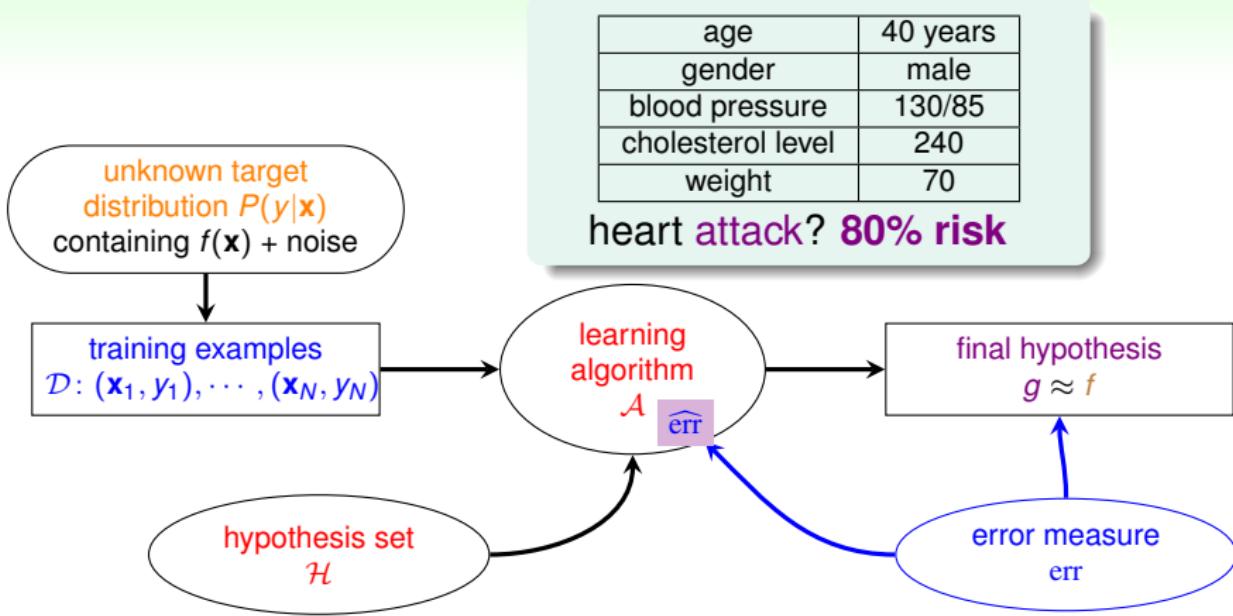
heart disease? **yes**

binary classification:

$$\text{ideal } f(\mathbf{x}) = \text{sign} (P(+1|\mathbf{x}) - \frac{1}{2}) \in \{-1, +1\}$$

because of **classification err**

Heart Attack Prediction Problem (2/2)



‘soft’ binary classification:

$$f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$$

Soft Binary Classification

target function $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$

ideal (noiseless) data

$$\begin{cases} \mathbf{x}_1, y'_1 = 0.9 = P(+1|\mathbf{x}_1) \\ \mathbf{x}_2, y'_2 = 0.2 = P(+1|\mathbf{x}_2) \end{cases}$$

⋮

$$\left(\mathbf{x}_N, y'_N = 0.6 = P(+1|\mathbf{x}_N) \right)$$

actual (noisy) data

$$\begin{cases} \mathbf{x}_1, y_1 = \circ \sim P(y|\mathbf{x}_1) \\ \mathbf{x}_2, y_2 = \times \sim P(y|\mathbf{x}_2) \end{cases}$$

⋮

$$\left(\mathbf{x}_N, y_N = \times \sim P(y|\mathbf{x}_N) \right)$$

same data as hard binary classification,
different **target function**

Soft Binary Classification

target function $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$

ideal (noiseless) data

$$\left\{ \begin{array}{l} \mathbf{x}_1, y'_1 = 0.9 = P(+1|\mathbf{x}_1) \\ \mathbf{x}_2, y'_2 = 0.2 = P(+1|\mathbf{x}_2) \end{array} \right.$$

⋮

$$\left(\mathbf{x}_N, y'_N = 0.6 = P(+1|\mathbf{x}_N) \right)$$

actual (noisy) data

$$\left\{ \begin{array}{l} \mathbf{x}_1, y'_1 = 1 = \left[\begin{array}{l} \circ \stackrel{?}{\sim} P(y|\mathbf{x}_1) \\ \circ \stackrel{?}{\sim} P(y|\mathbf{x}_2) \end{array} \right] \\ \mathbf{x}_2, y'_2 = 0 = \left[\begin{array}{l} \circ \stackrel{?}{\sim} P(y|\mathbf{x}_1) \\ \circ \stackrel{?}{\sim} P(y|\mathbf{x}_2) \end{array} \right] \end{array} \right.$$

⋮

$$\left(\mathbf{x}_N, y'_N = 0 = \left[\begin{array}{l} \circ \stackrel{?}{\sim} P(y|\mathbf{x}_1) \\ \circ \stackrel{?}{\sim} P(y|\mathbf{x}_2) \end{array} \right] \right)$$

same data as hard binary classification,
different **target function**

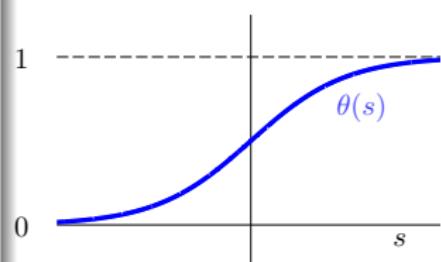
Logistic Hypothesis

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240

- For $\mathbf{x} = (x_0, x_1, x_2, \dots, x_d)$ ‘features of patient’, calculate a **weighted** ‘risk score’:

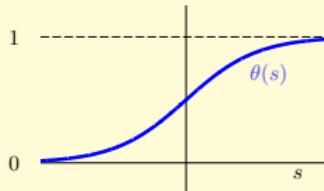
$$s = \sum_{i=0}^d w_i x_i$$

- convert the **score** to **estimated probability** by logistic function $\theta(s)$



logistic hypothesis: $h(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x})$

Logistic Function



$$\theta(-\infty) = 0;$$

$$\theta(0) = \frac{1}{2};$$

$$\theta(\infty) = 1$$

$$\theta(s) = \frac{e^s}{1 + e^s} = \frac{1}{1 + e^{-s}}$$

—smooth, monotonic, **sigmoid** function of s

logistic regression: use

$$h(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

to approximate target function $f(\mathbf{x}) = P(y|\mathbf{x})$

Fun Time

Logistic Regression and Binary Classification

Consider any logistic hypothesis $h(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T \mathbf{x})}$ that approximates $P(y|\mathbf{x})$. ‘Convert’ $h(\mathbf{x})$ to a binary classification prediction by taking sign ($h(\mathbf{x}) - \frac{1}{2}$). What is the equivalent formula for the binary classification prediction?

- ① sign ($\mathbf{w}^T \mathbf{x} - \frac{1}{2}$)
- ② sign ($\mathbf{w}^T \mathbf{x}$)
- ③ sign ($\mathbf{w}^T \mathbf{x} + \frac{1}{2}$)
- ④ none of the above

Reference Answer: ②

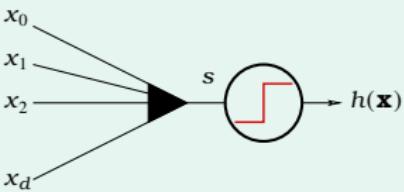
When $\mathbf{w}^T \mathbf{x} = 0$, $h(\mathbf{x})$ is exactly $\frac{1}{2}$. So thresholding $h(\mathbf{x})$ at $\frac{1}{2}$ is the same as thresholding ($\mathbf{w}^T \mathbf{x}$) at 0.

Three Linear Models

linear scoring function: $s = \mathbf{w}^T \mathbf{x}$

linear classification

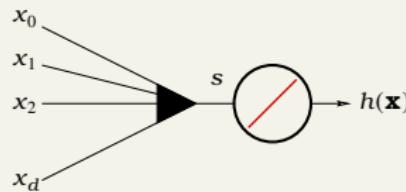
$$h(\mathbf{x}) = \text{sign}(s)$$



plausible err = 0/1
(small flipping noise)

linear regression

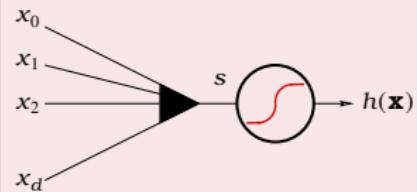
$$h(\mathbf{x}) = s$$



friendly err = squared
(easy to minimize)

logistic regression

$$h(\mathbf{x}) = \theta(s)$$



err = ?

how to define
 $E_{\text{in}}(\mathbf{w})$ for logistic regression?

Likelihood

target function
 $f(\mathbf{x}) = P(+1|\mathbf{x})$



$$P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases}$$

consider $\mathcal{D} = \{(\mathbf{x}_1, \circ), (\mathbf{x}_2, \times), \dots, (\mathbf{x}_N, \times)\}$

probability that f generates \mathcal{D}

$$P(\mathbf{x}_1)P(\circ|\mathbf{x}_1) \times$$

$$P(\mathbf{x}_2)P(\times|\mathbf{x}_2) \times$$

...

$$P(\mathbf{x}_N)P(\times|\mathbf{x}_N)$$

likelihood that h generates \mathcal{D}

$$P(\mathbf{x}_1)h(\mathbf{x}_1) \times$$

$$P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times$$

...

$$P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

- if $h \approx f$,
then likelihood(h) \approx probability using f
- probability using f usually **large**

Likelihood

target function
 $f(\mathbf{x}) = P(+1|\mathbf{x})$



$$P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases}$$

consider $\mathcal{D} = \{(\mathbf{x}_1, \circ), (\mathbf{x}_2, \times), \dots, (\mathbf{x}_N, \times)\}$

probability that f generates \mathcal{D}

$$\begin{aligned} P(\mathbf{x}_1)f(\mathbf{x}_1) \times \\ P(\mathbf{x}_2)(1 - f(\mathbf{x}_2)) \times \\ \dots \\ P(\mathbf{x}_N)(1 - f(\mathbf{x}_N)) \end{aligned}$$

likelihood that h generates \mathcal{D}

$$\begin{aligned} P(\mathbf{x}_1)h(\mathbf{x}_1) \times \\ P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times \\ \dots \\ P(\mathbf{x}_N)(1 - h(\mathbf{x}_N)) \end{aligned}$$

- if $h \approx f$,
then likelihood(h) \approx probability using f
- probability using f usually **large**

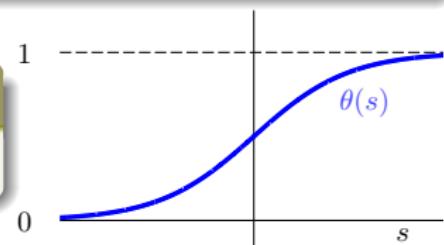
Likelihood of Logistic Hypothesis

likelihood(h) \approx (probability using f) \approx large

$$g = \operatorname{argmax}_h \text{ likelihood}(h)$$

when logistic: $h(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x})$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$



$$\text{likelihood}(h) = P(\mathbf{x}_1)h(\mathbf{x}_1) \times P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times \dots P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

$$\text{likelihood(logistic } h) \propto \prod_{n=1}^N h(y_n \mathbf{x}_n)$$

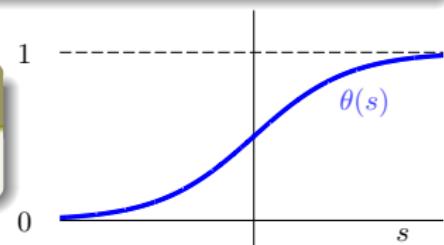
Likelihood of Logistic Hypothesis

likelihood(h) \approx (probability using f) \approx large

$$g = \operatorname{argmax}_h \text{ likelihood}(h)$$

when logistic: $h(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x})$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$



$$\text{likelihood}(h) = P(\mathbf{x}_1)h(+\mathbf{x}_1) \times P(\mathbf{x}_2)h(-\mathbf{x}_2) \times \dots P(\mathbf{x}_N)h(-\mathbf{x}_N)$$

$$\text{likelihood(logistic } h) \propto \prod_{n=1}^N h(y_n \mathbf{x}_n)$$

Cross-Entropy Error

$$\max_{\mathbf{h}} \text{likelihood}(\text{logistic } \mathbf{h}) \propto \prod_{n=1}^N \mathbf{h}(y_n \mathbf{x}_n)$$

Cross-Entropy Error

$$\max_{\mathbf{w}} \text{ likelihood}(\mathbf{w}) \propto \prod_{n=1}^N \theta(y_n \mathbf{w}^T \mathbf{x}_n)$$

Cross-Entropy Error

$$\max_{\mathbf{w}} \ln \prod_{n=1}^N \theta(y_n \mathbf{w}^T \mathbf{x}_n)$$

Cross-Entropy Error

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^N -\ln \theta(y_n \mathbf{w}^T \mathbf{x}_n)$$

$$\begin{aligned} \theta(s) = \frac{1}{1 + \exp(-s)} &: \min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^N \ln(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)) \\ \implies \min_{\mathbf{w}} & \underbrace{\frac{1}{N} \sum_{n=1}^N \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)}_{E_{\text{in}}(\mathbf{w})} \end{aligned}$$

$\text{err}(\mathbf{w}, \mathbf{x}, y) = \ln(1 + \exp(-y \mathbf{w}^T \mathbf{x}))$:
cross-entropy error

Fun Time

The four statements below help us understand more about the cross-entropy error $\text{err}(\mathbf{w}, \mathbf{x}, y) = \ln(1 + \exp(-y\mathbf{w}^T \mathbf{x}))$. Which statement is not true?

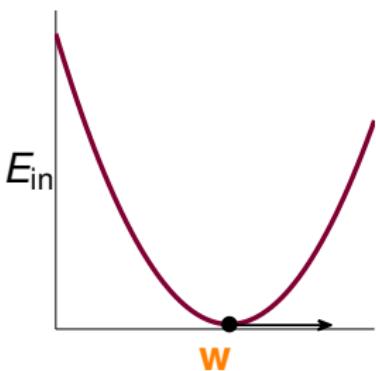
- ① For any \mathbf{w} , \mathbf{x} , and y , $\text{err}(\mathbf{w}, \mathbf{x}, y) > 0$.
- ② For any \mathbf{w} , \mathbf{x} , and y , $\text{err}(\mathbf{w}, \mathbf{x}, y) < 1126$.
- ③ When $y = \text{sign}(\mathbf{w}^T \mathbf{x})$, $\text{err}(\mathbf{w}, \mathbf{x}, y) < \ln 2$.
- ④ When $y \neq \text{sign}(\mathbf{w}^T \mathbf{x})$, $\text{err}(\mathbf{w}, \mathbf{x}, y) \geq \ln 2$.

Reference Answer: ②

1126, really? :-) You are highly encouraged to plot the curve of err with respect to some fixed y and some varying score $s = \mathbf{w}^T \mathbf{x}$ to know more about the error measure. After plotting, it is easy to see that err is not bounded above, and the other three choices are correct.

Minimizing $E_{\text{in}}(\mathbf{w})$

$$\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ln \left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n) \right)$$



- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, twice-differentiable, **convex**
- how to minimize? locate **valley**

want $\nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$

first: derive $\nabla E_{\text{in}}(\mathbf{w})$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ln \left(\underbrace{\frac{1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)}{\square}}_{\text{O}} \right)$$

$$\begin{aligned} \frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} &= \frac{1}{N} \sum_{n=1}^N \left(\frac{\partial \ln(\square)}{\partial \square} \right) \left(\frac{\partial (1 + \exp(\text{O}))}{\partial \text{O}} \right) \left(\frac{\partial -y_n \mathbf{w}^T \mathbf{x}_n}{\partial w_i} \right) \\ &= \frac{1}{N} \sum_{n=1}^N \left(\quad \right) \left(\quad \right) \left(\quad \right) \\ &= \frac{1}{N} \sum_{n=1}^N \left(\frac{\exp(\text{O})}{1 + \exp(\text{O})} \right) \left(-y_n x_{n,i} \right) = \frac{1}{N} \sum_{n=1}^N \theta(\text{O})(-y_n x_{n,i}) \end{aligned}$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \theta(-y_n \mathbf{w}^T \mathbf{x}_n)(-y_n \mathbf{x}_n)$$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ln \left(\underbrace{\frac{1}{1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)}}_{\square} \right)$$

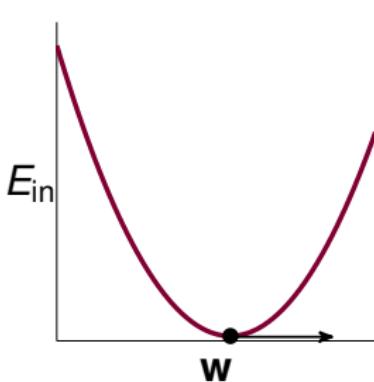
$$\begin{aligned} \frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} &= \frac{1}{N} \sum_{n=1}^N \left(\frac{\partial \ln(\square)}{\partial \square} \right) \left(\frac{\partial (1 + \exp(\bigcirc))}{\partial \bigcirc} \right) \left(\frac{\partial -y_n \mathbf{w}^T \mathbf{x}_n}{\partial w_i} \right) \\ &= \frac{1}{N} \sum_{n=1}^N \left(\frac{1}{\square} \right) \left(\exp(\bigcirc) \right) \left(-y_n x_{n,i} \right) \\ &= \frac{1}{N} \sum_{n=1}^N \left(\frac{\exp(\bigcirc)}{1 + \exp(\bigcirc)} \right) \left(-y_n x_{n,i} \right) = \frac{1}{N} \sum_{n=1}^N \theta(\bigcirc) (-y_n x_{n,i}) \end{aligned}$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \theta(-y_n \mathbf{w}^T \mathbf{x}_n) (-y_n \mathbf{x}_n)$$

Minimizing $E_{\text{in}}(\mathbf{w})$

$$\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ln \left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n) \right)$$

want $\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \theta \left(-y_n \mathbf{w}^T \mathbf{x}_n \right) (-y_n \mathbf{x}_n) = \mathbf{0}$



scaled θ -weighted sum of $-y_n \mathbf{x}_n$

- all $\theta(\cdot) = 0$: only if $y_n \mathbf{w}^T \mathbf{x}_n \gg 0$
—linear separable \mathcal{D}
- weighted sum = $\mathbf{0}$:
non-linear equation of \mathbf{w}

closed-form solution? no :-(

PLA Revisited: Iterative Optimization

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

For $t = 0, 1, \dots$

- 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

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

when stop, return last \mathbf{w} as g

PLA Revisited: Iterative Optimization

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

For $t = 0, 1, \dots$

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

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

- ② (try to) correct the mistake by

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

- ① (equivalently) pick some n , and update \mathbf{w}_t by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + [\text{sign}(\mathbf{w}_t^T \mathbf{x}_n) \neq y_n] y_n \mathbf{x}_n$$

when stop, return last \mathbf{w} as g

PLA Revisited: Iterative Optimization

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

For $t = 0, 1, \dots$

- ① (equivalently) pick some n , and update \mathbf{w}_t by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \underbrace{\eta}_{\eta} \cdot \underbrace{\left([\text{sign}(\mathbf{w}_t^T \mathbf{x}_n) \neq y_n] \cdot y_n \mathbf{x}_n \right)}_{\mathbf{v}}$$

when stop, return last \mathbf{w} as g

choice of (η, \mathbf{v}) and stopping condition defines
iterative optimization approach

Fun Time

Consider the gradient $\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \theta(-y_n \mathbf{w}^T \mathbf{x}_n) (-y_n \mathbf{x}_n)$. That is, each example (\mathbf{x}_n, y_n) contributes to the gradient by an amount of $\theta(-y_n \mathbf{w}^T \mathbf{x}_n)$. For any given \mathbf{w} , which example contributes the most amount to the gradient?

- ① the example with the smallest $y_n \mathbf{w}^T \mathbf{x}_n$ value
- ② the example with the largest $y_n \mathbf{w}^T \mathbf{x}_n$ value
- ③ the example with the smallest $\mathbf{w}^T \mathbf{x}_n$ value
- ④ the example with the largest $\mathbf{w}^T \mathbf{x}_n$ value

Reference Answer: ①

Using the fact that θ is a monotonic function, we see that the example with the smallest $y_n \mathbf{w}^T \mathbf{x}_n$ value contributes to the gradient the most.

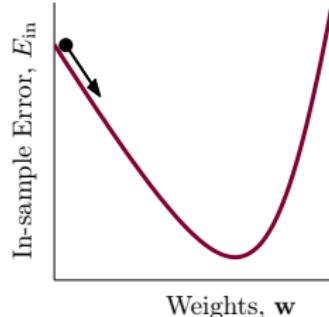
Iterative Optimization

For $t = 0, 1, \dots$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last \mathbf{w} as g

- PLA: \mathbf{v} comes from mistake correction
- smooth $E_{\text{in}}(\mathbf{w})$ for logistic regression:
choose \mathbf{v} to get the ball roll '**downhill**'?
 - direction \mathbf{v} :
(assumed) of unit length
 - step size η :
(assumed) positive



a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\underbrace{\mathbf{w}_t + \eta \mathbf{v}}_{\mathbf{w}_{t+1}})$$

Linear Approximation

a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\mathbf{w}_t + \eta \mathbf{v})$$

- still non-linear optimization, now **with constraints**
—not any easier than $\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w})$
- local approximation by linear formula makes problem easier

$$E_{\text{in}}(\mathbf{w}_t + \eta \mathbf{v}) \approx E_{\text{in}}(\mathbf{w}_t) + \eta \mathbf{v}^T \nabla E_{\text{in}}(\mathbf{w}_t)$$

if η really small (Taylor expansion)

an **approximate** greedy approach for some given **small** η :

$$\min_{\|\mathbf{v}\|=1} \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \underbrace{\mathbf{v}^T \nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

Gradient Descent

an **approximate** greedy approach for some given **small** η :

$$\min_{\|\mathbf{v}\|=1} \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

- optimal \mathbf{v} : opposite direction of $\nabla E_{\text{in}}(\mathbf{w}_t)$

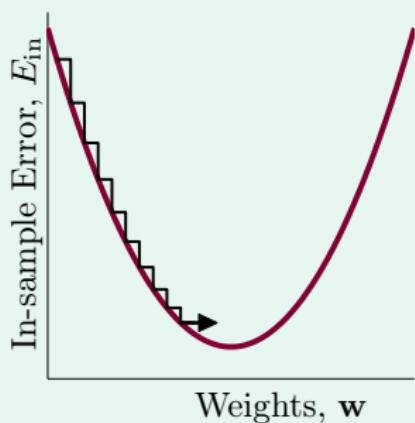
$$\mathbf{v} = - \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$$

- gradient descent: for **small** η , $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$

gradient descent:
a simple & popular optimization tool

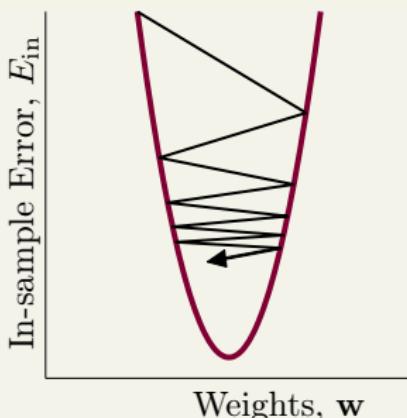
Choice of η

too small



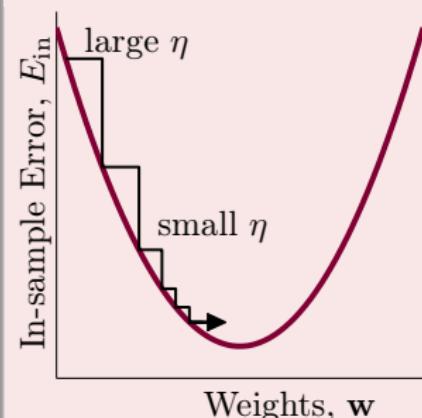
too slow :-)

too large



too unstable :-)

just right

use changing η

η better be **monotonic of $\|\nabla E_{in}(w_t)\|$**

Simple Heuristic for **Changing** η

η better be **monotonic** of $\|\nabla E_{\text{in}}(\mathbf{w}_t)\|$

- if red $\eta \propto \|\nabla E_{\text{in}}(\mathbf{w}_t)\|$ by ratio purple η

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$$

- call purple η the fixed learning rate

fixed learning rate gradient descent:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

Putting Everything Together

Logistic Regression Algorithm

initialize \mathbf{w}_0

For $t = 0, 1, \dots$

① compute

$$\nabla E_{\text{in}}(\mathbf{w}_t) = \frac{1}{N} \sum_{n=1}^N \theta(-y_n \mathbf{w}_t^T \mathbf{x}_n) (-y_n \mathbf{x}_n)$$

② update by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

...until $\nabla E_{\text{in}}(\mathbf{w}_{t+1}) = 0$ or enough iterations

return last \mathbf{w}_{t+1} as g

similar time complexity to **pocket** per iteration

Fun Time

If $\mathbf{w}_0 = \mathbf{0}$, and take $\eta = 0.1$. What is \mathbf{w}_1 in the logistic regression algorithm?

