Date 2021. 1. 21.

机器学习

→ 机器自动找函式

↳ For Example

1. 語音辨識
   輸入：聲音訊號
   輸出：語言辨識結果（文字）
2. 影像辨識
   輸入 = 图片
   輸出 = 图片有什麼樣的物件/東西
3. playing Go
   輸入：棋盤上黑子及白子的位置
   輸出 = 下一部落子的位置
4. 对話式系統
   輸入：你要对机器說的話
   輸出 = 机器的回应

想要找什麼函式？

- Regression
  輸出 = 數值
  Want 机器預測未來某個時刻的 PM 2.5
  ⇒ 找 function 輸入 = 過去的臭氧濃度/天氣等
  出 明天某一時刻的 PM2.5 數值

- binary classification
  輸出 = Yes / No
  eg: RNN = 輸入：一个句子
  輸出 = 句子是正面/負面

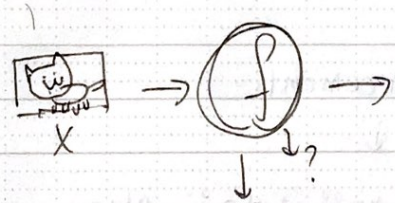
Date

- Multi-class classification
  輸入：讓机器做選擇題（n个選项）
  出：選出一个 ANS
  ↳ eg: CNN
  · 輸入一個图片，也是A一个麵包
  B: 蛋
  C: 湯
  which ?

- Generation（生成）
  → 產生有結构的複雜東
  → 讓机器學習如何創造
  eg: A 翻譯：產生文句
  B: 產生二次元人物

怎麼告訴机器根找什麼樣的函式?

- Supervised Learning
  · suppose 已經想好 輸入是一張图片：x
  輸入是這张图片屬於什麼樣的類別：y



need 訓練資料（Labeled Data）
↳ need 收集大量图片
⊖ 理想/正確的輸出是什麼
↳ 也就是 Labeled 的意思

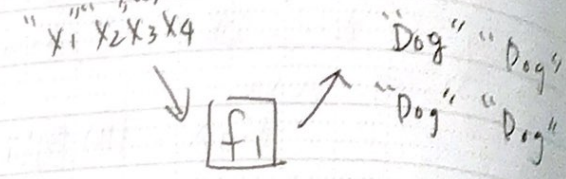
→ 提供給机器有 labeled 的資料進行學習
称 "Supervised Learning"

then 机器就可以評估一个函式的好壞 ∋ Loss

Loss Function

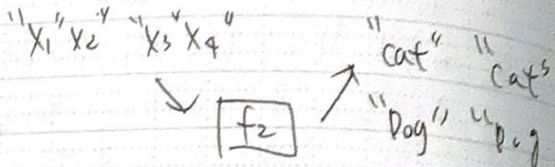
Labeled Data

x_1		y_1 = cat
x_2		y_2 = cat
x_3		y_3 = Dog
x_4		y_4 = Dog



Loss = 50 %

↳ 越小越好



Loss = 0 %

接下來机器会自动找出 Loss 最低的函式

↓ will use 什麼樣的演算法找最低的 Loss

Reinforcement Learning

- alpha Go

Supervised V.S. Reinforcement

↓

Now

Next move

"2-3"

↓

机器 & 机器 play

机器 & somebody play ⇒ win

Now

Next

3-3

↓

机器想辦法自己提高正確率

win / lose ⇒ Reward (引導)

Date

· Unsupervised Learning
→ 給一大堆圖片，but 沒有標注

机器怎麼找出你想要的函式
First：給定函式尋找的範圍
Ex A: linear function (前兩个HW)
B: Network Architecture
Use RNN : Seq to Seq
CNN = alot of
ex Meta Learning
Unsupervied Learning
Anomaly Detection
Transfer Learning
Explainable AI
⋮

函式尋找方法 - Gradient Descent
前兩个作业: Implement the algorithm by yourself

Date

前沿研究

- Explainable AI
  - why You(机器) think the picture is Cat ?
- Adversarial Attack
  - If 恶意攻擊 system 会發生 什麼事
  - If the 入的雜訊 不是一般的，是魔刻意制造的？
- Network compression
  - 把很大的 Network 縮小
    放入小的 page device 上

Use by CNN 結果

- Anomaly Detection
  If 看到 訓練沒看到的，能否知道要 say "我不知道"
- Domain Adversarial
  [Training Data
    Testing Data ⇒ have same distribution
  ex 手寫辨識 ⇒ 會有很高的正確率
  but if Training data is 黑白
  Testing data is colorful
  ⇒ 正確率 57.5% Very low

Date

- Meta Learning
  → learn to learn
  (學習如何學習)
  讓机器自己證明一个学習 Algo

- Life-long Learning (終身学習)
  i.e Continuous Learning, Never Ending Learning
  → what 又 tt. 難矣?

Date 110 1 23

Regression : case study

For example.

1、Stock Market Forecast (股票預測系統)

f(各种股票起伏的data) = 明天道瓊工業指數的漲數

2、Self-driving car (无人車、自动車)

f(无人車的紅外線感測的sensor) = 方向盤的角度

3、Recommendation (推薦 system)

f(使用者A 商品B) = 使用者A 購買商品 B的可能性

4、預測 pokemon的 cp值

f(某一隻 pokemon 的information) = 進化后的cp值 ↓ y

↓包含 x_{cp}, x_s (物种), x_{hp} (生命值), x_w (重量), x_h (高度)

ML step

1. 找一个 model (function set)
2. 定义 function set 裡某个function
   找一个 function, then evaluation 度好上裏
3. 找一个最好的 function

Step1 Model. = $y = b + w \cdot x_{cp}$

y: 進化后cp; b, w: 常數/參數; x_{cp}: 進化前 cp

Linear model $y = b + \sum w_i x_i$

b: bias; w_i: weight; x_i: Input x的各种 attribute (ex pokemon 属性)

Date

Step 2: Goodness of function

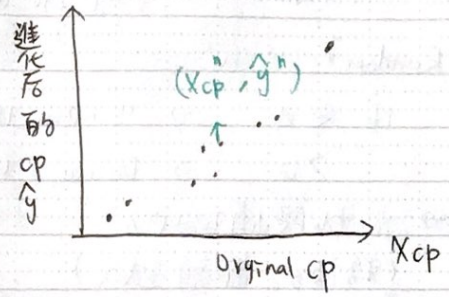
→ 收集 training data，才能找到這個 function

function input: x^1 = 水違　x^2 = 伊布