- ① $+0.1 \cdot \frac{1}{N} \sum_{n=1}^N y_n \mathbf{x}_n$
- ② $-0.1 \cdot \frac{1}{N} \sum_{n=1}^N y_n \mathbf{x}_n$
- ③ $+0.05 \cdot \frac{1}{N} \sum_{n=1}^N y_n \mathbf{x}_n$
- ④ $-0.05 \cdot \frac{1}{N} \sum_{n=1}^N y_n \mathbf{x}_n$

Reference Answer: ③

You can do a simple substitution using the fact that $\theta(0) = \frac{1}{2}$. This result shows that a scaled average of $y_n \mathbf{x}_n$ is somewhat ‘one-step’ better than the zero vector.

Summary

- ① When Can Machines Learn?
- ② Why Can Machines Learn?
- ③ How Can Machines Learn?

Lecture 9: Linear Regression

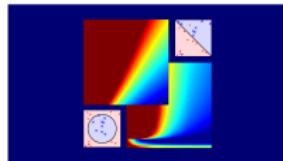
Lecture 10: Logistic Regression

- Logistic Regression Problem
 $P(+1|x)$ as target and $\theta(w^T x)$ as hypotheses
- Logistic Regression Error
cross-entropy (negative log likelihood)
- Gradient of Logistic Regression Error
 θ -weighted sum of data vectors
- Gradient Descent
roll downhill by $-\nabla E_{\text{in}}(w)$

- next: linear model 'S' for classification

- ④ How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 11: Linear Models for Classification

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?
- ② Why Can Machines Learn?
- ③ **How** Can Machines Learn?

Lecture 10: Logistic Regression

gradient descent on **cross-entropy error**
to get good **logistic hypothesis**

Lecture 11: Linear Models for Classification

- Linear Models for Binary Classification
- Stochastic Gradient Descent
- Multiclass via Logistic Regression
- Multiclass via Binary Classification

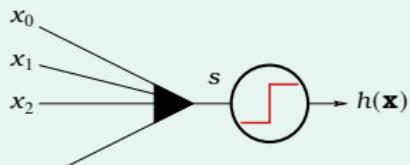
- ④ How Can Machines Learn Better?

Linear Models Revisited

linear scoring function: $s = \mathbf{w}^T \mathbf{x}$

linear classification

$$h(\mathbf{x}) = \text{sign}(s)$$

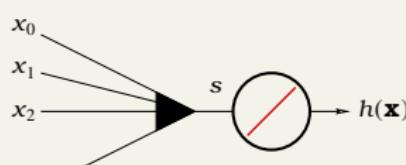


plausible err = 0/1

discrete $E_{in}(\mathbf{w})$:
NP-hard to solve

linear regression

$$h(\mathbf{x}) = s$$

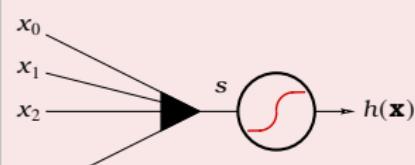


friendly err = squared

quadratic convex $E_{in}(\mathbf{w})$:
closed-form solution

logistic regression

$$h(\mathbf{x}) = \theta(s)$$



plausible err = cross-entropy

smooth convex $E_{in}(\mathbf{w})$:
gradient descent

can linear regression or logistic regression
help linear classification?

Error Functions Revisited

linear scoring function: $s = \mathbf{w}^T \mathbf{x}$

for binary classification $y \in \{-1, +1\}$

linear classification

$$\begin{aligned} h(\mathbf{x}) &= \text{sign}(s) \\ \text{err}(h, \mathbf{x}, y) &= \llbracket h(\mathbf{x}) \neq y \rrbracket \end{aligned}$$

$$\begin{aligned} &\text{err}_{0/1}(s, y) \\ &= \llbracket \text{sign}(s) \neq y \rrbracket \\ &= \llbracket \text{sign}(ys) \neq 1 \rrbracket \end{aligned}$$

linear regression

$$\begin{aligned} h(\mathbf{x}) &= s \\ \text{err}(h, \mathbf{x}, y) &= (h(\mathbf{x}) - y)^2 \end{aligned}$$

$$\begin{aligned} &\text{err}_{\text{SQR}}(s, y) \\ &= (s - y)^2 \\ &= (ys - 1)^2 \end{aligned}$$

logistic regression

$$\begin{aligned} h(\mathbf{x}) &= \theta(s) \\ \text{err}(h, \mathbf{x}, y) &= -\ln h(y\mathbf{x}) \end{aligned}$$

$$\begin{aligned} &\text{err}_{\text{CE}}(s, y) \\ &= \ln(1 + \exp(-ys)) \end{aligned}$$

(ys) : classification correctness score

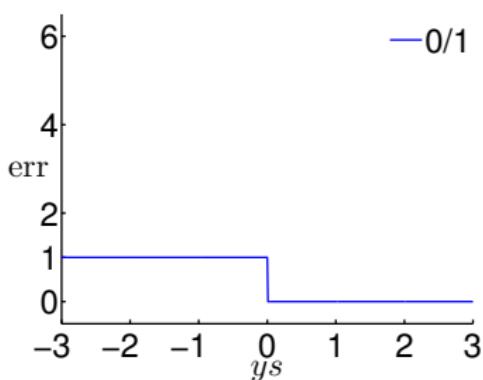
Visualizing Error Functions

$$0/1 \text{ err}_{0/1}(s, y) = \|\text{sign}(ys) \neq 1\|$$

$$\text{sqr err}_{\text{SQR}}(s, y) = (ys - 1)^2$$

$$\text{ce err}_{\text{CE}}(s, y) = \ln(1 + \exp(-ys))$$

$$\text{scaled ce err}_{\text{SCE}}(s, y) = \log_2(1 + \exp(-ys))$$



- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
but over-charge $ys \gg 1$
small $\text{err}_{\text{SQR}} \rightarrow$ small $\text{err}_{0/1}$
- ce: monotonic of ys
small $\text{err}_{\text{CE}} \leftrightarrow$ small $\text{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
small $\text{err}_{\text{SCE}} \leftrightarrow$ small $\text{err}_{0/1}$

upper bound:
useful for designing algorithmic error $\widehat{\text{err}}$

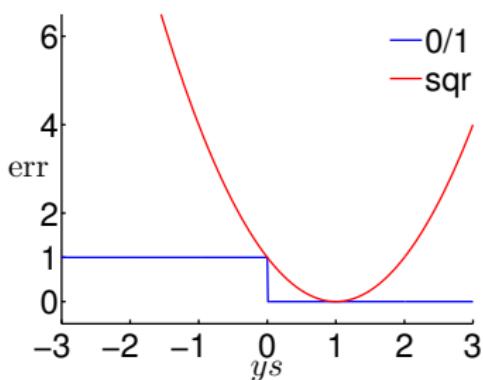
Visualizing Error Functions

$$0/1 \text{ err}_{0/1}(s, y) = \|\text{sign}(ys) \neq 1\|$$

$$\text{sqr } \text{err}_{\text{SQR}}(s, y) = (ys - 1)^2$$

$$\text{ce } \text{err}_{\text{CE}}(s, y) = \ln(1 + \exp(-ys))$$

$$\text{scaled ce } \text{err}_{\text{SCE}}(s, y) = \log_2(1 + \exp(-ys))$$



- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
but over-charge $ys \gg 1$
small $\text{err}_{\text{SQR}} \rightarrow$ small $\text{err}_{0/1}$
- ce: monotonic of ys
small $\text{err}_{\text{CE}} \leftrightarrow$ small $\text{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
small $\text{err}_{\text{SCE}} \leftrightarrow$ small $\text{err}_{0/1}$

upper bound:
useful for designing algorithmic error $\widehat{\text{err}}$

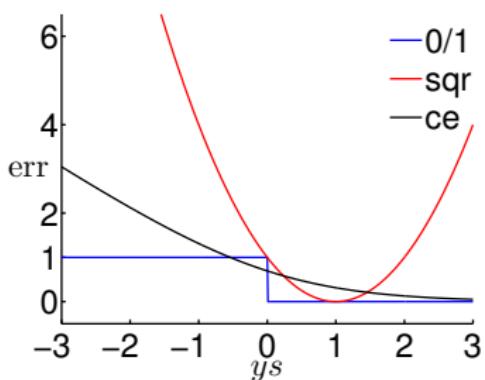
Visualizing Error Functions

$$0/1 \text{ err}_{0/1}(s, y) = \|\text{sign}(ys) \neq 1\|$$

$$\text{sqr } \text{err}_{\text{SQR}}(s, y) = (ys - 1)^2$$

$$\text{ce } \text{err}_{\text{CE}}(s, y) = \ln(1 + \exp(-ys))$$

$$\text{scaled ce } \text{err}_{\text{SCE}}(s, y) = \log_2(1 + \exp(-ys))$$



- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
but over-charge $ys \gg 1$
small $\text{err}_{\text{SQR}} \rightarrow$ small $\text{err}_{0/1}$
- ce: monotonic of ys
small $\text{err}_{\text{CE}} \leftrightarrow$ small $\text{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
small $\text{err}_{\text{SCE}} \leftrightarrow$ small $\text{err}_{0/1}$

upper bound:
useful for designing algorithmic error $\widehat{\text{err}}$

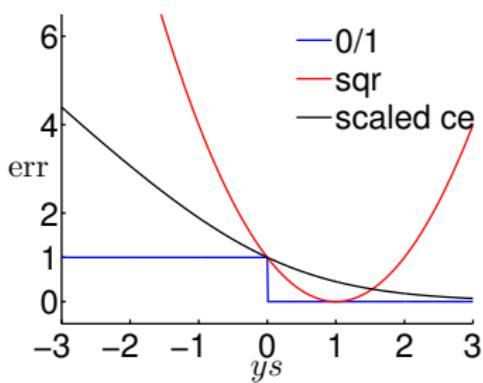
Visualizing Error Functions

$$0/1 \text{ err}_{0/1}(s, y) = \|\text{sign}(ys) \neq 1\|$$

$$\text{sqr } \text{err}_{\text{SQR}}(s, y) = (ys - 1)^2$$

$$\text{ce } \text{err}_{\text{CE}}(s, y) = \ln(1 + \exp(-ys))$$

$$\text{scaled ce } \text{err}_{\text{SCE}}(s, y) = \log_2(1 + \exp(-ys))$$



- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
but over-charge $ys \gg 1$
small $\text{err}_{\text{SQR}} \rightarrow$ small $\text{err}_{0/1}$
- ce: monotonic of ys
small $\text{err}_{\text{CE}} \leftrightarrow$ small $\text{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
small $\text{err}_{\text{SCE}} \leftrightarrow$ small $\text{err}_{0/1}$

upper bound:
useful for designing algorithmic error $\widehat{\text{err}}$

Theoretical Implication of Upper Bound

For any ys where $s = \mathbf{w}^T \mathbf{x}$

$$\text{err}_{0/1}(s, y) \leq \text{err}_{\text{SCE}}(s, y) = \frac{1}{\ln 2} \text{err}_{\text{CE}}(s, y).$$

$$\implies E_{\text{in}}^{0/1}(\mathbf{w}) \leq E_{\text{in}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w})$$

$$E_{\text{out}}^{0/1}(\mathbf{w}) \leq E_{\text{out}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(\mathbf{w})$$

VC on 0/1:

$$\begin{aligned} E_{\text{out}}^{0/1}(\mathbf{w}) &\leq E_{\text{in}}^{0/1}(\mathbf{w}) + \Omega^{0/1} \\ &\leq \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w}) + \Omega^{0/1} \end{aligned}$$

VC-Reg on CE :

$$\begin{aligned} E_{\text{out}}^{0/1}(\mathbf{w}) &\leq \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(\mathbf{w}) \\ &\leq \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w}) + \frac{1}{\ln 2} \Omega^{\text{CE}} \end{aligned}$$

small $E_{\text{in}}^{\text{CE}}(\mathbf{w}) \implies$ small $E_{\text{out}}^{0/1}(\mathbf{w})$:
logistic/linear reg. for linear classification

Regression for Classification

- ① run **logistic/linear reg.** on \mathcal{D} with $y_n \in \{-1, +1\}$ to get \mathbf{w}_{REG}
- ② return $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{REG}}^T \mathbf{x})$

PLA

- pros: **efficient + strong guarantee if lin. separable**
- cons: works only if lin. separable, otherwise needing **pocket** heuristic

linear regression

- pros: **'easiest' optimization**
- cons: loose bound of $\text{err}_{0/1}$ for large $|ys|$

logistic regression

- pros: **'easy' optimization**
- cons: loose bound of $\text{err}_{0/1}$ for very negative ys

- **linear regression** sometimes used to set \mathbf{w}_0 for **PLA/pocket/logistic regression**
- **logistic regression** often preferred over **pocket**

Fun Time

Following the definition in the lecture, which of the following is not always $\geq \text{err}_{0/1}(y, s)$ when $y \in \{-1, +1\}$?

- 1 $\text{err}_{0/1}(y, s)$
- 2 $\text{err}_{\text{SQR}}(y, s)$
- 3 $\text{err}_{\text{CE}}(y, s)$
- 4 $\text{err}_{\text{SCE}}(y, s)$

Reference Answer: 3

Too simple, uh? :-)
Anyway, note that $\text{err}_{0/1}$ is surely an upper bound of itself.

Two Iterative Optimization Schemes

For $t = 0, 1, \dots$

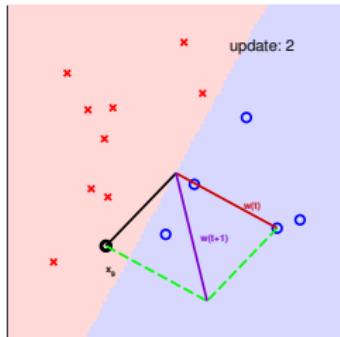
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last \mathbf{w} as g

PLA

pick (\mathbf{x}_n, y_n) and decide \mathbf{w}_{t+1} by
the one example

$O(1)$ time per iteration :-)



logistic regression (pocket)

check \mathcal{D} and decide \mathbf{w}_{t+1} (or
new $\hat{\mathbf{w}}$) by all examples

$O(N)$ time per iteration :-()

logistic regression with
 $O(1)$ time per iteration?

Logistic Regression Revisited

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \underbrace{\frac{1}{N} \sum_{n=1}^N \theta \left(-y_n \mathbf{w}_t^T \mathbf{x}_n \right)}_{-\nabla E_{\text{in}}(\mathbf{w}_t)} (\mathbf{y}_n \mathbf{x}_n)$$

- want: update direction $\mathbf{v} \approx -\nabla E_{\text{in}}(\mathbf{w}_t)$
while computing \mathbf{v} by one single (\mathbf{x}_n, y_n)
- technique on removing $\frac{1}{N} \sum_{n=1}^N$:
view as expectation \mathcal{E} over uniform choice of n !

stochastic gradient:

$\nabla_{\mathbf{w}} \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$ with random n

true gradient:

$$\nabla_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \underset{\text{random } n}{\mathcal{E}} \nabla_{\mathbf{w}} \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$$

Stochastic Gradient Descent (SGD)

stochastic gradient = true gradient + zero-mean ‘noise’ directions

Stochastic Gradient Descent

- idea: replace true gradient by stochastic gradient
- after enough steps,
average true gradient \approx average stochastic gradient
- pros: simple & cheaper computation :-)
 - useful for big data or online learning
- cons: less stable in nature

SGD logistic regression, looks familiar? :-):

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \underbrace{\left(-y_n \mathbf{w}_t^T \mathbf{x}_n \right) \left(y_n \mathbf{x}_n \right)}_{-\nabla \text{err}(\mathbf{w}_t, \mathbf{x}_n, y_n)}$$

PLA Revisited

SGD logistic regression:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \theta \left(-y_n \mathbf{w}_t^T \mathbf{x}_n \right) (\mathbf{y}_n \mathbf{x}_n)$$

PLA:

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

- SGD logistic regression \approx 'soft' PLA
- PLA \approx SGD logistic regression with $\eta = 1$ when $\mathbf{w}_t^T \mathbf{x}_n$ large

two practical rule-of-thumb:

- stopping condition? t large enough
- η ? 0.1 when \mathbf{x} in proper range

Fun Time

Consider applying SGD on linear regression for big data. What is the update direction when using the negative stochastic gradient?

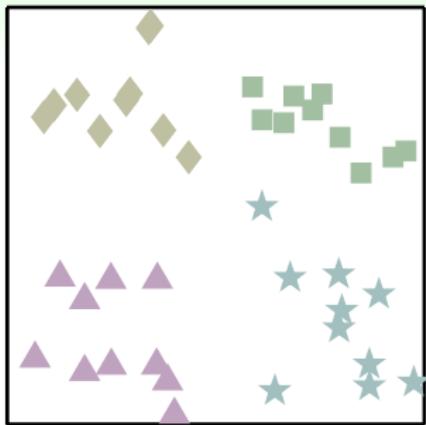
- ① \mathbf{x}_n
- ② $y_n \mathbf{x}_n$
- ③ $2(\mathbf{w}_t^T \mathbf{x}_n - y_n) \mathbf{x}_n$
- ④ $2(y_n - \mathbf{w}_t^T \mathbf{x}_n) \mathbf{x}_n$

Reference Answer: ④

Go check lecture 9 if you have forgotten about the gradient of squared error. :-)

Anyway, the update rule has a nice physical interpretation: improve \mathbf{w}_t by ‘correcting’ proportional to the residual $(y_n - \mathbf{w}_t^T \mathbf{x}_n)$.

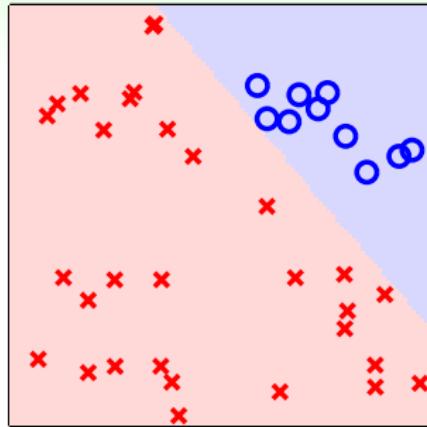
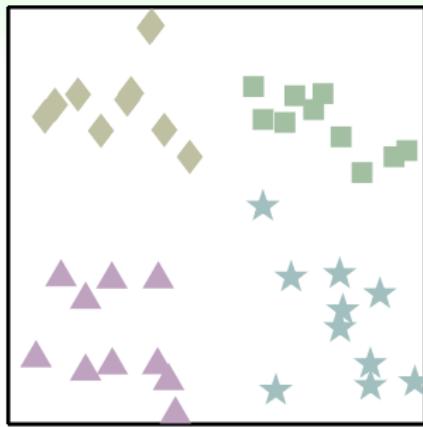
Multiclass Classification



- $\mathcal{Y} = \{\square, \diamond, \triangle, \star\}$
(4-class classification)
- **many applications** in practice, especially for 'recognition'

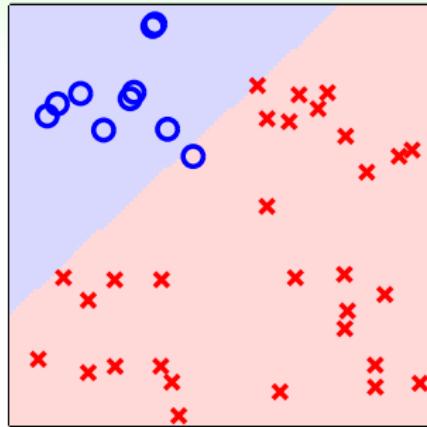
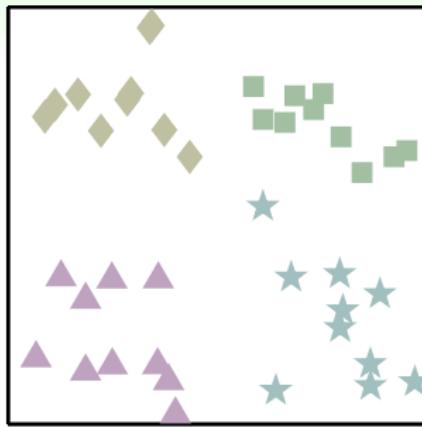
next: use **tools for $\{\times, \circ\}$ classification** to
 $\{\square, \diamond, \triangle, \star\}$ classification

One Class at a Time



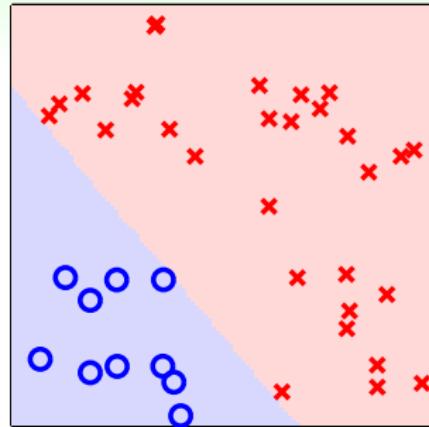
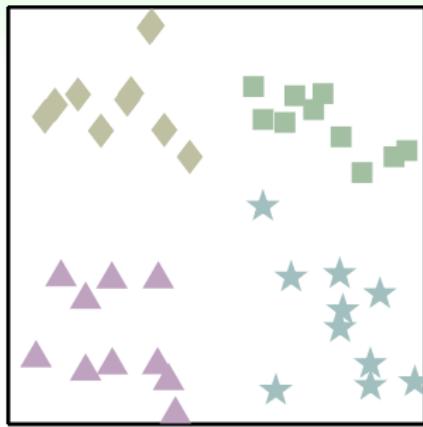
□ or not? {□ = o, ◇ = x, △ = x, ⋆ = x}

One Class at a Time



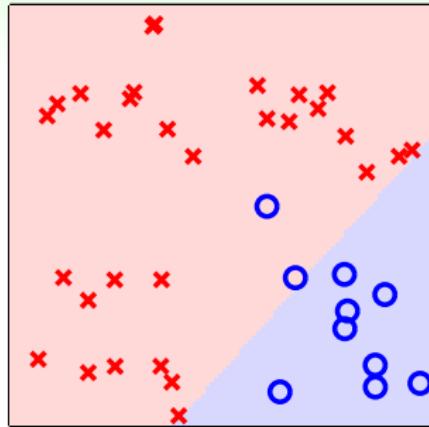
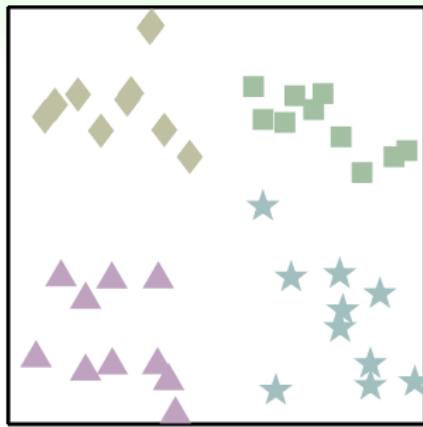
◊ or not? {□ = x, ◊ = o, △ = x, ★ = x}

One Class at a Time



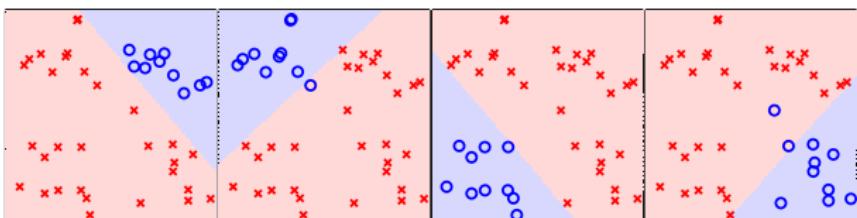
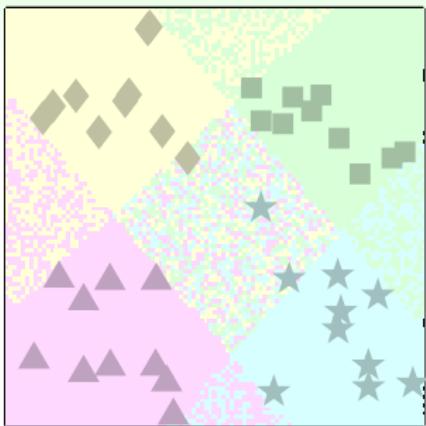
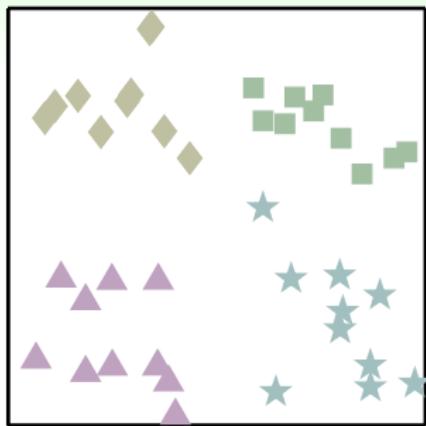
\triangle or not? $\{\square = \times, \diamond = \times, \triangle = \circ, \star = \times\}$

One Class at a Time



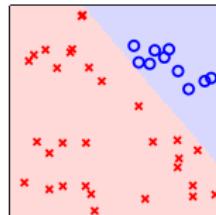
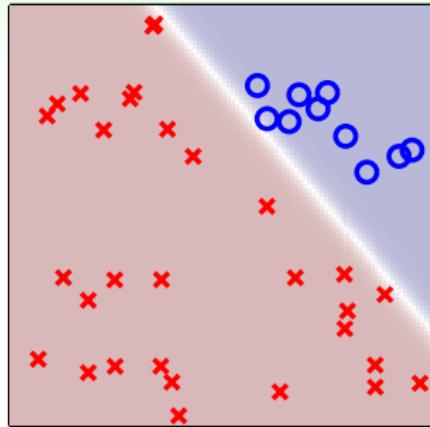
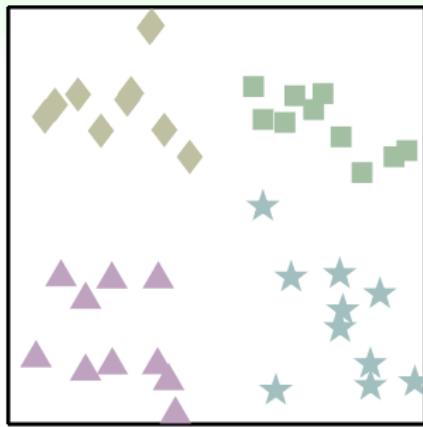
★ or not? $\{\square = \text{x}, \diamond = \text{x}, \triangle = \text{x}, \star = \text{o}\}$

Multiclass Prediction: Combine Binary Classifiers



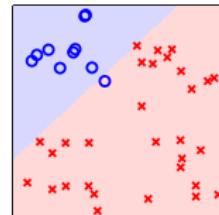
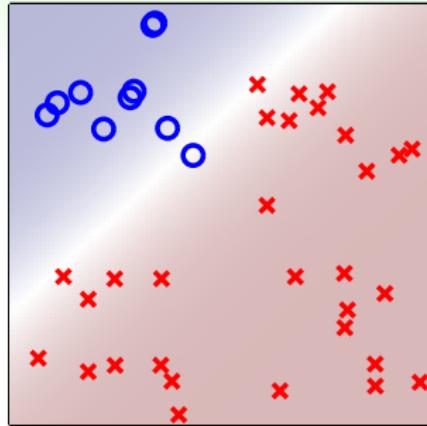
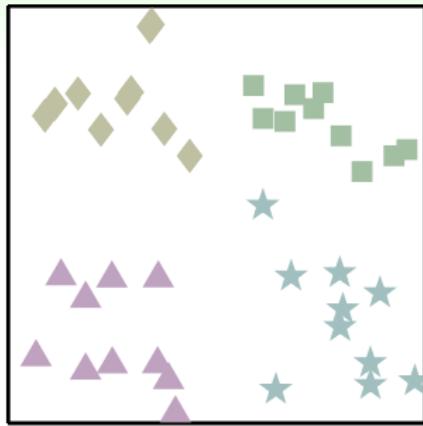
but ties? :-)

One Class at a Time **Softly**



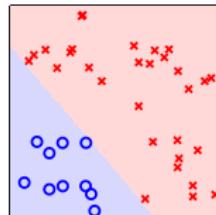
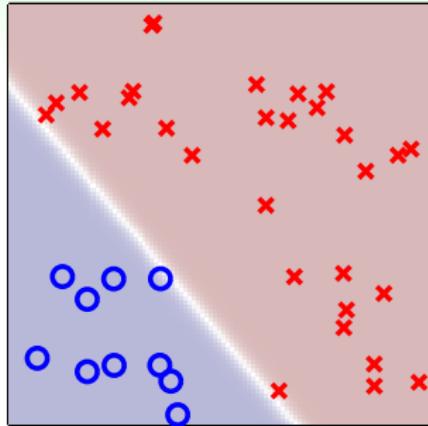
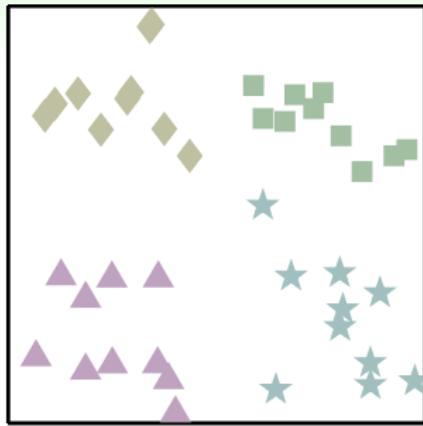
$$P(\square|x) ? \{ \square = o, \diamond = x, \triangle = x, \star = x \}$$

One Class at a Time **Softly**



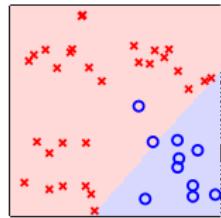
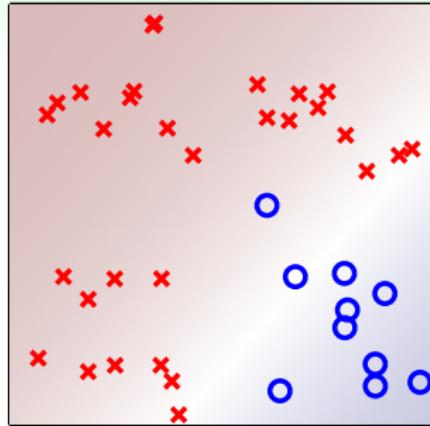
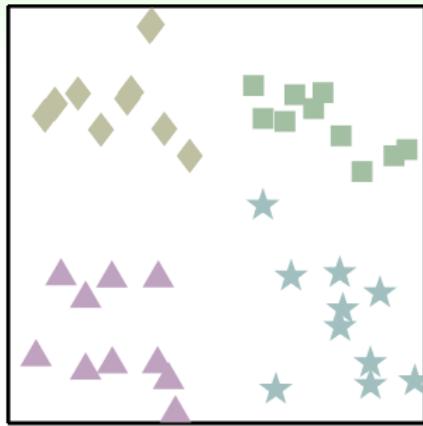
$$P(\diamond|x) ? \{ \square = \times, \diamond = \circ, \triangle = \times, \star = \times \}$$

One Class at a Time **Softly**



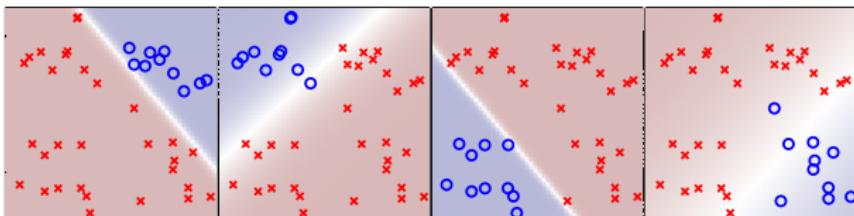
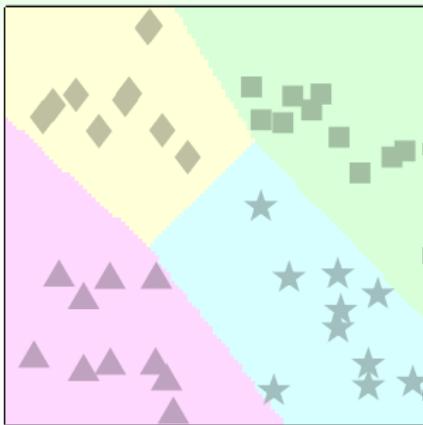
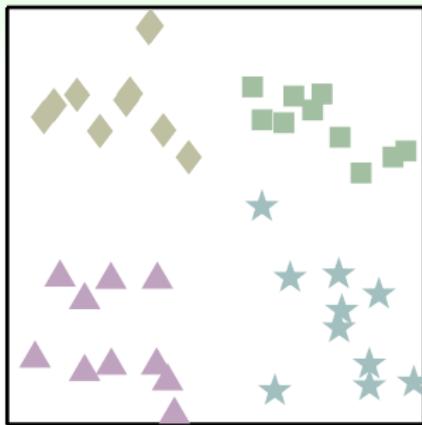
$$P(\Delta | \mathbf{x})? \{ \square = \times, \diamond = \times, \triangle = \circ, \star = \times \}$$

One Class at a Time **Softly**



$P(\star|\mathbf{x})?$ $\{\square = \times, \diamond = \times, \triangle = \times, \star = \circ\}$

Multiclass Prediction: Combine **Soft** Classifiers



$$g(\mathbf{x}) = \operatorname{argmax}_{k \in \mathcal{Y}} \theta(\mathbf{w}_{[k]}^T \mathbf{x})$$

One-Versus-All (OVA) Decomposition

1 for $k \in \mathcal{Y}$

obtain $\mathbf{w}_{[k]}$ by running logistic regression on

$$\mathcal{D}_{[k]} = \{(\mathbf{x}_n, y'_n = 2[\![y_n = k]\!]-1)\}_{n=1}^N$$

2 return $g(\mathbf{x}) = \operatorname{argmax}_{k \in \mathcal{Y}} (\mathbf{w}_{[k]}^T \mathbf{x})$

- pros: efficient,
can be coupled with any logistic regression-like approaches
- cons: often unbalanced $\mathcal{D}_{[k]}$ when K large
- extension: multinomial ('coupled') logistic regression

OVA: a simple multiclass **meta-algorithm**
to keep in your toolbox

Fun Time

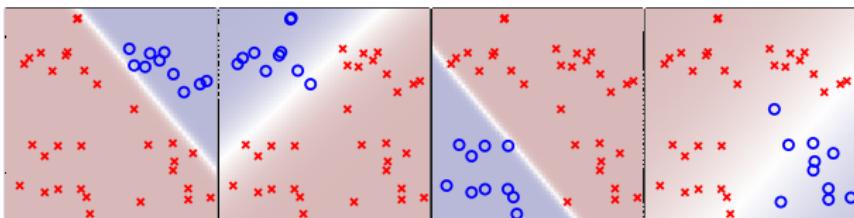
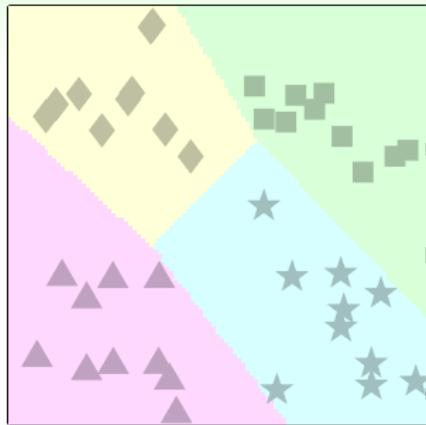
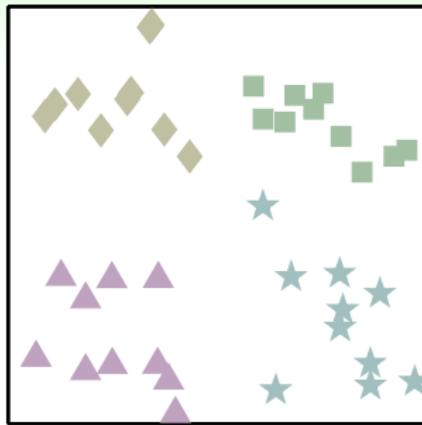
Which of the following best describes the training effort of OVA decomposition based on logistic regression on some K -class classification data of size N ?

- ① learn K logistic regression hypotheses, each from data of size N/K
- ② learn K logistic regression hypotheses, each from data of size $N \ln K$
- ③ learn K logistic regression hypotheses, each from data of size N
- ④ learn K logistic regression hypotheses, each from data of size NK

Reference Answer: ③

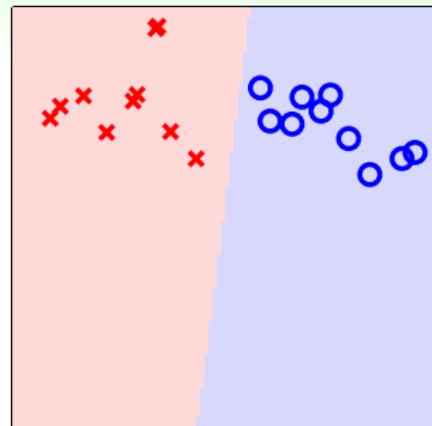
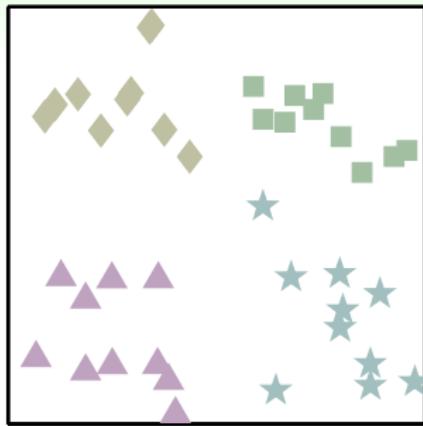
Note that the **learning part can be easily done in parallel**, while the data is essentially of the same size as the original data.

Source of Unbalance: One versus All



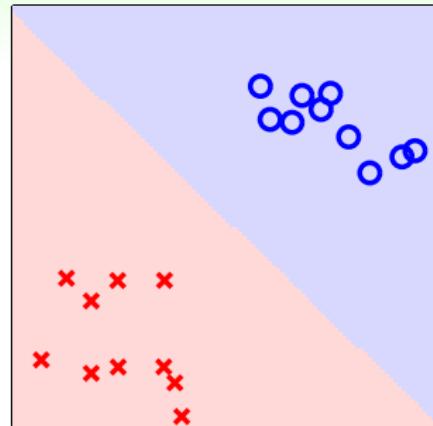
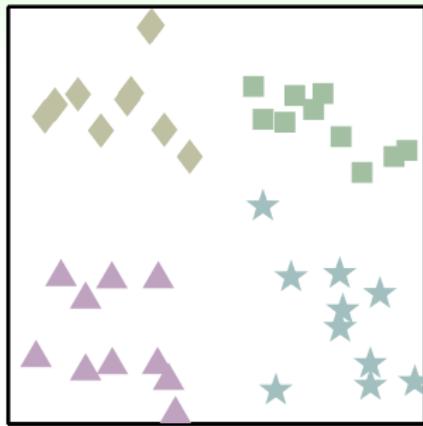
idea: make binary classification problems
more **balanced** by one versus **one**

One versus One at a Time



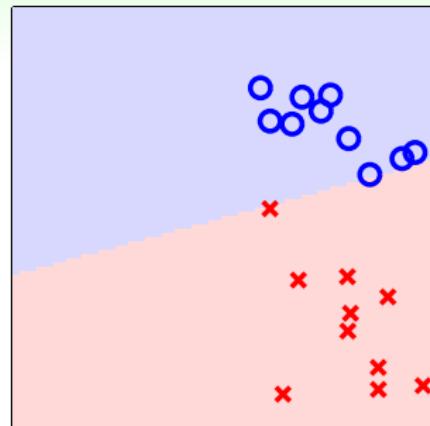
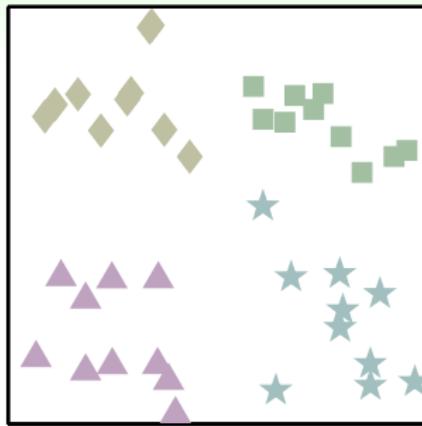
or ? { = o, = x, = nil, = nil}

One versus One at a Time



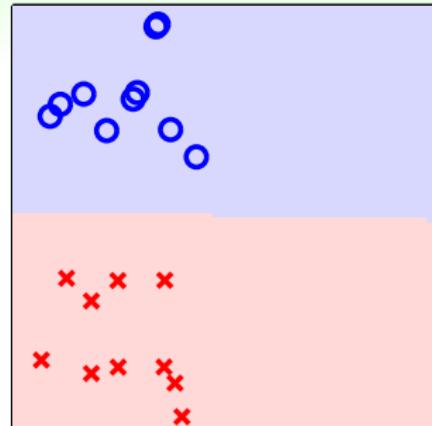
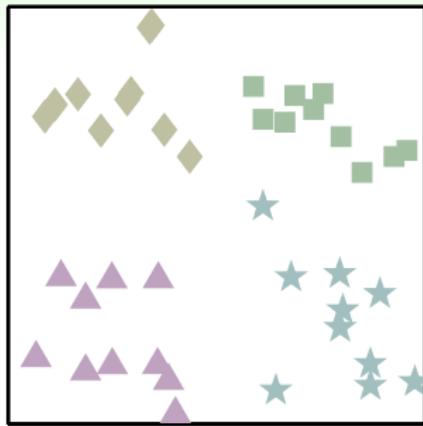
□ or △? {□ = o, ◇ = nil, △ = x, ★ = nil}

One versus One at a Time



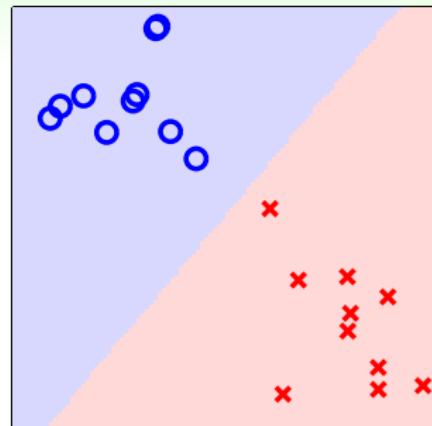
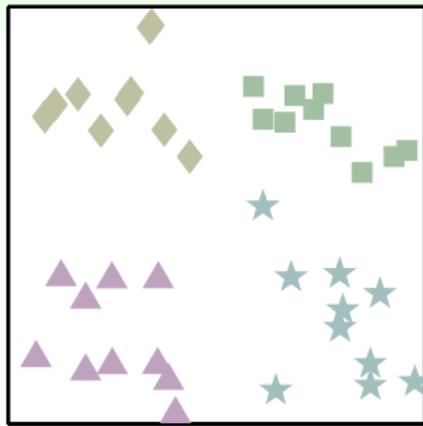
□ or ⋆? {□ = o, ◊ = nil, △ = nil, ⋆ = x}

One versus One at a Time



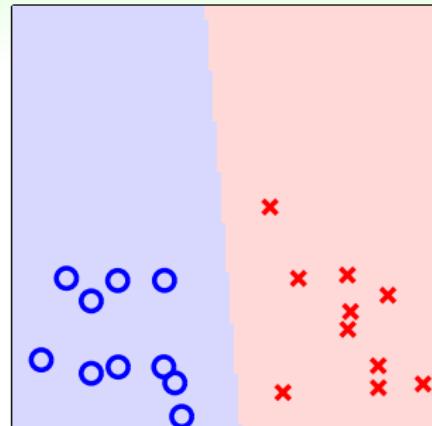
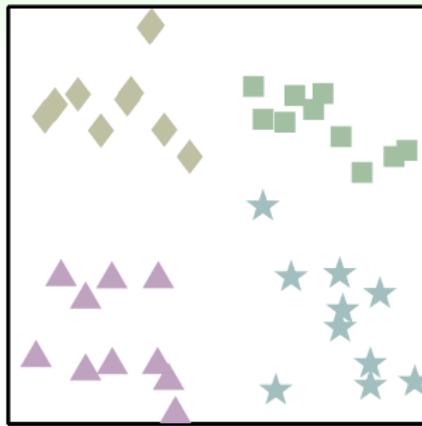
◊ or △? {□ = nil, ◊ = o, △ = x, ★ = nil}

One versus One at a Time



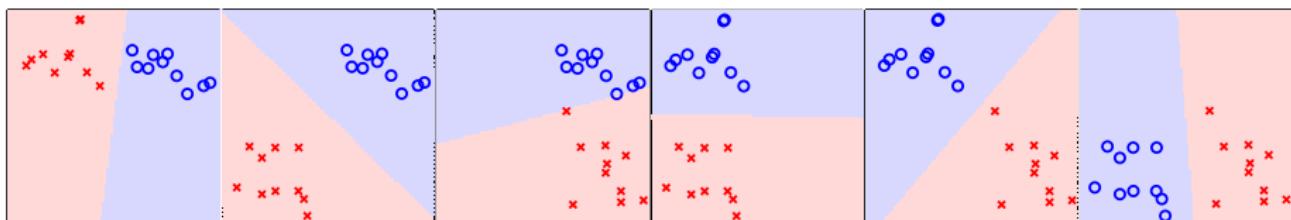
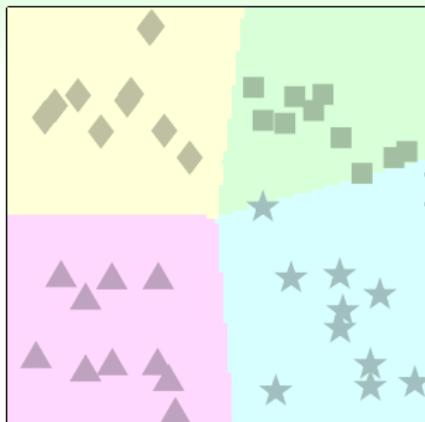
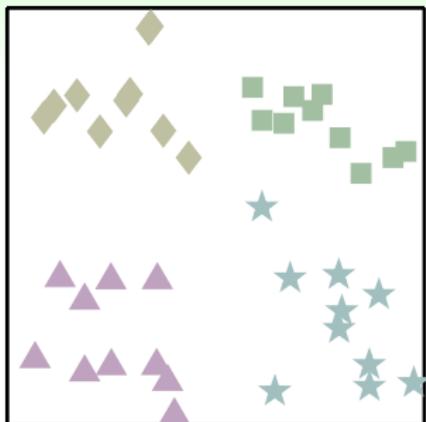
◊ or ⋆? {□ = nil, ◊ = ○, △ = nil, ⋆ = ✗}

One versus One at a Time



\triangle or \star ? $\{\square = \text{nil}, \diamond = \text{nil}, \triangle = \circ, \star = \times\}$

Multiclass Prediction: Combine **Pairwise** Classifiers



$g(\mathbf{x}) = \text{tournament champion } \left\{ \mathbf{w}_{[k,\ell]}^T \mathbf{x} \right\}$
(voting of classifiers)

One-versus-one (OVO) Decomposition

- 1 for $(k, \ell) \in \mathcal{Y} \times \mathcal{Y}$
obtain $\mathbf{w}_{[k,\ell]}$ by running linear binary classification on
$$\mathcal{D}_{[k,\ell]} = \{(\mathbf{x}_n, y'_n = 2[\![y_n = k]\!] - 1) : y_n = k \text{ or } y_n = \ell\}$$
 - 2 return $g(\mathbf{x}) = \text{tournament champion} \left\{ \mathbf{w}_{[k,\ell]}^T \mathbf{x} \right\}$
- pros: efficient ('smaller' training problems), stable,
can be coupled with any binary classification approaches
 - cons: use $O(K^2)$ $\mathbf{w}_{[k,\ell]}$
—**more space, slower prediction, more training**

OVO: another simple multiclass
meta-algorithm to keep in your toolbox

Fun Time

Assume that some binary classification algorithm takes exactly N^3 CPU-seconds for data of size N . Also, for some 10-class multiclass classification problem, assume that there are $N/10$ examples for each class. Which of the following is total CPU-seconds needed for OVO decomposition based on the binary classification algorithm?

- 1 $\frac{9}{200}N^3$
- 2 $\frac{9}{25}N^3$
- 3 $\frac{4}{5}N^3$
- 4 N^3

Reference Answer: 2

There are 45 binary classifiers, each trained with data of size $(2N)/10$. Note that OVA decomposition with the same algorithm would take $10N^3$ time, much worse than OVO.

Summary

- 1 When Can Machines Learn?
- 2 Why Can Machines Learn?
- 3 **How** Can Machines Learn?

Lecture 10: Logistic Regression

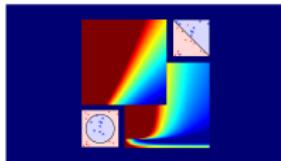
Lecture 11: Linear Models for Classification

- Linear Models for Binary Classification
three models useful in different ways
- Stochastic Gradient Descent
follow negative stochastic gradient
- Multiclass via Logistic Regression
predict with maximum estimated $P(k|x)$
- Multiclass via Binary Classification
predict the tournament champion

- **next: from linear to nonlinear**

- 4 How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 12: Nonlinear Transformation

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?
- ② Why Can Machines Learn?
- ③ **How** Can Machines Learn?

Lecture 11: Linear Models for Classification

binary classification via **(logistic) regression**;
multiclass via **OVA/OVO decomposition**

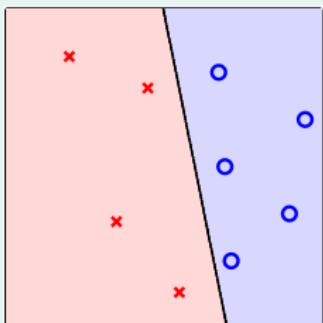
Lecture 12: Nonlinear Transformation

- Quadratic Hypotheses
- Nonlinear Transform
- Price of Nonlinear Transform
- Structured Hypothesis Sets

- ④ How Can Machines Learn Better?

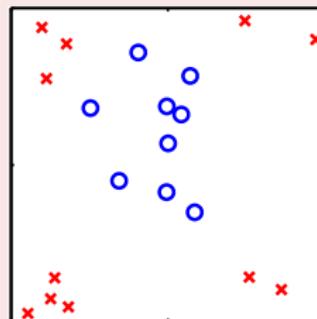
Linear Hypotheses

up to now: linear hypotheses



- visually: 'line'-like boundary
- mathematically: linear scores $s = \mathbf{w}^T \mathbf{x}$

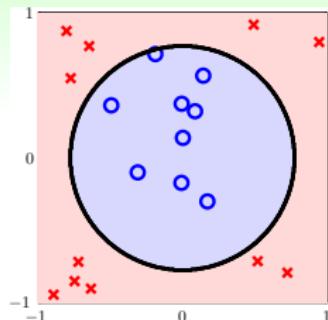
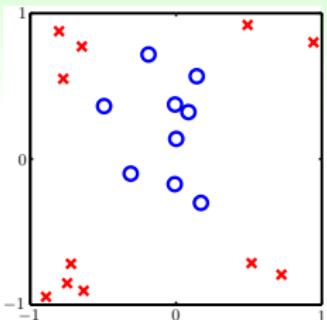
but limited ...



- theoretically: d_{VC} under control :-)
- practically: on some \mathcal{D} , large E_{in} for every line :-(

how to break the limit of linear hypotheses

Circular Separable



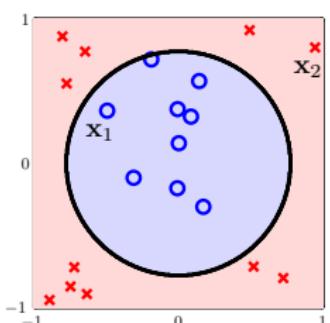
- \mathcal{D} not linear separable
- but **circular separable** by a circle of radius $\sqrt{0.6}$ centered at origin:

$$h_{\text{SEP}}(\mathbf{x}) = \text{sign}(-x_1^2 - x_2^2 + 0.6)$$

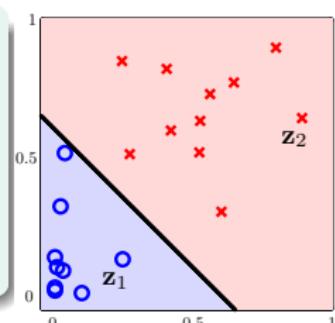
re-derive **Circular**-PLA, **Circular**-Regression,
blahblah ... all over again? :-)

Circular Separable and Linear Separable

$$\begin{aligned}
 h(\mathbf{x}) &= \text{sign} \left(\underbrace{0.6}_{\tilde{w}_0} \cdot \underbrace{1}_{z_0} + \underbrace{(-1)}_{\tilde{w}_1} \cdot \underbrace{x_1^2}_{z_1} + \underbrace{(-1)}_{\tilde{w}_2} \cdot \underbrace{x_2^2}_{z_2} \right) \\
 &= \text{sign} \left(\tilde{\mathbf{w}}^T \mathbf{z} \right)
 \end{aligned}$$



- $\{(\mathbf{x}_n, y_n)\}$ circular separable
 $\Rightarrow \{(\mathbf{z}_n, y_n)\}$ linear separable
- $\mathbf{x} \in \mathcal{X} \xrightarrow{\Phi} \mathbf{z} \in \mathcal{Z}$:
(nonlinear) feature transform Φ



circular separable in $\mathcal{X} \Rightarrow$ linear separable in \mathcal{Z}
vice versa?

Linear Hypotheses in \mathcal{Z} -Space

$$(z_0, z_1, z_2) = \mathbf{z} = \Phi(\mathbf{x}) = (1, x_1^2, x_2^2)$$

$$h(\mathbf{x}) = \tilde{h}(\mathbf{z}) = \text{sign} \left(\tilde{\mathbf{w}}^T \Phi(\mathbf{x}) \right) = \text{sign} \left(\tilde{w}_0 + \tilde{w}_1 x_1^2 + \tilde{w}_2 x_2^2 \right)$$

$$\tilde{\mathbf{w}} = (\tilde{w}_0, \tilde{w}_1, \tilde{w}_2)$$

- $(0.6, -1, -1)$: circle (○ inside)
- $(-0.6, +1, +1)$: circle (○ outside)
- $(0.6, -1, -2)$: ellipse
- $(0.6, -1, +2)$: hyperbola
- $(0.6, +1, +2)$: **constant** ○ :-)

lines in \mathcal{Z} -space
 \iff **special** quadratic curves in \mathcal{X} -space

General Quadratic Hypothesis Set

a ‘bigger’ \mathcal{Z} -space with $\Phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

perceptrons in \mathcal{Z} -space \iff quadratic hypotheses in \mathcal{X} -space

$$\mathcal{H}_{\Phi_2} = \left\{ h(\mathbf{x}) : h(\mathbf{x}) = \tilde{h}(\Phi_2(\mathbf{x})) \text{ for some linear } \tilde{h} \text{ on } \mathcal{Z} \right\}$$

- can implement all possible quadratic curve boundaries: circle, ellipse, rotated ellipse, hyperbola, parabola, ...

$$\text{ellipse } 2(x_1 + x_2 - 3)^2 + (x_1 - x_2 - 4)^2 = 1$$

$$\iff \tilde{\mathbf{w}}^T = [33, -20, -4, 3, 2, 3]$$

- include lines and constants as degenerate cases

next: learn a good quadratic hypothesis g

Fun Time

Using the transform $\Phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$, which of the following weights $\tilde{\mathbf{w}}^T$ in the \mathcal{Z} -space implements the parabola $2x_1^2 + x_2 = 1$?

- ① $[-1, 2, 1, 0, 0, 0]$
- ② $[0, 2, 1, 0, -1, 0]$
- ③ $[-1, 0, 1, 2, 0, 0]$
- ④ $[-1, 2, 0, 0, 0, 1]$

Fun Time

Using the transform $\Phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$, which of the following weights $\tilde{\mathbf{w}}^T$ in the \mathcal{Z} -space implements the parabola $2x_1^2 + x_2 = 1$?

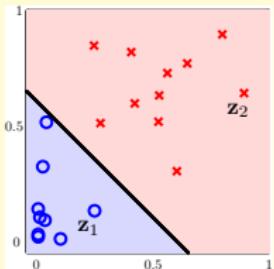
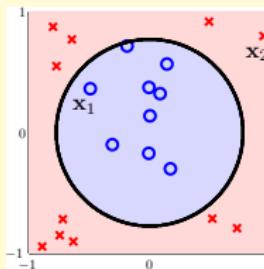
- ① $[-1, 2, 1, 0, 0, 0]$
- ② $[0, 2, 1, 0, -1, 0]$
- ③ $[-1, 0, 1, 2, 0, 0]$
- ④ $[-1, 2, 0, 0, 0, 1]$

Reference Answer: ③

Too simple, uh? :-) Flexibility to implement arbitrary quadratic curves opens new possibilities for minimizing E_{in} !

Good Quadratic Hypothesis

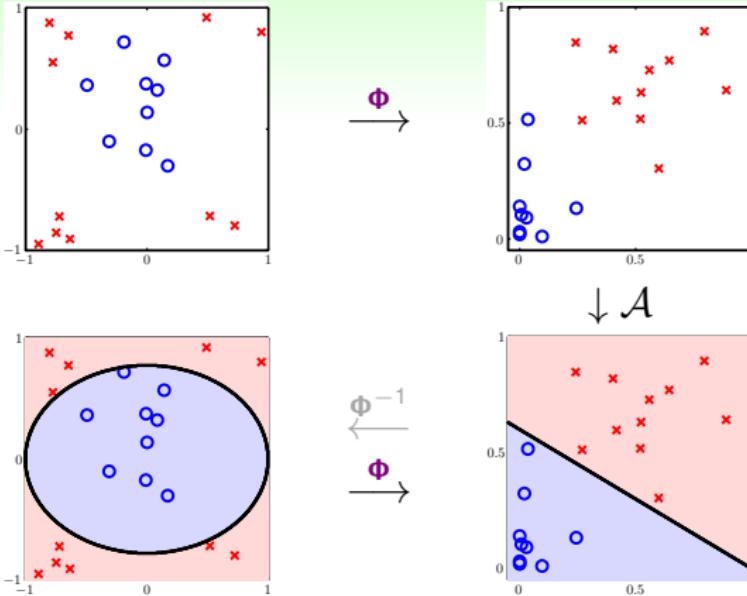
\mathcal{Z} -space	\iff	\mathcal{X} -space
perceptrons	\iff	quadratic hypotheses
good perceptron	\iff	good quadratic hypothesis
separating perceptron	\iff	separating quadratic hypothesis


 \iff


- want: get **good perceptron** in \mathcal{Z} -space
- known: get **good perceptron** in \mathcal{X} -space with data $\{(\mathbf{x}_n, y_n)\}$

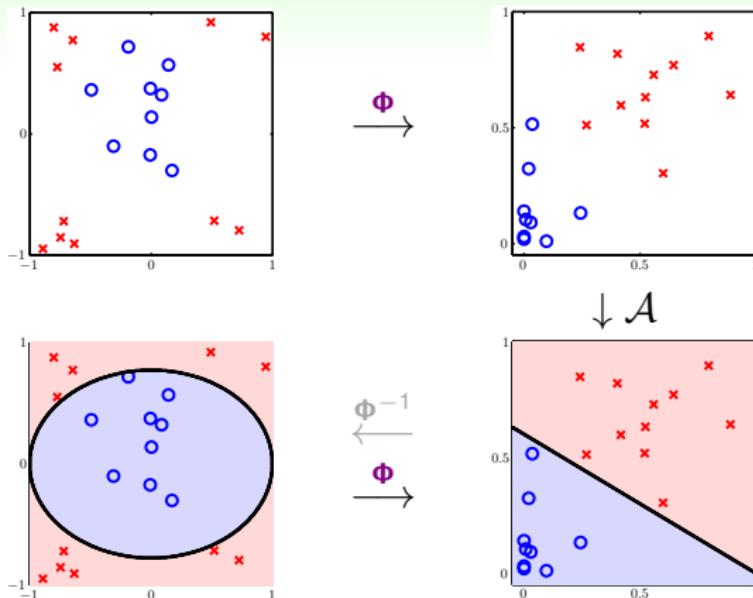
todo: get **good perceptron** in \mathcal{Z} -space with data $\{(\mathbf{z}_n = \Phi_2(\mathbf{x}_n), y_n)\}$

The Nonlinear Transform Steps



- 1 transform original data $\{(\mathbf{x}_n, y_n)\}$ to $\{(\mathbf{z}_n = \Phi(\mathbf{x}_n), y_n)\}$ by Φ
- 2 get a good perceptron $\tilde{\mathbf{w}}$ using $\{(\mathbf{z}_n, y_n)\}$ and your favorite linear classification algorithm \mathcal{A}
- 3 return $g(\mathbf{x}) = \text{sign}(\tilde{\mathbf{w}}^T \Phi(\mathbf{x}))$

Nonlinear Model via Nonlinear Φ + Linear Models



two choices:

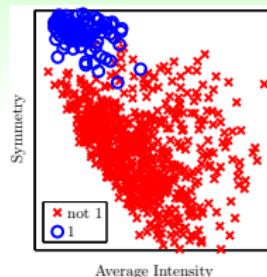
- feature transform Φ
- linear model \mathcal{A} , **not just binary classification**

Pandora's box :-):

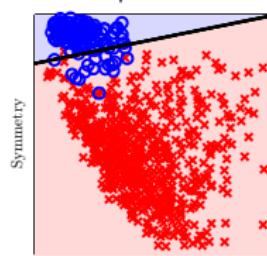
can now freely do **quadratic PLA, quadratic regression, cubic regression, . . . , polynomial regression**

Feature Transform Φ 

$$\Phi \rightarrow$$



$$\Phi^{-1} \uparrow \Phi \rightarrow$$



not new, not just polynomial:

raw (pixels) $\xrightarrow{\text{domain knowledge}}$ concrete (intensity, symmetry)

the force, too good to be true? :-)

Fun Time

Consider the quadratic transform $\Phi_2(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^d$ instead of in \mathbb{R}^2 . The transform should include all different quadratic, linear, and constant terms formed by (x_1, x_2, \dots, x_d) . What is the number of dimensions of $\mathbf{z} = \Phi_2(\mathbf{x})$?

- 1 d
- 2 $\frac{d^2}{2} + \frac{3d}{2} + 1$
- 3 $d^2 + d + 1$
- 4 2^d

Fun Time

Consider the quadratic transform $\Phi_2(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^d$ instead of in \mathbb{R}^2 . The transform should include all different quadratic, linear, and constant terms formed by (x_1, x_2, \dots, x_d) . What is the number of dimensions of $\mathbf{z} = \Phi_2(\mathbf{x})$?

- 1 d
- 2 $\frac{d^2}{2} + \frac{3d}{2} + 1$
- 3 $d^2 + d + 1$
- 4 2^d

Reference Answer: (2)

Number of different quadratic terms is $\binom{d}{2} + d$;
number of different linear terms is d ;
number of different constant term is 1.

Computation/Storage Price

Q -th order polynomial transform: $\Phi_Q(\mathbf{x}) = \left(\begin{array}{l} 1, \\ x_1, x_2, \dots, x_d, \\ x_1^2, x_1 x_2, \dots, x_d^2, \\ \dots, \\ x_1^Q, x_1^{Q-1} x_2, \dots, x_d^Q \end{array} \right)$

$\underbrace{1}_{\tilde{w}_0} + \underbrace{\tilde{d}}_{\text{others}} \text{ dimensions}$

= # ways of $\leq Q$ -combination from d kinds with repetitions

$$= \binom{Q+d}{Q} = \binom{Q+d}{d} = O(Q^d)$$

= efforts needed for computing/storing $\mathbf{z} = \Phi_Q(\mathbf{x})$ and $\tilde{\mathbf{w}}$

Q large \implies difficult to compute/store

Model Complexity Price

Q -th order polynomial transform: $\Phi_Q(\mathbf{x}) = \begin{pmatrix} 1, \\ x_1, x_2, \dots, x_d, \\ x_1^2, x_1 x_2, \dots, x_d^2, \\ \dots, \\ x_1^Q, x_1^{Q-1} x_2, \dots, x_d^Q \end{pmatrix}$

$$\underbrace{1}_{\tilde{w}_0} + \underbrace{\tilde{d}}_{\text{others}} \text{ dimensions} = O(Q^d)$$

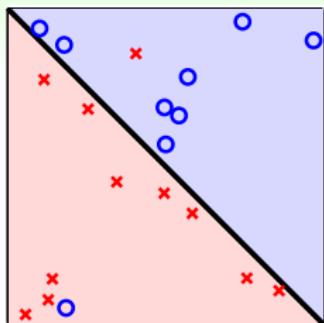
- number of free parameters $\tilde{w}_i = \tilde{d} + 1 \approx d_{VC}(\mathcal{H}_{\Phi_Q})$

- $d_{VC}(\mathcal{H}_{\Phi_Q}) \leq \tilde{d} + 1$, why?

any $\tilde{d} + 2$ inputs not shattered in \mathcal{Z}
 \implies any $\tilde{d} + 2$ inputs not shattered in \mathcal{X}

Q large \implies **large d_{VC}**

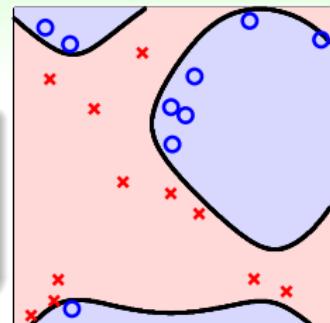
Generalization Issue



Φ_1 (original \mathbf{x})

which one do you prefer? :-)

- Φ_1 'visually' preferred
- Φ_4 : $E_{\text{in}}(g) = 0$ but overkill



Φ_4

- ① can we make sure that $E_{\text{out}}(g)$ is close enough to $E_{\text{in}}(g)$?
- ② can we make $E_{\text{in}}(g)$ small enough?

	$\tilde{d}(Q)$	1	2
trade-off:	higher	:-)	:-D
	lower	-D	:-)

how to pick Q ? **visually**, maybe?

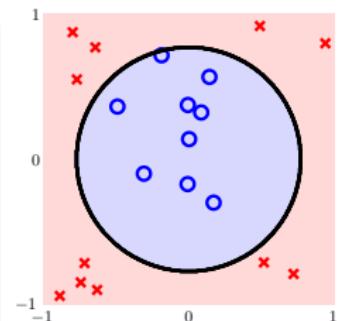
Danger of Visual Choices

first of all, can you really ‘visualize’ when $\mathcal{X} = \mathbb{R}^{10}$? (well, I can’t :-))

Visualize $\mathcal{X} = \mathbb{R}^2$

- full Φ_2 : $\mathbf{z} = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$, $d_{VC} = 6$
- or $\mathbf{z} = (1, x_1^2, x_2^2)$, $d_{VC} = 3$, after visualizing?
- or better $\mathbf{z} = (1, x_1^2 + x_2^2)$, $d_{VC} = 2$?
- or even better $\mathbf{z} = (\text{sign}(0.6 - x_1^2 - x_2^2))$?

—careful about your brain’s ‘model complexity’



for VC-safety, Φ shall be decided without ‘peeking’ data

Fun Time

Consider the Q -th order polynomial transform $\Phi_Q(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^2$. Recall that $\tilde{d} = \binom{Q+2}{2} - 1$. When $Q = 50$, what is the value of \tilde{d} ?

- 1 1126
- 2 1325
- 3 2651
- 4 6211

Fun Time

Consider the Q -th order polynomial transform $\Phi_Q(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^2$. Recall that $\tilde{d} = \binom{Q+2}{2} - 1$. When $Q = 50$, what is the value of \tilde{d} ?

- 1 1126
- 2 1325
- 3 2651
- 4 6211

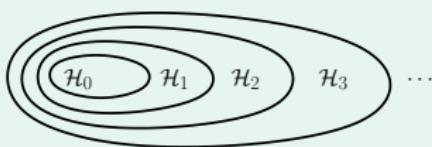
Reference Answer: ②

It's just a simple calculation, but shows you how \tilde{d} becomes hundreds of times of $d = 2$ after the transform.

Polynomial Transform Revisited

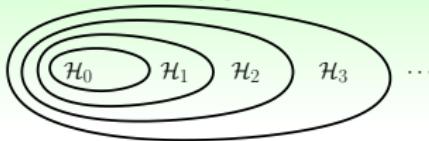
$$\begin{aligned}
 \Phi_0(\mathbf{x}) &= (1), \quad \Phi_1(\mathbf{x}) = (\Phi_0(\mathbf{x}), \quad x_1, x_2, \dots, x_d) \\
 \Phi_2(\mathbf{x}) &= (\Phi_1(\mathbf{x}), \quad x_1^2, x_1x_2, \dots, x_d^2) \\
 \Phi_3(\mathbf{x}) &= (\Phi_2(\mathbf{x}), \quad x_1^3, x_1^2x_2, \dots, x_d^3) \\
 &\quad \dots \quad \dots \\
 \Phi_Q(\mathbf{x}) &= (\Phi_{Q-1}(\mathbf{x}), \quad x_1^Q, x_1^{Q-1}x_2, \dots, x_d^Q)
 \end{aligned}$$

$$\begin{array}{ccccccccc}
 \mathcal{H}_{\Phi_0} & \subset & \mathcal{H}_{\Phi_1} & \subset & \mathcal{H}_{\Phi_2} & \subset & \mathcal{H}_{\Phi_3} & \subset & \dots \subset \mathcal{H}_{\Phi_Q} \\
 \parallel & & \parallel & & \parallel & & \parallel & & \parallel \\
 \mathcal{H}_0 & & \mathcal{H}_1 & & \mathcal{H}_2 & & \mathcal{H}_3 & & \dots \mathcal{H}_Q
 \end{array}$$



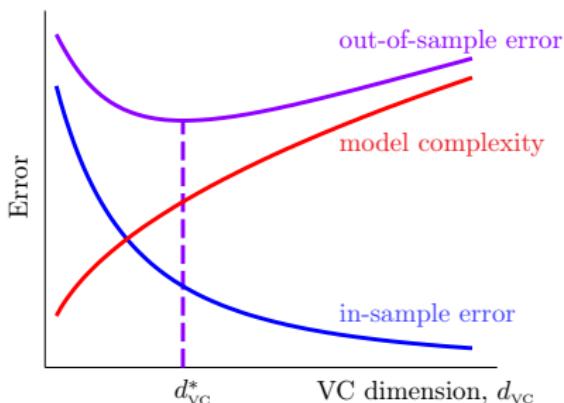
structure: **nested \mathcal{H}_i**

Structured Hypothesis Sets



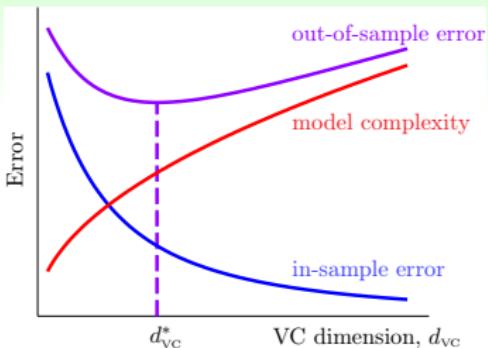
Let $g_i = \operatorname{argmin}_{h \in \mathcal{H}_i} E_{\text{in}}(h)$:

$$\begin{array}{ccccccc} \mathcal{H}_0 & \subset & \mathcal{H}_1 & \subset & \mathcal{H}_2 & \subset & \mathcal{H}_3 & \subset & \dots \\ d_{\text{VC}}(\mathcal{H}_0) & \leq & d_{\text{VC}}(\mathcal{H}_1) & \leq & d_{\text{VC}}(\mathcal{H}_2) & \leq & d_{\text{VC}}(\mathcal{H}_3) & \leq & \dots \\ E_{\text{in}}(g_0) & \geq & E_{\text{in}}(g_1) & \geq & E_{\text{in}}(g_2) & \geq & E_{\text{in}}(g_3) & \geq & \dots \end{array}$$



use \mathcal{H}_{1126} won't be good! :-(

Linear Model First



- tempting sin: use \mathcal{H}_{1126} , low $E_{in}(g_{1126})$ to fool your boss
—**really? :-(a dangerous path of no return**
- safe route: \mathcal{H}_1 first
 - if $E_{in}(g_1)$ good enough, **live happily thereafter :-)**
 - otherwise, move right of the curve
with nothing lost except ‘wasted’ computation

linear model first:
simple, efficient, **safe, and workable!**

Fun Time

Consider two hypothesis sets, \mathcal{H}_1 and \mathcal{H}_{1126} , where $\mathcal{H}_1 \subset \mathcal{H}_{1126}$. Which of the following relationship between $d_{\text{VC}}(\mathcal{H}_1)$ and $d_{\text{VC}}(\mathcal{H}_{1126})$ is not possible?

- ① $d_{\text{VC}}(\mathcal{H}_1) = d_{\text{VC}}(\mathcal{H}_{1126})$
- ② $d_{\text{VC}}(\mathcal{H}_1) \neq d_{\text{VC}}(\mathcal{H}_{1126})$
- ③ $d_{\text{VC}}(\mathcal{H}_1) < d_{\text{VC}}(\mathcal{H}_{1126})$
- ④ $d_{\text{VC}}(\mathcal{H}_1) > d_{\text{VC}}(\mathcal{H}_{1126})$

Fun Time

Consider two hypothesis sets, \mathcal{H}_1 and \mathcal{H}_{1126} , where $\mathcal{H}_1 \subset \mathcal{H}_{1126}$. Which of the following relationship between $d_{VC}(\mathcal{H}_1)$ and $d_{VC}(\mathcal{H}_{1126})$ is not possible?

- ① $d_{VC}(\mathcal{H}_1) = d_{VC}(\mathcal{H}_{1126})$
- ② $d_{VC}(\mathcal{H}_1) \neq d_{VC}(\mathcal{H}_{1126})$
- ③ $d_{VC}(\mathcal{H}_1) < d_{VC}(\mathcal{H}_{1126})$
- ④ $d_{VC}(\mathcal{H}_1) > d_{VC}(\mathcal{H}_{1126})$

Reference Answer: ④

Every input combination that \mathcal{H}_1 shatters can be shattered by \mathcal{H}_{1126} , so d_{VC} cannot decrease.

Summary

- ① When Can Machines Learn?
- ② Why Can Machines Learn?
- ③ **How** Can Machines Learn?

Lecture 11: Linear Models for Classification

Lecture 12: Nonlinear Transformation

- Quadratic Hypotheses

linear hypotheses on quadratic-transformed data

- Nonlinear Transform

happy linear modeling after $\mathcal{Z} = \Phi(\mathcal{X})$

- Price of Nonlinear Transform

computation/storage/[model complexity]

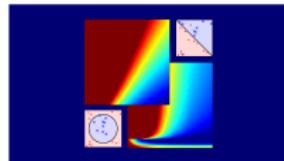
- Structured Hypothesis Sets

linear/simpler model first

- next: dark side of the force :-)

- ④ How Can Machines Learn Better?

Machine Learning Foundations (機器學習基石)



Lecture 13: Hazard of Overfitting

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?
- ② Why Can Machines Learn?
- ③ How Can Machines Learn?

Lecture 12: Nonlinear Transform

nonlinear \square via nonlinear feature transform Φ
plus linear \square with price of model complexity

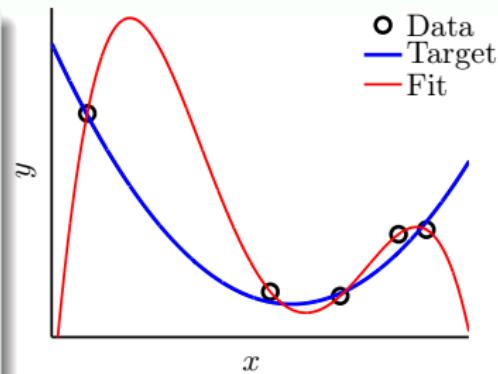
- ④ How Can Machines Learn **Better?**

Lecture 13: Hazard of Overfitting

- What is Overfitting?
- The Role of Noise and Data Size
- Deterministic Noise
- Dealing with Overfitting

Bad Generalization

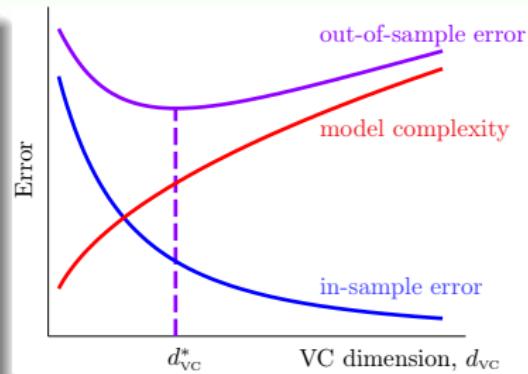
- regression for $x \in \mathbb{R}$ with $N = 5$ examples
- target $f(x) = 2\text{nd order polynomial}$
- label $y_n = f(x_n) + \text{very small noise}$
- linear regression in \mathcal{Z} -space +
 $\Phi = 4\text{th order polynomial}$
- unique solution passing all examples
 $\Rightarrow E_{\text{in}}(g) = 0$
- $E_{\text{out}}(g)$ huge



bad generalization: low E_{in} , high E_{out}

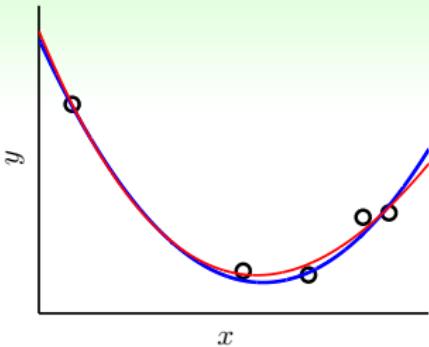
Bad Generalization and Overfitting

- take $d_{VC} = 1126$ for learning:
bad generalization
—(E_{out} - E_{in}) large
- switch from $d_{VC} = d_{VC}^*$ to $d_{VC} = 1126$:
overfitting
— $E_{in} \downarrow$, $E_{out} \uparrow$
- switch from $d_{VC} = d_{VC}^*$ to $d_{VC} = 1$:
underfitting
— $E_{in} \uparrow$, $E_{out} \uparrow$

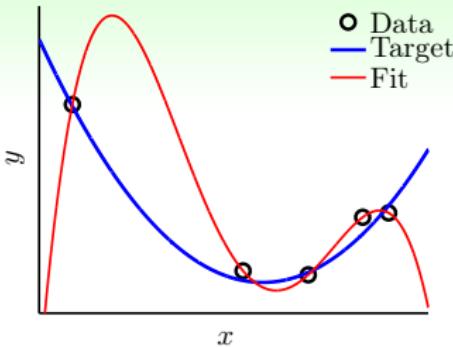


bad generalization: low E_{in} , high E_{out} ;
overfitting: lower E_{in} , higher E_{out}

Cause of Overfitting: A Driving Analogy



'good fit'



overfit

learning

driving

overfit

commit a car accident

use excessive d_{VC}

'drive too fast'

noise

bumpy road

limited data size N

limited observations about road condition

next: how does **noise** & **data size** affect
overfitting?

Fun Time

Based on our discussion, for data of fixed size, which of the following situation is relatively of the lowest risk of overfitting?

- ① small noise, fitting from small d_{VC} to median d_{VC}
- ② small noise, fitting from small d_{VC} to large d_{VC}
- ③ large noise, fitting from small d_{VC} to median d_{VC}
- ④ large noise, fitting from small d_{VC} to large d_{VC}

Fun Time

Based on our discussion, for data of fixed size, which of the following situation is relatively of the lowest risk of overfitting?

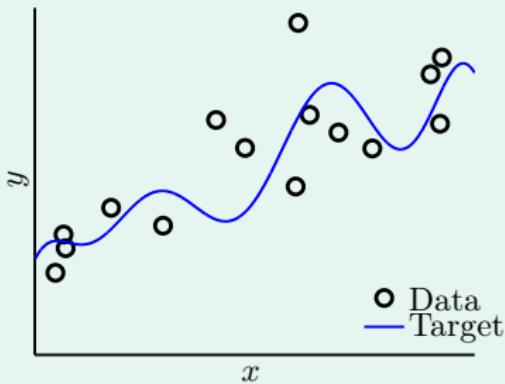
- ① small noise, fitting from small d_{VC} to median d_{VC}
- ② small noise, fitting from small d_{VC} to large d_{VC}
- ③ large noise, fitting from small d_{VC} to median d_{VC}
- ④ large noise, fitting from small d_{VC} to large d_{VC}

Reference Answer: ①

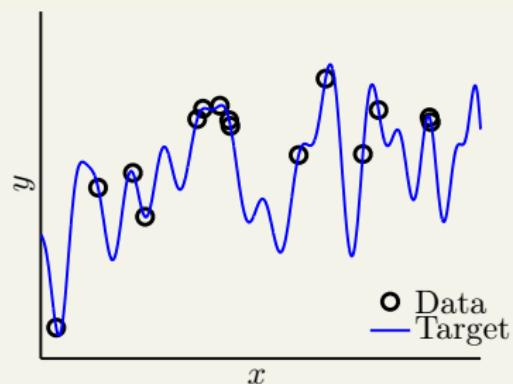
Two causes of overfitting are noise and excessive d_{VC} . So if both are relatively ‘under control’, the risk of overfitting is smaller.

Case Study (1/2)

10-th order target function
+ noise



50-th order target function
noiselessly



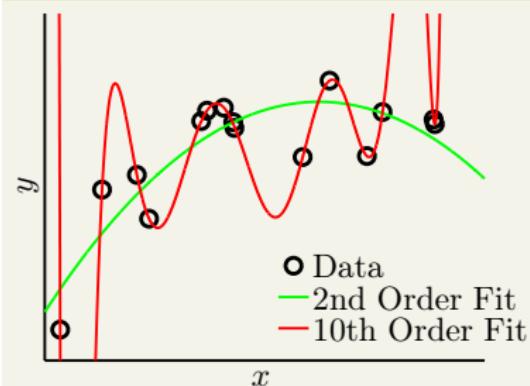
overfitting from best $g_2 \in \mathcal{H}_2$ to best $g_{10} \in \mathcal{H}_{10}$?

Case Study (2/2)

10-th order target function
+ noise



50-th order target function
noiselessly



$$g_2 \in \mathcal{H}_2 \quad g_{10} \in \mathcal{H}_{10}$$

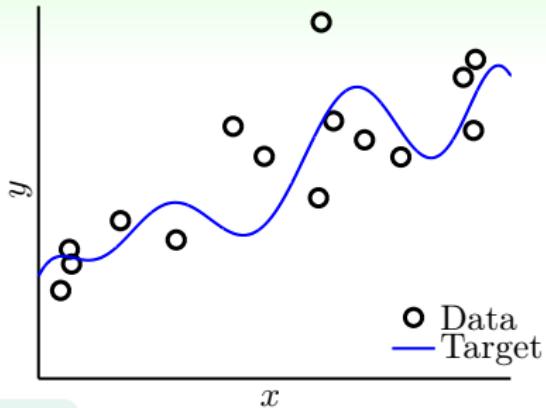
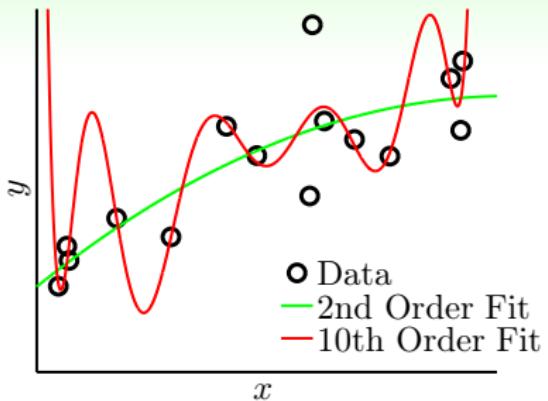
	E_{in}	E_{out}
g_2	0.050	0.034
g_{10}	0.127	9.00

$$g_2 \in \mathcal{H}_2 \quad g_{10} \in \mathcal{H}_{10}$$

	E_{in}	E_{out}
g_2	0.029	0.00001
g_{10}	0.120	7680

overfitting from g_2 to g_{10} ? **both yes!**

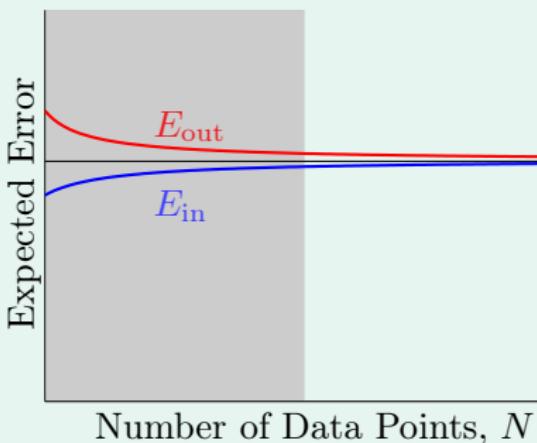
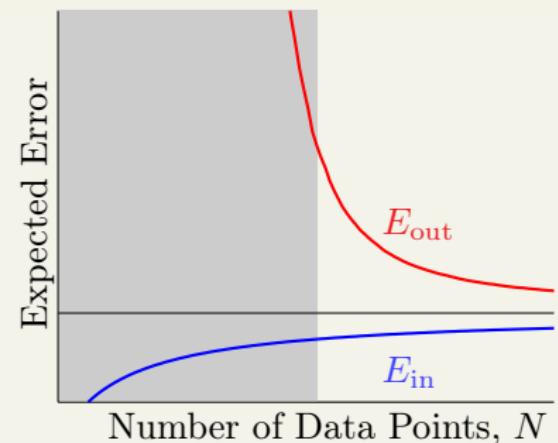
Irony of Two Learners



- learner *Overfit*: pick $g_{10} \in \mathcal{H}_{10}$
- learner *Restrict*: pick $g_2 \in \mathcal{H}_2$
- when both **know that target = 10th**
—R 'gives up' ability to fit

but R wins in E_{out} a lot!
philosophy: concession for advantage? :-)

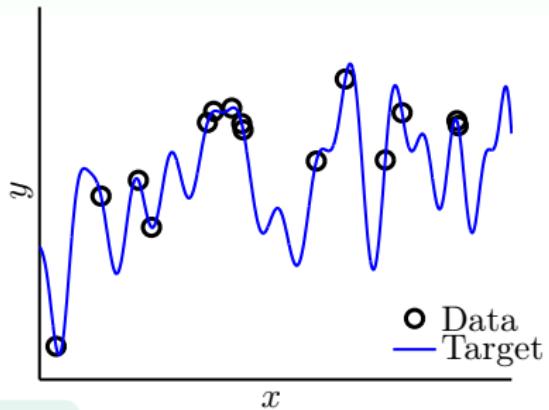
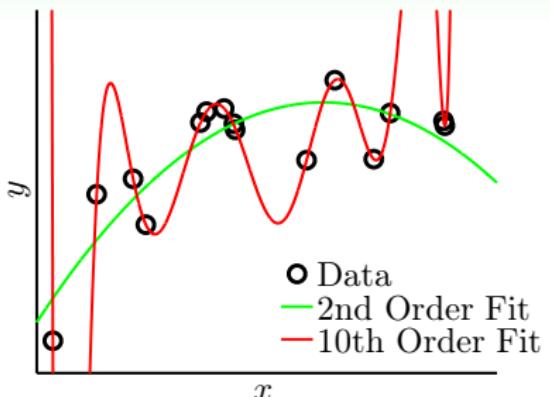
Learning Curves Revisited

 \mathcal{H}_2  \mathcal{H}_{10} 

- \mathcal{H}_{10} : lower $\overline{E_{out}}$ when $N \rightarrow \infty$,
but much larger generalization error for small N
- gray area : $\textcolor{red}{O}$ overfits! ($\overline{E_{in}} \downarrow$, $\overline{E_{out}} \uparrow$)

R always wins in $\overline{E_{out}}$ if N small!

The ‘No Noise’ Case



- learner *Overfit*: pick $g_{10} \in \mathcal{H}_{10}$
- learner *Restrict*: pick $g_2 \in \mathcal{H}_2$
- when both **know that there is no noise** — *R* still wins

is there really **no noise**?
 ‘target complexity’ acts like noise

Fun Time

When having limited data, in which of the following case would learner *R* perform better than learner *O*?

- ① limited data from a 10-th order target function with some noise
- ② limited data from a 1126-th order target function with no noise
- ③ limited data from a 1126-th order target function with some noise
- ④ all of the above

Fun Time

When having limited data, in which of the following case would learner R perform better than learner O ?

- ① limited data from a 10-th order target function with some noise
- ② limited data from a 1126-th order target function with no noise
- ③ limited data from a 1126-th order target function with some noise
- ④ all of the above

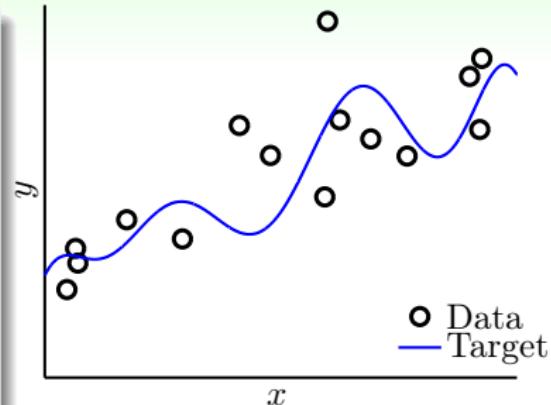
Reference Answer: ④

We discussed about ① and ②, but you shall be able to '**generalize**' :-)**)** that R also wins in the more difficult case of ③.

A Detailed Experiment

$$y = f(x) + \epsilon$$

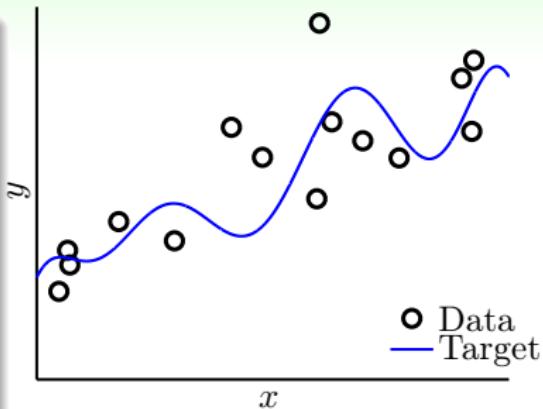
$$\sim Gaussian\left(\underbrace{\sum_{q=0}^{Q_f} \alpha_q x^q}_{f(x)}, \sigma^2\right)$$



- Gaussian iid noise ϵ with level σ^2
- some ‘uniform’ distribution on $f(x)$ with complexity level Q_f
- data size N

goal: ‘overfit level’ for different (N, σ^2) and (N, Q_f) ?

The Overfit Measure

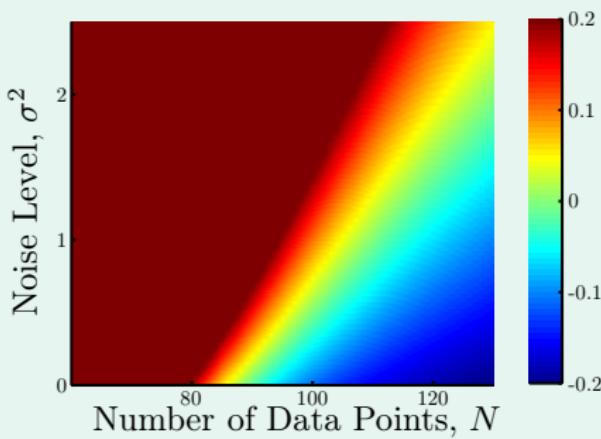


- $g_2 \in \mathcal{H}_2$
- $g_{10} \in \mathcal{H}_{10}$
- $E_{\text{in}}(g_{10}) \leq E_{\text{in}}(g_2)$ for sure

overfit measure $E_{\text{out}}(g_{10}) - E_{\text{out}}(g_2)$

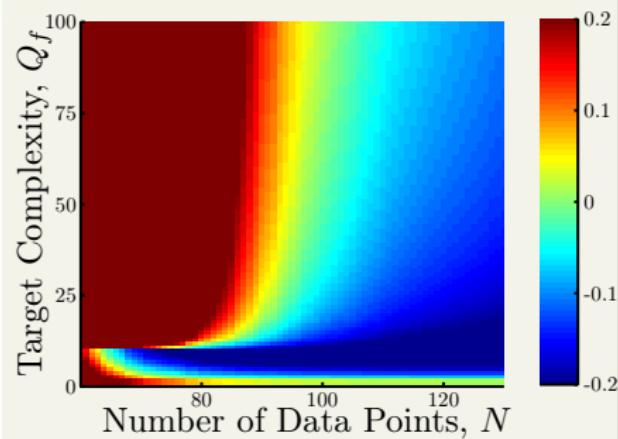
The Results

impact of σ^2 versus N



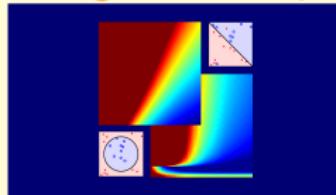
fixed $Q_f = 20$

impact of Q_f versus N



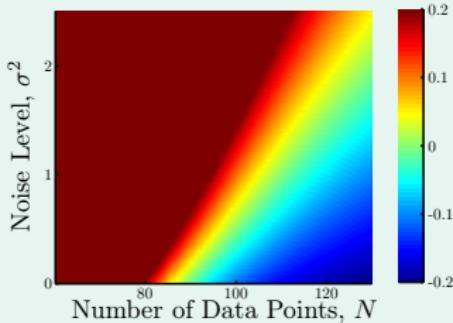
fixed $\sigma^2 = 0.1$

ring a bell? :-)

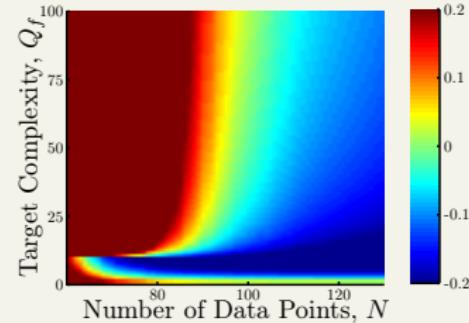


Impact of Noise and Data Size

impact of σ^2 versus N :
stochastic noise



impact of Q_f versus N :
deterministic noise



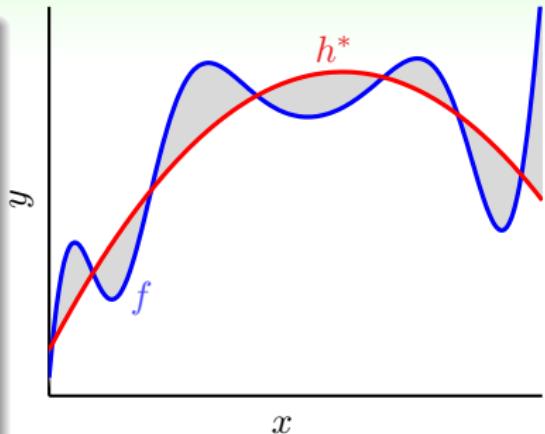
four reasons of serious overfitting:

data size $N \downarrow$	overfit ↑
stochastic noise ↑	overfit ↑
deterministic noise ↑	overfit ↑
excessive power ↑	overfit ↑

overfitting ‘easily’ happens

Deterministic Noise

- if $f \notin \mathcal{H}$: something of f cannot be captured by \mathcal{H}
- deterministic noise : difference between best $h^* \in \mathcal{H}$ and f
- acts like ‘stochastic noise’—not new to CS: pseudo-random generator
- difference to stochastic noise:
 - depends on \mathcal{H}
 - fixed for a given x



philosophy: when teaching a kid,
perhaps better not to use examples
from a complicated target function? :-)

Fun Time

Consider the target function being $\sin(1126x)$ for $x \in [0, 2\pi]$. When x is uniformly sampled from the range, and we use all possible linear hypotheses $h(x) = w \cdot x$ to approximate the target function with respect to the squared error, what is the level of deterministic noise for each x ?

- ① $|\sin(1126x)|$
- ② $|\sin(1126x) - x|$
- ③ $|\sin(1126x) + x|$
- ④ $|\sin(1126x) - 1126x|$

Fun Time

Consider the target function being $\sin(1126x)$ for $x \in [0, 2\pi]$. When x is uniformly sampled from the range, and we use all possible linear hypotheses $h(x) = w \cdot x$ to approximate the target function with respect to the squared error, what is the level of deterministic noise for each x ?

- ① $|\sin(1126x)|$
- ② $|\sin(1126x) - x|$
- ③ $|\sin(1126x) + x|$
- ④ $|\sin(1126x) - 1126x|$

Reference Answer: ①

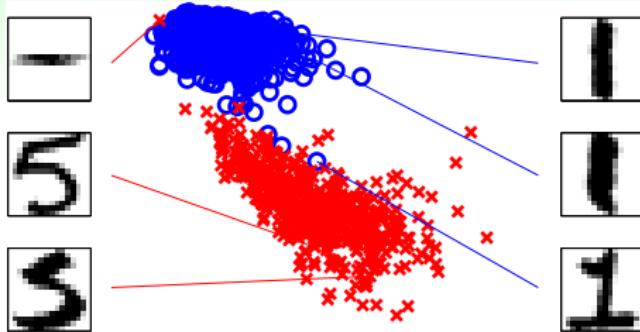
You can try a few different w and convince yourself that the best hypothesis h^* is $h^*(x) = 0$. The deterministic noise is the difference between f and h^* .

Driving Analogy Revisited

learning	driving
overfit	commit a car accident
use excessive d_{vc}	'drive too fast'
noise	bumpy road
limited data size N	limited observations about road condition
start from simple model	drive slowly
data cleaning/pruning	use more accurate road information
data hinting	exploit more road information
regularization	put the brakes
validation	monitor the dashboard

all very **practical** techniques
to combat overfitting

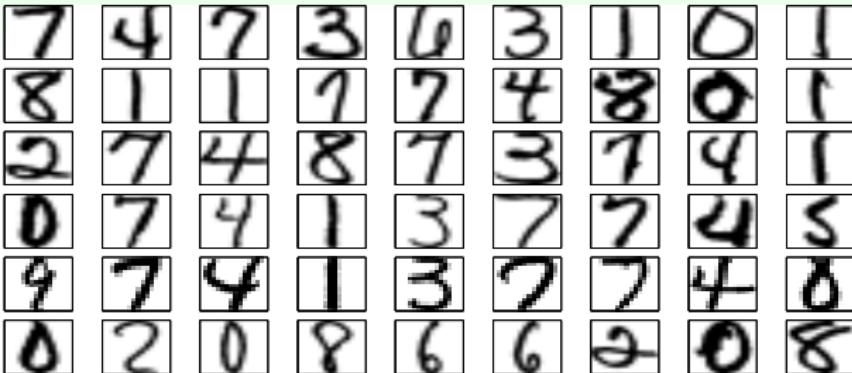
Data Cleaning/Pruning



- if 'detect' the outlier 5 at the top by
 - too close to other o, or too far from other x
 - wrong by current classifier
 - ...
- possible action 1: correct the label (**data cleaning**)
- possible action 2: remove the example (**data pruning**)

possibly helps, but **effect varies**

Data Hinting



- slightly shifted/rotated digits carry the same meaning
- possible action: add **virtual examples** by shifting/rotating the given digits (**data hinting**)

possibly helps, but **watch out**
—**virtual example not** $\stackrel{iid}{\sim} P(\mathbf{x}, y)!$

Fun Time

Assume we know that $f(x)$ is symmetric for some 1D regression application. That is, $f(x) = f(-x)$. One possibility of using the knowledge is to consider symmetric hypotheses only. On the other hand, you can also generate virtual examples from the original data $\{(x_n, y_n)\}$ as hints. What virtual examples suit your needs best?

- ① $\{(x_n, -y_n)\}$
- ② $\{(-x_n, -y_n)\}$
- ③ $\{(-x_n, y_n)\}$
- ④ $\{(2x_n, 2y_n)\}$

Fun Time

Assume we know that $f(x)$ is symmetric for some 1D regression application. That is, $f(x) = f(-x)$. One possibility of using the knowledge is to consider symmetric hypotheses only. On the other hand, you can also generate virtual examples from the original data $\{(x_n, y_n)\}$ as hints. What virtual examples suit your needs best?

- ① $\{(x_n, -y_n)\}$
- ② $\{(-x_n, -y_n)\}$
- ③ $\{(-x_n, y_n)\}$
- ④ $\{(2x_n, 2y_n)\}$

Reference Answer: ③

We want the virtual examples to encode the invariance when $x \rightarrow -x$.

Summary

- ① When Can Machines Learn?
- ② Why Can Machines Learn?
- ③ How Can Machines Learn?

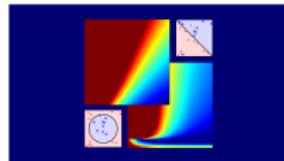
Lecture 12: Nonlinear Transform

- ④ How Can Machines Learn **Better?**

Lecture 13: Hazard of Overfitting

- What is Overfitting?
lower E_{in} but higher E_{out}
 - The Role of Noise and Data Size
overfitting ‘easily’ happens!
 - Deterministic Noise
what \mathcal{H} cannot capture acts like noise
 - Dealing with Overfitting
data cleaning/pruning/hinting, and more
-
- next: putting the brakes with regularization

Machine Learning Foundations (機器學習基石)



Lecture 14: Regularization

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?
- ② Why Can Machines Learn?
- ③ How Can Machines Learn?
- ④ How Can Machines Learn **Better?**

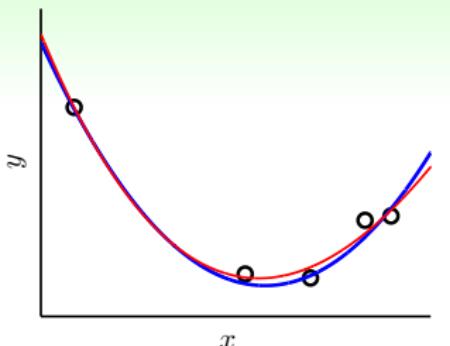
Lecture 13: Hazard of Overfitting

overfitting happens with **excessive power**,
stochastic/deterministic noise, and **limited data**

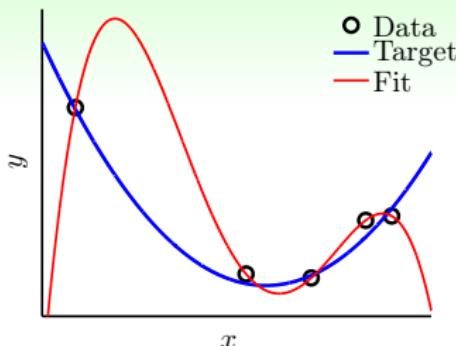
Lecture 14: Regularization

- Regularized Hypothesis Set
- Weight Decay Regularization
- Regularization and VC Theory
- General Regularizers

Regularization: The Magic

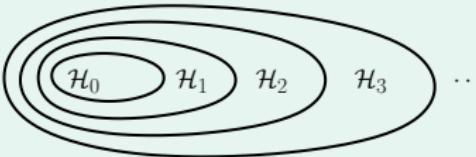


'regularized fit'



overfit

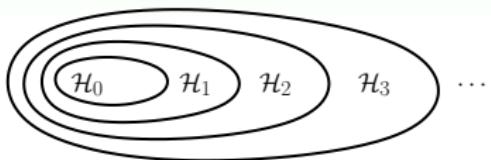
- idea: 'step back' from \mathcal{H}_{10} to \mathcal{H}_2



- name history: function approximation for **ill-posed problems**

how to step back?

Stepping Back as Constraint



Q -th order polynomial transform for $x \in \mathbb{R}$:

$$\Phi_Q(x) = (1, x, x^2, \dots, x^Q)$$

+ linear regression, denote $\tilde{\mathbf{w}}$ by \mathbf{w}

hypothesis \mathbf{w} in \mathcal{H}_{10} : $w_0 + w_1x + w_2x^2 + w_3x^3 + \dots + w_{10}x^{10}$

hypothesis \mathbf{w} in \mathcal{H}_2 : $w_0 + w_1x + w_2x^2$

that is, $\mathcal{H}_2 = \mathcal{H}_{10}$ AND ‘constraint that $w_3 = w_4 = \dots = w_{10} = 0$ ’

step back = **constraint**

Regression with Constraint

$$\mathcal{H}_{10} \equiv \left\{ \mathbf{w} \in \mathbb{R}^{10+1} \right\}$$

$$\mathcal{H}_2 \equiv \left\{ \mathbf{w} \in \mathbb{R}^{10+1} \text{ while } w_3 = w_4 = \dots = w_{10} = 0 \right\}$$

regression with \mathcal{H}_{10} :

$$\min_{\mathbf{w} \in \mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w})$$

regression with \mathcal{H}_2 :

$$\begin{aligned} & \min_{\mathbf{w} \in \mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w}) \\ \text{s.t. } & w_3 = w_4 = \dots = w_{10} = 0 \end{aligned}$$

step back = constrained optimization of E_{in}

why don't you just use $\mathbf{w} \in \mathbb{R}^{2+1}$? :-)

Regression with Looser Constraint

$$\mathcal{H}_2 \equiv \left\{ \mathbf{w} \in \mathbb{R}^{10+1} \text{ while } w_3 = \dots = w_{10} = 0 \right\}$$

regression with \mathcal{H}_2 :

$$\min_{\mathbf{w} \in \mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w})$$

$$\text{s.t. } w_3 = \dots = w_{10} = 0$$

$$\mathcal{H}'_2 \equiv \left\{ \mathbf{w} \in \mathbb{R}^{10+1} \text{ while } \geq 8 \text{ of } w_q = 0 \right\}$$

regression with \mathcal{H}'_2 :

$$\min_{\mathbf{w} \in \mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w})$$

$$\text{s.t. } \sum_{q=0}^{10} [\![w_q \neq 0]\!] \leq 3$$

- more flexible than \mathcal{H}_2 : $\mathcal{H}_2 \subset \mathcal{H}'_2$
- less risky than \mathcal{H}_{10} : $\mathcal{H}'_2 \subset \mathcal{H}_{10}$

bad news for sparse hypothesis set \mathcal{H}'_2 :
NP-hard to solve :-)

Regression with Softer Constraint

$$\mathcal{H}'_2 \equiv \left\{ \mathbf{w} \in \mathbb{R}^{10+1} \text{ while } \geq 8 \text{ of } w_q = 0 \right\}$$

regression with \mathcal{H}'_2 :

$$\min_{\mathbf{w} \in \mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w}) \text{ s.t. } \sum_{q=0}^{10} [\![w_q \neq 0]\!] \leq 3$$

$$\mathcal{H}(C) \equiv \left\{ \mathbf{w} \in \mathbb{R}^{10+1} \text{ while } \|\mathbf{w}\|^2 \leq C \right\}$$

regression with $\mathcal{H}(C)$:

$$\min_{\mathbf{w} \in \mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w}) \text{ s.t. } \sum_{q=0}^{10} w_q^2 \leq C$$

- $\mathcal{H}(C)$: overlaps but not exactly the same as \mathcal{H}'_2
- soft and smooth structure over $C \geq 0$:
 $\mathcal{H}(0) \subset \mathcal{H}(1.126) \subset \dots \subset \mathcal{H}(1126) \subset \dots \subset \mathcal{H}(\infty) = \mathcal{H}_{10}$

regularized hypothesis \mathbf{w}_{REG} :
optimal solution from
regularized hypothesis set $\mathcal{H}(C)$

Fun Time

For $Q \geq 1$, which of the following hypothesis (weight vector $\mathbf{w} \in \mathbb{R}^{Q+1}$) is not in the regularized hypothesis set $\mathcal{H}(1)$?

- ① $\mathbf{w}^T = [0, 0, \dots, 0]$
- ② $\mathbf{w}^T = [1, 0, \dots, 0]$
- ③ $\mathbf{w}^T = [1, 1, \dots, 1]$
- ④ $\mathbf{w}^T = \left[\sqrt{\frac{1}{Q+1}}, \sqrt{\frac{1}{Q+1}}, \dots, \sqrt{\frac{1}{Q+1}} \right]$

Fun Time

For $Q \geq 1$, which of the following hypothesis (weight vector $\mathbf{w} \in \mathbb{R}^{Q+1}$) is not in the regularized hypothesis set $\mathcal{H}(1)$?

- ① $\mathbf{w}^T = [0, 0, \dots, 0]$
- ② $\mathbf{w}^T = [1, 0, \dots, 0]$
- ③ $\mathbf{w}^T = [1, 1, \dots, 1]$
- ④ $\mathbf{w}^T = \left[\sqrt{\frac{1}{Q+1}}, \sqrt{\frac{1}{Q+1}}, \dots, \sqrt{\frac{1}{Q+1}} \right]$

Reference Answer: ③

The squared length of \mathbf{w} in ③ is $Q + 1$, which is not ≤ 1 .

Matrix Form of Regularized Regression Problem

$$\min_{\mathbf{w} \in \mathbb{R}^{Q+1}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \underbrace{\sum_{n=1}^N (\mathbf{w}^T \mathbf{z}_n - y_n)^2}_{(\mathbf{Z}\mathbf{w} - \mathbf{y})^T (\mathbf{Z}\mathbf{w} - \mathbf{y})}$$

s.t.

$$\sum_{q=0}^Q w_q^2 \leq C$$

$\underbrace{w^T w}_{\mathbf{w}^T \mathbf{w}}$

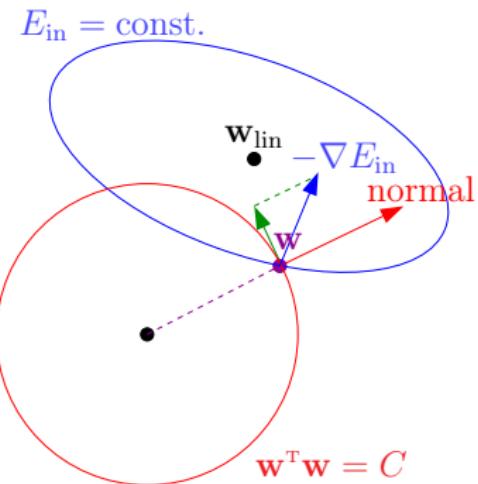
- $\sum_n \dots = (\mathbf{Z}\mathbf{w} - \mathbf{y})^T (\mathbf{Z}\mathbf{w} - \mathbf{y})$, remember? :-)
- $\mathbf{w}^T \mathbf{w} \leq C$: feasible \mathbf{w} within a radius- \sqrt{C} hypersphere

how to solve
constrained optimization problem?

The Lagrange Multiplier

$$\min_{\mathbf{w} \in \mathbb{R}^{Q+1}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N} (\mathbf{Z}\mathbf{w} - \mathbf{y})^T (\mathbf{Z}\mathbf{w} - \mathbf{y}) \text{ s.t. } \mathbf{w}^T \mathbf{w} \leq C$$

- decreasing direction: $-\nabla E_{\text{in}}(\mathbf{w})$, **remember? :-)**
- normal vector of $\mathbf{w}^T \mathbf{w} = C$: \mathbf{w}
- if $-\nabla E_{\text{in}}(\mathbf{w})$ and \mathbf{w} not parallel: can **decrease $E_{\text{in}}(\mathbf{w})$ without violating the constraint**
- at optimal solution \mathbf{w}_{REG} ,
 $-\nabla E_{\text{in}}(\mathbf{w}_{\text{REG}}) \propto \boxed{\mathbf{w}_{\text{REG}}}$



want: find **Lagrange multiplier $\lambda > 0$** and \mathbf{w}_{REG}
such that $\nabla E_{\text{in}}(\mathbf{w}_{\text{REG}}) + \frac{2\lambda}{N} \boxed{\mathbf{w}_{\text{REG}}} = \mathbf{0}$

Augmented Error

- if oracle tells you $\lambda > 0$, then

solving $\nabla E_{\text{in}}(\mathbf{w}_{\text{REG}}) + \frac{2\lambda}{N} \boxed{\mathbf{w}_{\text{REG}}} = \mathbf{0}$

$$\frac{2}{N} (Z^T Z \mathbf{w}_{\text{REG}} - Z^T \mathbf{y}) + \frac{2\lambda}{N} \boxed{\mathbf{w}_{\text{REG}}} = \mathbf{0}$$

- optimal solution:

$$\mathbf{w}_{\text{REG}} \leftarrow (Z^T Z + \lambda I)^{-1} Z^T \mathbf{y}$$

—called **ridge regression** in Statistics

minimizing **unconstrained E_{aug}** effectively
minimizes some **C -constrained E_{in}**

Augmented Error

- if oracle tells you $\lambda > 0$, then

solving $\nabla E_{\text{in}}(\mathbf{w}_{\text{REG}}) + \frac{2\lambda}{N} \boxed{\mathbf{w}_{\text{REG}}} = \mathbf{0}$

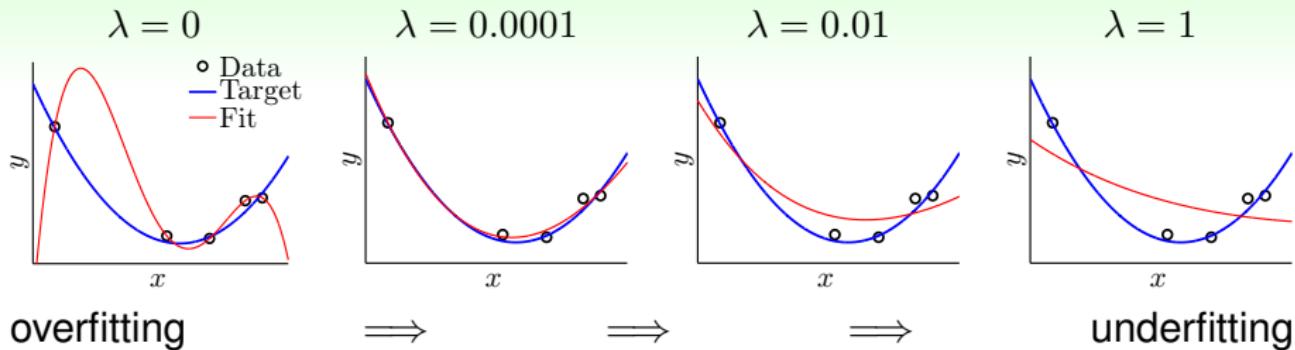
equivalent to minimizing $E_{\text{in}}(\mathbf{w}) + \underbrace{\frac{\lambda}{N} \overbrace{\mathbf{w}^T \mathbf{w}}^{\text{regularizer}}}_{\text{augmented error } E_{\text{aug}}(\mathbf{w})}$

- regularization with augmented error instead of constrained E_{in}

$$\mathbf{w}_{\text{REG}} \leftarrow \operatorname{argmin}_{\mathbf{w}} E_{\text{aug}}(\mathbf{w}) \text{ for given } \lambda > 0 \text{ or } \lambda = 0$$

minimizing unconstrained E_{aug} effectively
minimizes some C -constrained E_{in}

The Results



philosophy: *a little regularization goes a long way!*

call ' $+ \frac{\lambda}{N} \mathbf{w}^T \mathbf{w}$ ' **weight-decay** regularization:

larger λ

\iff prefer shorter \mathbf{w}

\iff effectively smaller C

—go with ‘any’ transform + linear model

Some Detail: Legendre Polynomials

$$\min_{\mathbf{w} \in \mathbb{R}^{Q+1}} \frac{1}{N} \sum_{n=0}^N (\mathbf{w}^T \Phi(x_n) - y_n)^2 + \frac{\lambda}{N} \sum_{q=0}^Q w_q^2$$

naïve polynomial transform:

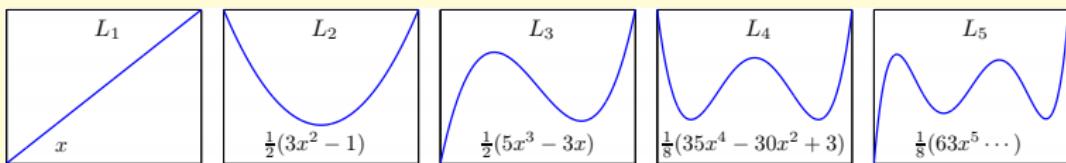
$$\Phi(\mathbf{x}) = (1, x, x^2, \dots, x^Q)$$

—when $x_n \in [-1, +1]$, x_n^q really small, needing large w_q

normalized polynomial transform:

$$(1, L_1(x), L_2(x), \dots, L_Q(x))$$

—‘orthonormal basis functions’ called **Legendre polynomials**



Fun Time

When would \mathbf{w}_{REG} equal \mathbf{w}_{LIN} ?

- 1 $\lambda = 0$
- 2 $C = \infty$
- 3 $C \geq \|\mathbf{w}_{\text{LIN}}\|^2$
- 4 all of the above

Fun Time

When would \mathbf{w}_{REG} equal \mathbf{w}_{LIN} ?

- 1 $\lambda = 0$
- 2 $C = \infty$
- 3 $C \geq \|\mathbf{w}_{\text{LIN}}\|^2$
- 4 all of the above

Reference Answer: ④

① and ② shall be easy; ③ means that there are effectively no constraint on \mathbf{w} , hence the equivalence.

Regularization and VC Theory

Regularization by
Constrained-Minimizing E_{in}

$$\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) \text{ s.t. } \mathbf{w}^T \mathbf{w} \leq C$$



VC Guarantee of
Constrained-Minimizing E_{in}

$$E_{\text{out}}(\mathbf{w}) \leq E_{\text{in}}(\mathbf{w}) + \Omega(\mathcal{H}(C))$$

\Updownarrow C equivalent to some λ

Regularization by
Minimizing E_{aug}

$$\min_{\mathbf{w}} E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N} \mathbf{w}^T \mathbf{w}$$

minimizing E_{aug} : indirectly getting VC
guarantee **without confining to $\mathcal{H}(C)$**

Another View of Augmented Error

Augmented Error

$$E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N} \mathbf{w}^T \mathbf{w}$$

VC Bound

$$E_{\text{out}}(\mathbf{w}) \leq E_{\text{in}}(\mathbf{w}) + \Omega(\mathcal{H})$$

- regularizer $\mathbf{w}^T \mathbf{w}$: complexity of a single hypothesis
- generalization price $\Omega(\mathcal{H})$: complexity of a hypothesis set
- if $\frac{\lambda}{N} \Omega(\mathbf{w})$ ‘represents’ $\Omega(\mathcal{H})$ well,
 E_{aug} is a better proxy of E_{out} than E_{in}

minimizing E_{aug} :

(heuristically) operating with the better proxy;
(technically) enjoying flexibility of whole \mathcal{H}

Effective VC Dimension

$$\min_{\mathbf{w} \in \mathbb{R}^{\tilde{d}+1}} E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N} \Omega(\mathbf{w})$$

- model complexity?
 $d_{\text{VC}}(\mathcal{H}) = \tilde{d} + 1$, because $\{\mathbf{w}\}$ ‘all considered’ during minimization
- $\{\mathbf{w}\}$ ‘actually needed’: $\mathcal{H}(\mathcal{C})$, with some \mathcal{C} equivalent to λ
- $d_{\text{VC}}(\mathcal{H}(\mathcal{C}))$:
 effective VC dimension $d_{\text{EFF}}(\mathcal{H}, \underbrace{\mathcal{A}}_{\min E_{\text{aug}}})$

explanation of regularization:

$d_{\text{VC}}(\mathcal{H})$ large,
 while $d_{\text{EFF}}(\mathcal{H}, \mathcal{A})$ small if \mathcal{A} regularized

Fun Time

Consider the weight-decay regularization with regression. When increasing λ in \mathcal{A} , what would happen with $d_{\text{EFF}}(\mathcal{H}, \mathcal{A})$?

- 1 $d_{\text{EFF}} \uparrow$
- 2 $d_{\text{EFF}} \downarrow$
- 3 $d_{\text{EFF}} = d_{\text{VC}}(\mathcal{H})$ and does not depend on λ
- 4 $d_{\text{EFF}} = 1126$ and does not depend on λ

Fun Time

Consider the weight-decay regularization with regression. When increasing λ in \mathcal{A} , what would happen with $d_{\text{EFF}}(\mathcal{H}, \mathcal{A})$?

- ① $d_{\text{EFF}} \uparrow$
- ② $d_{\text{EFF}} \downarrow$
- ③ $d_{\text{EFF}} = d_{\text{VC}}(\mathcal{H})$ and does not depend on λ
- ④ $d_{\text{EFF}} = 1126$ and does not depend on λ

Reference Answer: ②

larger λ
 \iff smaller C
 \iff smaller $\mathcal{H}(C)$
 \iff smaller d_{EFF}

General Regularizers $\Omega(\mathbf{w})$

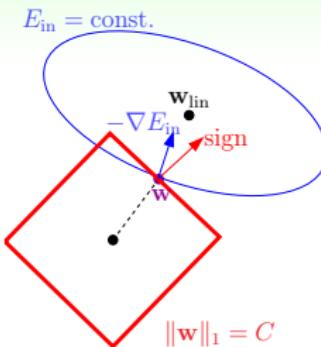
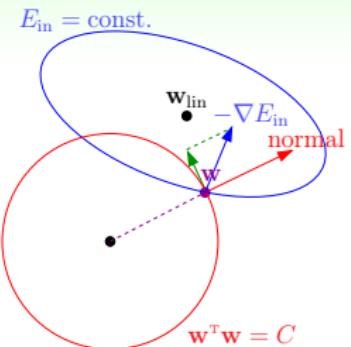
want: constraint in the '**direction**' of target function

- target-dependent: some **properties** of target, if known
 - **symmetry** regularizer: $\sum [q \text{ is odd}] w_q^2$
- plausible: direction towards **smoother** or **simpler**
stochastic/deterministic noise both **non-smooth**
 - **sparsity** (L1) regularizer: $\sum |w_q|$ (next slide)
- friendly: easy to **optimize**
 - **weight-decay** (L2) regularizer: $\sum w_q^2$
- **bad? :-)**: no worries, guard by λ

augmented error = error $\widehat{\text{err}}$ + regularizer Ω
regularizer: **target-dependent**, **plausible**, or **friendly**
ringing a bell? :-)

error measure: **user-dependent**, **plausible**, or **friendly**

L2 and L1 Regularizer



L2 Regularizer

$$\Omega(\mathbf{w}) = \sum_{q=0}^Q w_q^2 = \|\mathbf{w}\|_2^2$$

- convex, differentiable everywhere
- easy to optimize

L1 Regularizer

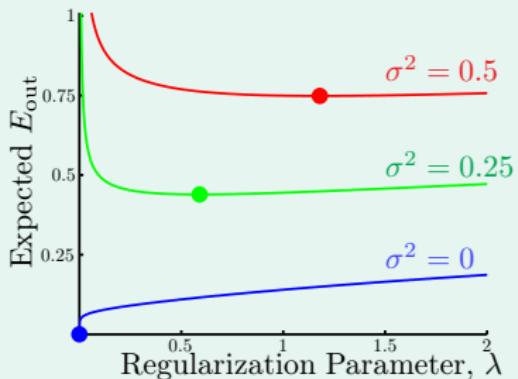
$$\Omega(\mathbf{w}) = \sum_{q=0}^Q |w_q| = \|\mathbf{w}\|_1$$

- convex, **not** differentiable everywhere
- **sparsity** in solution

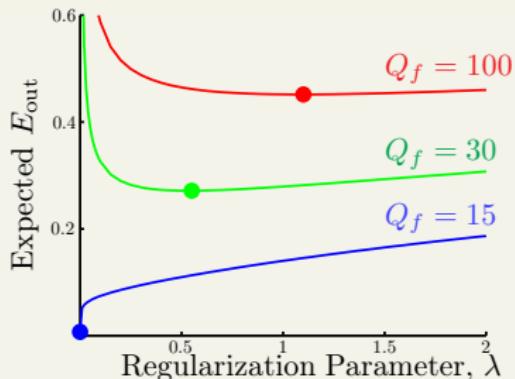
L1 useful if needing **sparse solution**

The Optimal λ

stochastic noise



deterministic noise



- more noise \iff more regularization needed
—more bumpy road \iff putting brakes more
- noise **unknown**—important to **make proper choices**

how to choose?
stay tuned for the next lecture! :-)

Fun Time

Consider using a regularizer $\Omega(\mathbf{w}) = \sum_{q=0}^Q 2^q w_q^2$ to work with Legendre polynomial regression. Which kind of hypothesis does the regularizer prefer?

- ① symmetric polynomials satisfying $h(x) = h(-x)$
- ② low-dimensional polynomials
- ③ high-dimensional polynomials
- ④ no specific preference

Fun Time

Consider using a regularizer $\Omega(\mathbf{w}) = \sum_{q=0}^Q 2^q w_q^2$ to work with Legendre polynomial regression. Which kind of hypothesis does the regularizer prefer?

- ① symmetric polynomials satisfying $h(x) = h(-x)$
- ② low-dimensional polynomials
- ③ high-dimensional polynomials
- ④ no specific preference

Reference Answer: ②

There is a higher ‘penalty’ for higher-order terms, and hence the regularizer prefers low-dimensional polynomials.

Summary

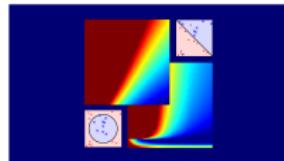
- 1 When Can Machines Learn?
- 2 Why Can Machines Learn?
- 3 How Can Machines Learn?
- 4 How Can Machines Learn **Better?**

Lecture 13: Hazard of Overfitting

Lecture 14: Regularization

- Regularized Hypothesis Set
original \mathcal{H} + constraint
- Weight Decay Regularization
add $\frac{\lambda}{N} \mathbf{w}^T \mathbf{w}$ in E_{aug}
- Regularization and VC Theory
regularization decreases d_{EFF}
- General Regularizers
target-dependent, [plausible], or [friendly]
- next: choosing from the so-many models/parameters

Machine Learning Foundations (機器學習基石)



Lecture 15: Validation
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?
- ② Why Can Machines Learn?
- ③ How Can Machines Learn?
- ④ How Can Machines Learn **Better?**

Lecture 14: Regularization

minimizes **augmented error**, where the added **regularizer** effectively **limits model complexity**

Lecture 15: Validation

- Model Selection Problem
- Validation
- Leave-One-Out Cross Validation
- V-Fold Cross Validation

So Many Models Learned

Even Just for Binary Classification . . .

$$\mathcal{A} \in \{ \text{PLA, pocket, linear regression, logistic regression} \}$$



$$T \in \{ 100, 1000, 10000 \}$$



$$\eta \in \{ 1, 0.01, 0.0001 \}$$



$$\Phi \in \{ \text{linear, quadratic, poly-10, Legendre-poly-10} \}$$



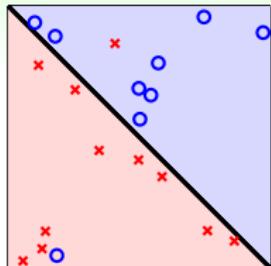
$$\Omega(\mathbf{w}) \in \{ \text{L2 regularizer, L1 regularizer, symmetry regularizer} \}$$



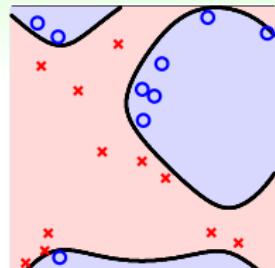
$$\lambda \in \{ 0, 0.01, 1 \}$$

in addition to your **favorite** combination, may need to try other combinations to get a good g

Model Selection Problem

 \mathcal{H}_1

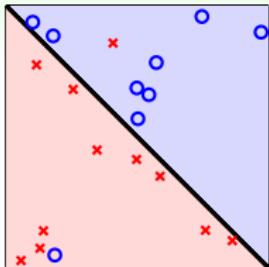
which one do you prefer? :-)

 \mathcal{H}_2

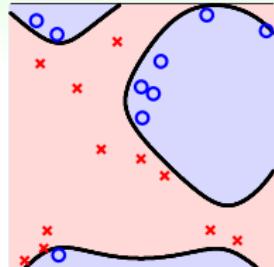
- given: M models $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_M$, each with corresponding algorithm $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_M$
- goal: select \mathcal{H}_{m^*} such that $g_{m^*} = \mathcal{A}_{m^*}(\mathcal{D})$ is of low $E_{\text{out}}(g_{m^*})$
- unknown E_{out} due to unknown $P(\mathbf{x})$ & $P(y|\mathbf{x})$, as always :-)
- arguably the **most important** practical problem of ML

how to select? **visually?**
—no, remember Lecture 12? :-)

Model Selection by Best E_{in}

 \mathcal{H}_1 select by best E_{in} ?

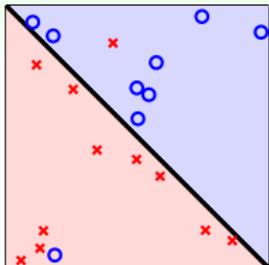
$$m^* = \operatorname{argmin}_{1 \leq m \leq M} (E_m = E_{\text{in}}(\mathcal{A}_m(\mathcal{D})))$$

 \mathcal{H}_2

- Φ_{1126} always more preferred over Φ_1 ;
 $\lambda = 0$ always more preferred over $\lambda = 0.1$ —**overfitting?**
- if \mathcal{A}_1 minimizes E_{in} over \mathcal{H}_1 and \mathcal{A}_2 minimizes E_{in} over \mathcal{H}_2 ,
 $\implies g_{m^*}$ achieves minimal E_{in} over $\mathcal{H}_1 \cup \mathcal{H}_2$
 \implies ‘model selection + learning’ pays $d_{\text{VC}}(\mathcal{H}_1 \cup \mathcal{H}_2)$
—**bad generalization?**

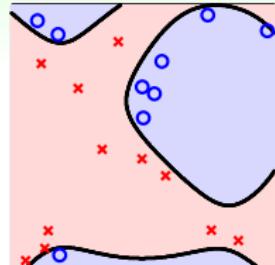
selecting by E_{in} is **dangerous**

Model Selection by Best E_{test}

 \mathcal{H}_1

select by best E_{test} , which is evaluated on a fresh $\mathcal{D}_{\text{test}}$?

$$m^* = \operatorname{argmin}_{1 \leq m \leq M} (E_m = E_{\text{test}}(\mathcal{A}_m(\mathcal{D})))$$

 \mathcal{H}_2

- generalization guarantee (finite-bin Hoeffding):

$$E_{\text{out}}(g_{m^*}) \leq E_{\text{test}}(g_{m^*}) + O\left(\sqrt{\frac{\log M}{N_{\text{test}}}}\right)$$

—yes! strong guarantee :-)

- but where is $\mathcal{D}_{\text{test}}$?—your boss's safe, maybe? :-)

selecting by E_{test} is **infeasible** and **cheating**

Comparison between E_{in} and E_{test}

in-sample error E_{in}

- calculated from \mathcal{D}
- **feasible** on hand
- ‘contaminated’ as \mathcal{D} also used by \mathcal{A}_m to ‘select’ g_m

test error E_{test}

- calculated from $\mathcal{D}_{\text{test}}$
- **infeasible** in boss’s safe
- ‘clean’ as $\mathcal{D}_{\text{test}}$ never used for selection before

something in between: E_{val}

- calculated from $\mathcal{D}_{\text{val}} \subset \mathcal{D}$
- **feasible** on hand
- ‘clean’ **if** \mathcal{D}_{val} never used by \mathcal{A}_m before

selecting by E_{val} : **legal cheating :-)**

Fun Time

For $\mathcal{X} = \mathbb{R}^d$, consider two hypothesis sets, \mathcal{H}_+ and \mathcal{H}_- . The first hypothesis set contains all perceptrons with $w_1 \geq 0$, and the second hypothesis set contains all perceptrons with $w_1 \leq 0$. Denote g_+ and g_- as the minimum- E_{in} hypothesis in each hypothesis set, respectively. Which statement below is true?

- 1 If $E_{\text{in}}(g_+) < E_{\text{in}}(g_-)$, then g_+ is the minimum- E_{in} hypothesis of all perceptrons in \mathbb{R}^d .
- 2 If $E_{\text{test}}(g_+) < E_{\text{test}}(g_-)$, then g_+ is the minimum- E_{test} hypothesis of all perceptrons in \mathbb{R}^d .
- 3 The two hypothesis sets are disjoint.
- 4 None of the above

Fun Time

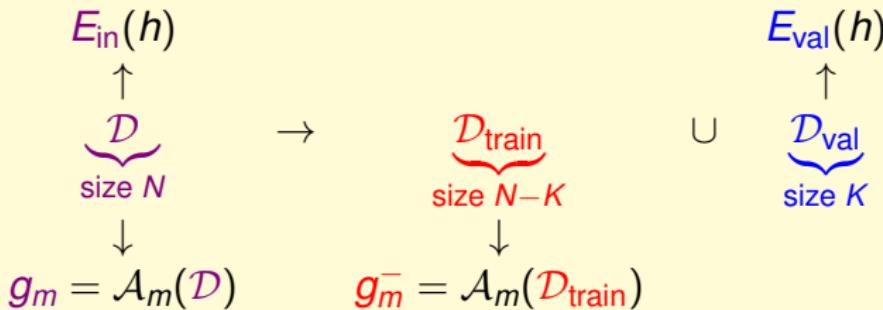
For $\mathcal{X} = \mathbb{R}^d$, consider two hypothesis sets, \mathcal{H}_+ and \mathcal{H}_- . The first hypothesis set contains all perceptrons with $w_1 \geq 0$, and the second hypothesis set contains all perceptrons with $w_1 \leq 0$. Denote g_+ and g_- as the minimum- E_{in} hypothesis in each hypothesis set, respectively. Which statement below is true?

- ① If $E_{\text{in}}(g_+) < E_{\text{in}}(g_-)$, then g_+ is the minimum- E_{in} hypothesis of all perceptrons in \mathbb{R}^d .
- ② If $E_{\text{test}}(g_+) < E_{\text{test}}(g_-)$, then g_+ is the minimum- E_{test} hypothesis of all perceptrons in \mathbb{R}^d .
- ③ The two hypothesis sets are disjoint.
- ④ None of the above

Reference Answer: ①

Note that the two hypothesis sets are not disjoint (sharing ' $w_1 = 0$ ' perceptrons) but their union is all perceptrons.

Validation Set \mathcal{D}_{val}



- $\mathcal{D}_{\text{val}} \subset \mathcal{D}$: called **validation set**—‘on-hand’ simulation of test set
- to connect E_{val} with E_{out} :
 $\mathcal{D}_{\text{val}} \stackrel{iid}{\sim} P(\mathbf{x}, y) \iff$ select K examples from \mathcal{D} at random
- to make sure \mathcal{D}_{val} ‘clean’: feed only $\mathcal{D}_{\text{train}}$ to \mathcal{A}_m for model selection

$$E_{\text{out}}(g_m^-) \leq E_{\text{val}}(g_m^-) + O\left(\sqrt{\frac{\log M}{K}}\right)$$

Model Selection by Best E_{val}

$$m^* = \underset{1 \leq m \leq M}{\operatorname{argmin}}(E_m = E_{\text{val}}(\mathcal{A}_m(\mathcal{D}_{\text{train}})))$$

- generalization guarantee for all m :

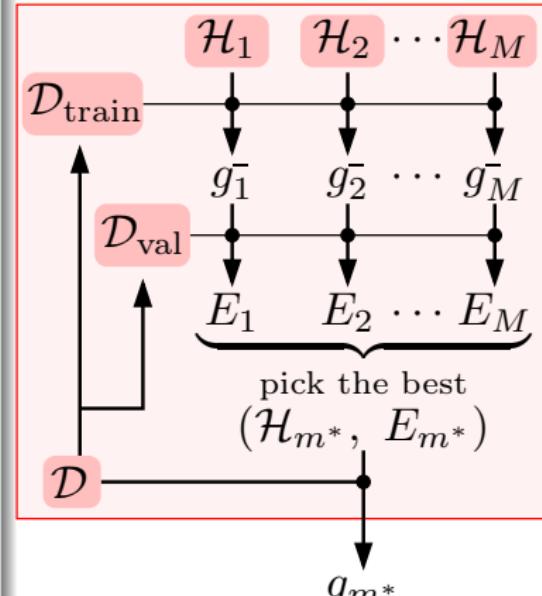
$$E_{\text{out}}(\mathbf{g}_m^-) \leq E_{\text{val}}(\mathbf{g}_m^-) + O\left(\sqrt{\frac{\log M}{K}}\right)$$

- heuristic gain from $N - K$ to N :

$$E_{\text{out}}\left(\underbrace{\mathbf{g}_{m^*}}_{\mathcal{A}_{m^*}(\mathcal{D})}\right) \leq E_{\text{out}}\left(\underbrace{\mathbf{g}_{m^*}^-}_{\mathcal{A}_{m^*}(\mathcal{D}_{\text{train}})}\right)$$

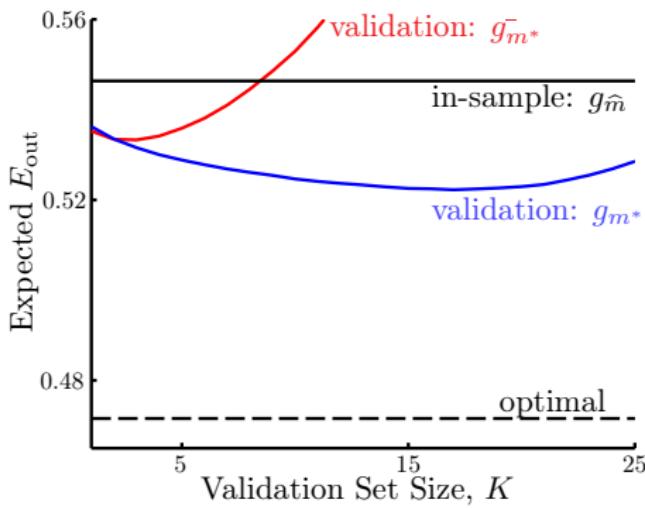
—learning curve, remember? :-)

$$E_{\text{out}}(\mathbf{g}_{m^*}) \leq E_{\text{out}}(\mathbf{g}_{m^*}^-) \leq E_{\text{val}}(\mathbf{g}_{m^*}^-) + O\left(\sqrt{\frac{\log M}{K}}\right)$$



Validation in Practice

use validation to select between \mathcal{H}_{Φ_5} and $\mathcal{H}_{\Phi_{10}}$



- in-sample: selection with E_{in}
 - optimal: cheating-selection with E_{test}
 - **sub-g:** selection with E_{val} and report \bar{g}_{m^*}
 - **full-g:** selection with E_{val} and report g_{m^*}
- $-E_{out}(g_{m^*}) \leq E_{out}(\bar{g}_{m^*})$
indeed

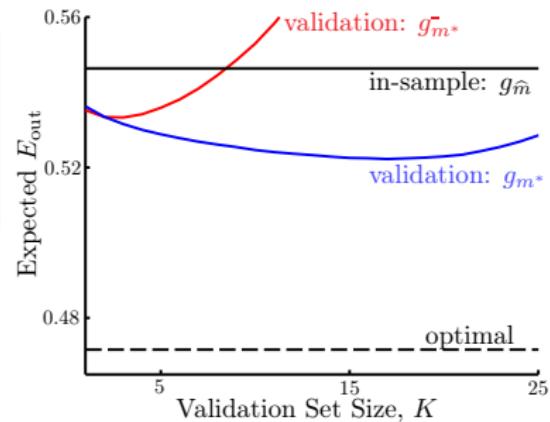
why is **sub-g** worse than in-sample some time?

The Dilemma about K

reasoning of validation:

$$E_{\text{out}}(g) \approx E_{\text{out}}(g^-) \quad (\text{small } K) \qquad \qquad E_{\text{out}}(g^-) \approx E_{\text{val}}(g^-) \quad (\text{large } K)$$

- large K : every $E_{\text{val}} \approx E_{\text{out}}$, but all g_m^- much worse than g_m
- small K : every $g_m^- \approx g_m$, but E_{val} far from E_{out}



practical rule of thumb: $K = \frac{N}{5}$

Fun Time

For a learning model that takes N^2 seconds of training when using N examples, what is the total amount of seconds needed when running the whole validation procedure with $K = \frac{N}{5}$ on 25 such models with different parameters to get the final g_{m^*} ?

- 1 $6N^2$
- 2 $17N^2$
- 3 $25N^2$
- 4 $26N^2$

Fun Time

For a learning model that takes N^2 seconds of training when using N examples, what is the total amount of seconds needed when running the whole validation procedure with $K = \frac{N}{5}$ on 25 such models with different parameters to get the final g_{m^*} ?

- 1 $6N^2$
- 2 $17N^2$
- 3 $25N^2$
- 4 $26N^2$

Reference Answer: (2)

To get all the $\bar{g_m}$, we need $\frac{16}{25}N^2 \cdot 25$ seconds.
Then to get g_{m^*} , we need another N^2 seconds.
So in total we need $17N^2$ seconds.

Extreme Case: $K = 1$

reasoning of validation:

$$E_{\text{out}}(\textcolor{violet}{g}) \approx E_{\text{out}}(\textcolor{red}{g}^-) \approx E_{\text{val}}(\textcolor{red}{g}^-)$$

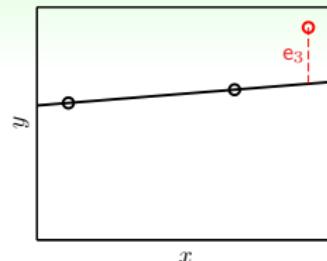
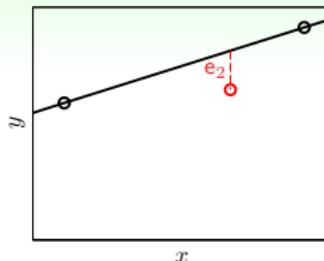
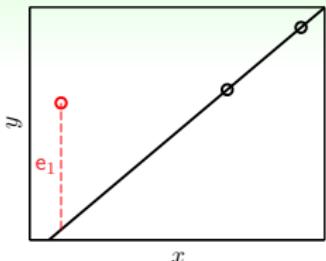
(small K) **(large K)**

- take $K = 1$? $\mathcal{D}_{\text{val}}^{(n)} = \{(\mathbf{x}_n, y_n)\}$ and $E_{\text{val}}^{(n)}(\mathbf{g}_n^-) = \text{err}(\mathbf{g}_n^-(\mathbf{x}_n), y_n) = e_n$
 - make e_n closer to $E_{\text{out}}(\mathbf{g})$?—average over possible $E_{\text{val}}^{(n)}$
 - leave-one-out cross validation estimate:

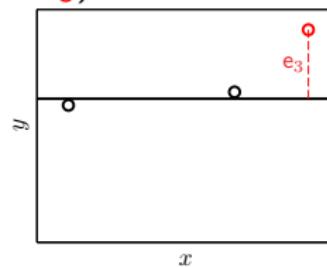
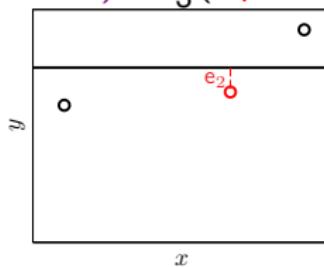
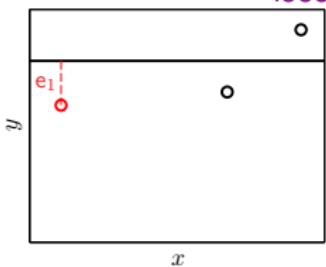
$$E_{\text{loocv}}(\mathcal{H}, \mathcal{A}) = \frac{1}{N} \sum_{n=1}^N e_n = \frac{1}{N} \sum_{n=1}^N \text{err}(\mathbf{g}_n^-(\mathbf{x}_n), y_n)$$

hope: $E_{\text{loocv}}(\mathcal{H}, \mathcal{A}) \approx E_{\text{out}}(g)$

Illustration of Leave-One-Out



$$E_{\text{loocv}}(\text{linear}) = \frac{1}{3}(e_1 + e_2 + e_3)$$



$$E_{\text{loocv}}(\text{constant}) = \frac{1}{3}(e_1 + e_2 + e_3)$$

which one would you choose?

$$m^* = \operatorname{argmin}_{1 \leq m \leq M} (E_m = E_{\text{loocv}}(\mathcal{H}_m, \mathcal{A}_m))$$

Theoretical Guarantee of Leave-One-Out Estimate

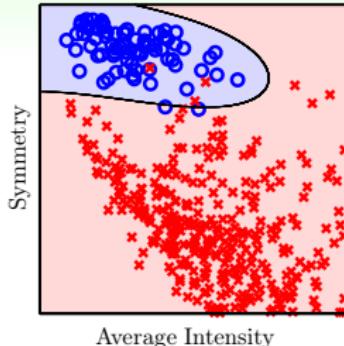
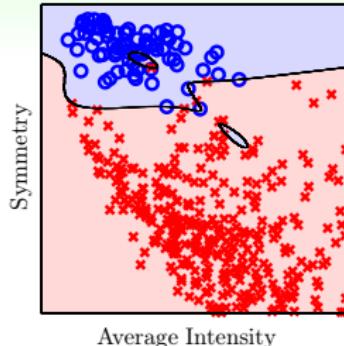
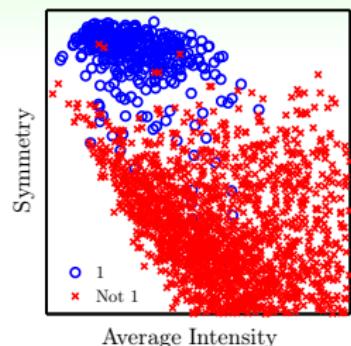
does $E_{\text{loocv}}(\mathcal{H}, \mathcal{A})$ say something about $E_{\text{out}}(g)$?

yes, for average E_{out} on size- $(N - 1)$ data

$$\begin{aligned}
 \mathop{\mathcal{E}}_{\mathcal{D}} E_{\text{loocv}}(\mathcal{H}, \mathcal{A}) &= \mathop{\mathcal{E}}_{\mathcal{D}} \frac{1}{N} \sum_{n=1}^N e_n = \frac{1}{N} \sum_{n=1}^N \mathop{\mathcal{E}}_{\mathcal{D}} e_n \\
 &= \frac{1}{N} \sum_{n=1}^N \mathop{\mathcal{E}}_{\mathcal{D}_n(\mathbf{x}_n, y_n)} \mathop{\mathcal{E}}_{\mathcal{D}} \text{err}(g_n^-(\mathbf{x}_n), y_n) \\
 &= \frac{1}{N} \sum_{n=1}^N \mathop{\mathcal{E}}_{\mathcal{D}_n} E_{\text{out}}(g_n^-) \\
 &= \frac{1}{N} \sum_{n=1}^N \overline{E_{\text{out}}}(N-1) = \overline{E_{\text{out}}}(N-1)
 \end{aligned}$$

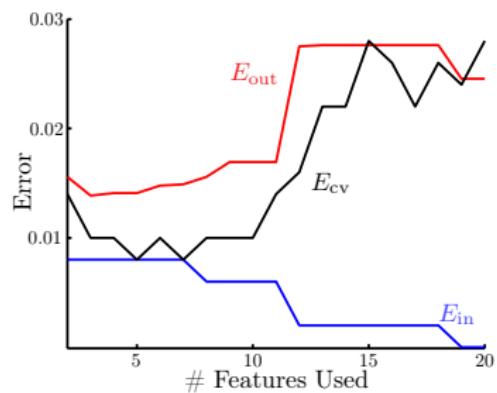
expected $E_{\text{loocv}}(\mathcal{H}, \mathcal{A})$ says something about expected $E_{\text{out}}(g^-)$
 —often called ‘almost unbiased estimate of $E_{\text{out}}(g)$ ’

Leave-One-Out in Practice



select by E_{in}

select by E_{loocv}



E_{loocv} much better than E_{in}

Fun Time

Consider three examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3)$ with $y_1 = 1, y_2 = 5, y_3 = 7$. If we use E_{loocv} to estimate the performance of a learning algorithm that predicts with the average y value of the data set—the optimal constant prediction with respect to the squared error. What is E_{loocv} (squared error) of the algorithm?

- 1 0
- 2 $\frac{56}{9}$
- 3 $\frac{60}{9}$
- 4 14

Fun Time

Consider three examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3)$ with $y_1 = 1, y_2 = 5, y_3 = 7$. If we use E_{loocv} to estimate the performance of a learning algorithm that predicts with the average y value of the data set—the optimal constant prediction with respect to the squared error. What is E_{loocv} (squared error) of the algorithm?

- 1 0
- 2 $\frac{56}{9}$
- 3 $\frac{60}{9}$
- 4 14

Reference Answer: 4

This is based on a simple calculation of
 $e_1 = (1 - 6)^2, e_2 = (5 - 4)^2, e_3 = (7 - 3)^2.$

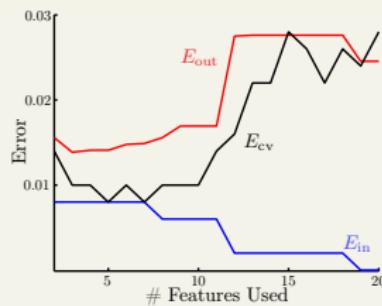
Disadvantages of Leave-One-Out Estimate

Computation

$$E_{\text{loocv}}(\mathcal{H}, \mathcal{A}) = \frac{1}{N} \sum_{n=1}^N e_n = \frac{1}{N} \sum_{n=1}^N \text{err}(g_n^-(\mathbf{x}_n), y_n)$$

- N ‘additional’ training per model, not always feasible in practice
- except ‘special case’ like analytic solution for linear regression

Stability—due to variance of single-point estimates

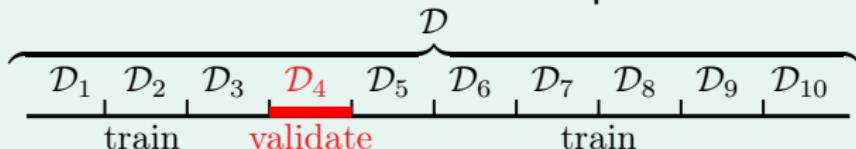


E_{loocv} : not often used practically

V-fold Cross Validation

how to **decrease computation need** for cross validation?

- essence of leave-one-out cross validation: partition \mathcal{D} to N parts, taking $N - 1$ for training and 1 for validation orderly
- V -fold cross-validation: random-partition of \mathcal{D} **to V equal parts**,



take $V - 1$ for training and 1 for validation orderly

$$E_{cv}(\mathcal{H}, \mathcal{A}) = \frac{1}{V} \sum_{v=1}^V E_{val}^{(v)}(\mathbf{g}_v^-)$$

- selection by E_{cv} : $m^* = \operatorname{argmin}_{1 \leq m \leq M} (E_m = E_{cv}(\mathcal{H}_m, \mathcal{A}_m))$

practical rule of thumb: $V = 10$

Final Words on Validation

'Selecting' Validation Tool

- **V-Fold** generally preferred over single validation if computation allows
- **5-Fold or 10-Fold** generally works well:
not necessary to trade V-Fold with Leave-One-Out

Nature of Validation

- all training models: select among hypotheses
- all validation schemes: **select among finalists**
- all testing methods: just **evaluate**

validation still **more optimistic than testing**

do not fool yourself and others :-),
report test result, not **best validation result**

Fun Time

For a learning model that takes N^2 seconds of training when using N examples, what is the total amount of seconds needed when running 10-fold cross validation on 25 such models with different parameters to get the final g_{m^*} ?

- 1 $\frac{47}{2}N^2$
- 2 $47N^2$
- 3 $\frac{407}{2}N^2$
- 4 $407N^2$

Fun Time

For a learning model that takes N^2 seconds of training when using N examples, what is the total amount of seconds needed when running 10-fold cross validation on 25 such models with different parameters to get the final g_{m^*} ?

- 1 $\frac{47}{2}N^2$
- 2 $47N^2$
- 3 $\frac{407}{2}N^2$
- 4 $407N^2$

Reference Answer: ③

To get all the E_{cv} , we need $\frac{81}{100}N^2 \cdot 10 \cdot 25$ seconds. Then to get g_{m^*} , we need another N^2 seconds. So in total we need $\frac{407}{2}N^2$ seconds.

Summary

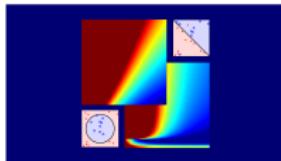
- ① When Can Machines Learn?
- ② Why Can Machines Learn?
- ③ How Can Machines Learn?
- ④ How Can Machines Learn **Better?**

Lecture 14: Regularization

Lecture 15: Validation

- Model Selection Problem
dangerous by E_{in} and dishonest by E_{test}
 - Validation
select with $E_{\text{val}}(\mathcal{D}_{\text{train}})$ while returning $\mathcal{A}_{m^*}(\mathcal{D})$
 - Leave-One-Out Cross Validation
huge computation for almost unbiased estimate
 - V-Fold Cross Validation
reasonable computation and performance
-
- next: something ‘up my sleeve’

Machine Learning Foundations (機器學習基石)



Lecture 16: Three Learning Principles

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?
- ② Why Can Machines Learn?
- ③ How Can Machines Learn?
- ④ How Can Machines Learn **Better?**

Lecture 15: Validation

(crossly) reserve **validation data** to simulate testing procedure for **model selection**

Lecture 16: Three Learning Principles

- Occam's Razor
- Sampling Bias
- Data Snooping
- Power of Three

Occam's Razor

An explanation of the data should be made as simple as possible, but no simpler.—Albert Einstein? (1879-1955)

entia non sunt multiplicanda praeter necessitatem
(entities must not be multiplied **beyond necessity**)
—William of Occam (1287-1347)

'Occam's razor' for trimming down unnecessary explanation

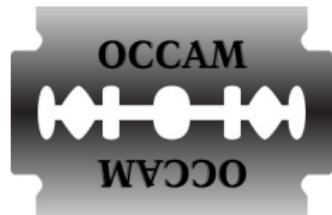
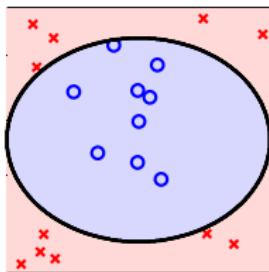


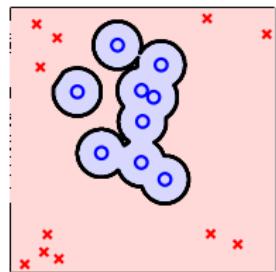
figure by Fred the Oyster (Own work) [CC-BY-SA-3.0], via Wikimedia Commons

Occam's Razor for Learning

The simplest model that fits the data is also the most plausible.



which one do you prefer? :-)



two questions:

- ① What does it mean for a model to be simple?
- ② How do we know that simpler is better?

Simple Model

simple hypothesis h

- small $\Omega(h)$ = ‘looks’ simple
- specified by **few parameters**

simple model \mathcal{H}

- small $\Omega(\mathcal{H})$ = not many
- contains **small number of hypotheses**

connection

h specified by ℓ bits $\Leftarrow |\mathcal{H}|$ of size 2^ℓ

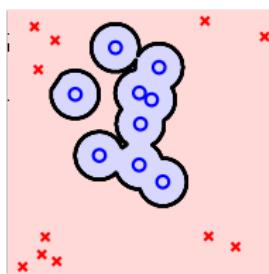
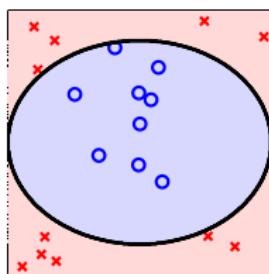
small $\Omega(h) \Leftarrow$ small $\Omega(\mathcal{H})$

simple: **small hypothesis/model complexity**

Simple is Better

in addition to **math proof** that you have seen, philosophically:

- simple \mathcal{H}
- \implies smaller $m_{\mathcal{H}}(N)$
- \implies less 'likely' to fit data perfectly $\frac{m_{\mathcal{H}}(N)}{2^N}$
- \implies more significant when fit happens



direct action: **linear first**;
always ask whether **data over-modeled**

Fun Time

Consider the decision stumps in \mathbb{R}^1 as the hypothesis set \mathcal{H} . Recall that $m_{\mathcal{H}}(N) = 2N$. Consider 10 different inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{10}$ coupled with labels y_n generated iid from a fair coin. What is the probability that the data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{10}$ is separable by \mathcal{H} ?

- 1 $\frac{1}{1024}$
- 2 $\frac{10}{1024}$
- 3 $\frac{20}{1024}$
- 4 $\frac{100}{1024}$

Fun Time

Consider the decision stumps in \mathbb{R}^1 as the hypothesis set \mathcal{H} . Recall that $m_{\mathcal{H}}(N) = 2N$. Consider 10 different inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{10}$ coupled with labels y_n generated iid from a fair coin. What is the probability that the data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{10}$ is separable by \mathcal{H} ?

- 1 $\frac{1}{1024}$
- 2 $\frac{10}{1024}$
- 3 $\frac{20}{1024}$
- 4 $\frac{100}{1024}$

Reference Answer: ③

Of all 1024 possible \mathcal{D} , only $2N = 20$ of them is separable by \mathcal{H} .

Presidential Story

- 1948 US President election: Truman versus Dewey
- a newspaper phone-poll of how people **voted**,
and set the title '**Dewey Defeats Truman**' based on polling



who is this? :-)

The Big Smile Came from ...



Truman, and **yes he won**

suspect of the mistake:

- editorial bug?—**no**
- bad luck of polling (δ)?—**no**

hint: phones were **expensive :-)**

Sampling Bias

If the data is sampled in a biased way, learning will produce a similarly biased outcome.

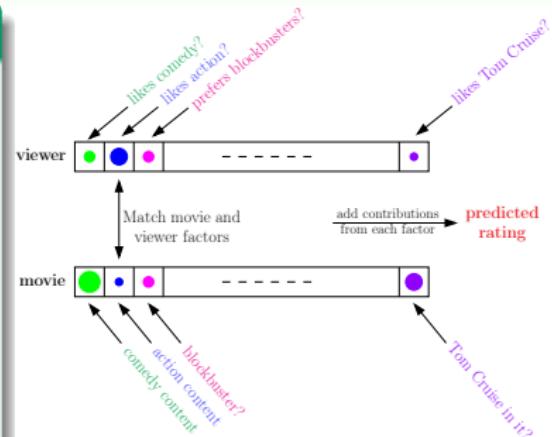
- technical explanation:
data from $P_1(x, y)$ but test under $P_2 \neq P_1$: **VC fails**
- philosophical explanation:
study Math hard but test English: **no strong test guarantee**

‘minor’ VC assumption:
data and testing **both iid from P**

Sampling Bias in Learning

A True Personal Story

- Netflix competition for movie recommender system:
10% improvement = 1M US dollars
- formed \mathcal{D}_{val} ,
in my **first shot**,
 $E_{\text{val}}(g)$ showed **13% improvement**
- **why am I still teaching here? :-)**



validation: random examples within \mathcal{D} ;
test: 'last' user records 'after' \mathcal{D}

Dealing with Sampling Bias

If the data is sampled in a biased way, learning will produce a similarly biased outcome.

- practical rule of thumb:
match test scenario as much as possible
- e.g. if test: ‘last’ user records ‘after’ \mathcal{D}
 - training: emphasize later examples (KDDCup 2011)
 - validation: use ‘late’ user records

last puzzle:

danger when learning ‘credit card approval’
with **existing bank records**?

Fun Time

If the data \mathcal{D} is an unbiased sample from the underlying distribution P for binary classification, which of the following subset of \mathcal{D} is also an unbiased sample from P ?

- ① all the positive ($y_n > 0$) examples
- ② half of the examples that are randomly and uniformly picked from \mathcal{D} without replacement
- ③ half of the examples with the smallest $\|\mathbf{x}_n\|$ values
- ④ the largest subset that is linearly separable

Fun Time

If the data \mathcal{D} is an unbiased sample from the underlying distribution P for binary classification, which of the following subset of \mathcal{D} is also an unbiased sample from P ?

- ① all the positive ($y_n > 0$) examples
- ② half of the examples that are randomly and uniformly picked from \mathcal{D} without replacement
- ③ half of the examples with the smallest $\|\mathbf{x}_n\|$ values
- ④ the largest subset that is linearly separable

Reference Answer: ②

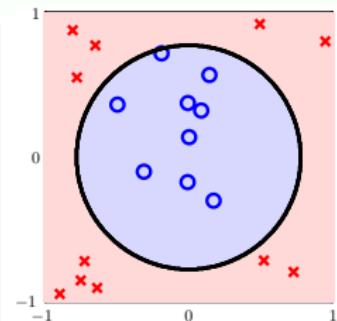
That's how we form the validation set,
remember? :-)

Visual Data Snooping

Visualize $\mathcal{X} = \mathbb{R}^2$

- full Φ_2 : $\mathbf{z} = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$, $d_{VC} = 6$
- or $\mathbf{z} = (1, x_1^2, x_2^2)$, $d_{VC} = 3$, **after visualizing?**
- or better $\mathbf{z} = (1, x_1^2 + x_2^2)$, $d_{VC} = 2$?
- or even better $\mathbf{z} = (\text{sign}(0.6 - x_1^2 - x_2^2))$?

—careful about **your brain's 'model complexity'**

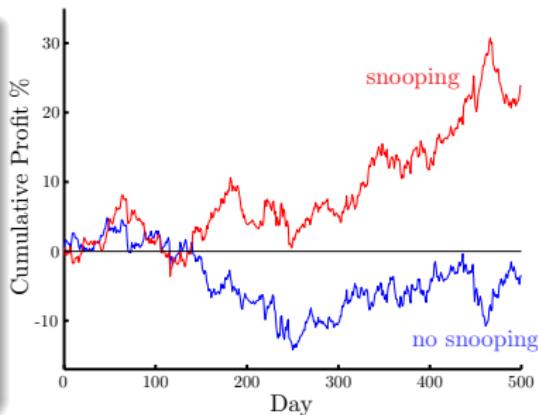


for VC-safety, Φ shall be decided **without 'snooping'** data

Data Snooping by Mere Shifting-Scaling

If a data set has affected any step in the learning process, its ability to assess the outcome has been compromised.

- 8 years of currency trading data
- first 6 years for **training**,
last two 2 years for **testing**
- x = previous 20 days,
 y = 21th day
- **snooping** versus **no snooping**:
superior profit possible



- **snooping**: shift-scale all values by **training + testing**
- **no snooping**: shift-scale all values by **training** only

Data Snooping by Data Reusing

Research Scenario

benchmark data \mathcal{D}

- paper 1: propose \mathcal{H}_1 that works well on \mathcal{D}
 - paper 2: find room for improvement, propose \mathcal{H}_2
 - and **publish only if better** than \mathcal{H}_1 on \mathcal{D}
 - paper 3: find room for improvement, propose \mathcal{H}_3
 - and **publish only if better** than \mathcal{H}_2 on \mathcal{D}
 - ...
-
- if all papers from the same author in **one big paper**:
bad generalization due to $d_{VC}(\cup_m \mathcal{H}_m)$
 - step-wise: later author **snooped** data by reading earlier papers,
bad generalization worsen by **publish only if better**

if you torture the data long enough, it will confess :-)

Dealing with Data Snooping

- truth—**very hard to avoid**, unless being extremely honest
 - extremely honest: **lock your test data in safe**
 - less honest: **reserve validation and use cautiously**
-
- be blind: avoid **making modeling decision by data**
 - be suspicious: interpret research results (including your own) by proper **feeling of contamination**

one secret to winning KDDCups:

careful balance between
data-driven modeling (snooping) and
validation (no-snooping)

Fun Time

Which of the following can result in unsatisfactory test performance in machine learning?

- ① data snooping
- ② overfitting
- ③ sampling bias
- ④ all of the above

Fun Time

Which of the following can result in unsatisfactory test performance in machine learning?

- ① data snooping
- ② overfitting
- ③ sampling bias
- ④ all of the above

Reference Answer: ④

A professional like you should be aware of those! :-)

Three Related Fields

Power of Three

Data Mining

- use **(huge)** data to **find property** that is interesting
- difficult to distinguish ML and DM in reality

Artificial Intelligence

- compute something that shows **intelligent behavior**
- ML is one possible route to realize AI

Statistics

- use data to **make inference** about an unknown process
- statistics contains many useful tools for ML

Three Theoretical Bounds

Power of Three

Hoeffding

$$\begin{aligned} P[\text{BAD}] \\ \leq 2 \exp(-2\epsilon^2 N) \end{aligned}$$

- one hypothesis
- useful for verifying/testing

Multi-Bin Hoeffding

$$\begin{aligned} P[\text{BAD}] \\ \leq 2M \exp(-2\epsilon^2 N) \end{aligned}$$

- M hypotheses
- useful for validation

VC

$$\begin{aligned} P[\text{BAD}] \\ \leq 4m_{\mathcal{H}}(2N) \exp(\dots) \end{aligned}$$

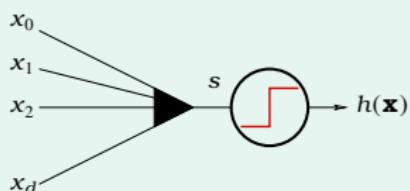
- all \mathcal{H}
- useful for training

Three Linear Models

Power of Three

PLA/pocket

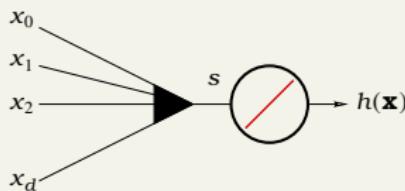
$$h(\mathbf{x}) = \text{sign}(s)$$



plausible err = 0/1
 (small flipping noise)
 minimize **specially**

linear regression

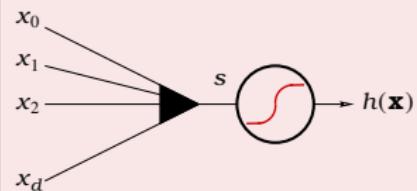
$$h(\mathbf{x}) = s$$



friendly err = squared
 (easy to minimize)
 minimize **analytically**

logistic regression

$$h(\mathbf{x}) = \theta(s)$$



plausible err = CE
 (maximum likelihood)
 minimize **iteratively**

Three Key Tools

Power of Three

Feature Transform

$$\begin{aligned} E_{\text{in}}(\mathbf{w}) &\rightarrow E_{\text{in}}(\tilde{\mathbf{w}}) \\ d_{\text{VC}}(\mathcal{H}) &\rightarrow d_{\text{VC}}(\mathcal{H}_\Phi) \end{aligned}$$

- by using **more complicated Φ**
- **lower E_{in}**
- **higher d_{VC}**

Regularization

$$\begin{aligned} E_{\text{in}}(\mathbf{w}) &\rightarrow E_{\text{in}}(\mathbf{w}_{\text{REG}}) \\ d_{\text{VC}}(\mathcal{H}) &\rightarrow d_{\text{EFF}}(\mathcal{H}, \mathcal{A}) \end{aligned}$$

- by augmenting **regularizer Ω**
- **lower d_{EFF}**
- **higher E_{in}**

Validation

$$\begin{aligned} E_{\text{in}}(h) &\rightarrow E_{\text{val}}(h) \\ \mathcal{H} &\rightarrow \{g_1^-, \dots, g_M^-\} \end{aligned}$$

- by reserving **K examples as \mathcal{D}_{val}**
- **fewer choices**
- **fewer examples**

Three Learning Principles

Power of Three

Occam's Razer

simple is good

Sampling Bias

class matches exam

Data Snooping

honesty is best policy

Three Future Directions

Power of Three

More Transform

More Regularization

Less Label

bagging decision tree support vector machine neural network kernel
AdaBoost aggregation sparsity autoencoder coordinate descent

dual uniform blending deep learning nearest neighbor decision stump

kernel LogReg large-margin prototype quadratic programming SVR

GBDT PCA random forest matrix factorization Gaussian kernel
soft-margin k-means OOB error RBF network probabilistic SVM

ready for the **jungle!**

Fun Time

What are the magic numbers that repeatedly appear in this class?

- 1 3
- 2 1126
- 3 both 3 and 1126
- 4 neither 3 nor 1126

Fun Time

What are the magic numbers that repeatedly appear in this class?

- 1 3
- 2 1126
- 3 both 3 and 1126
- 4 neither 3 nor 1126

Reference Answer: ③

3 as illustrated, and **you may recall 1126 somewhere :-)**

Summary

- ① When Can Machines Learn?
- ② Why Can Machines Learn?
- ③ How Can Machines Learn?
- ④ How Can Machines Learn **Better?**

Lecture 15: Validation

Lecture 16: Three Learning Principles

- Occam's Razor
simple, simple, simple!
 - Sampling Bias
match test scenario as much as possible
 - Data Snooping
any use of data is 'contamination'
 - Power of Three
relatives, bounds, models, tools, principles
-
- next: ready for jungle!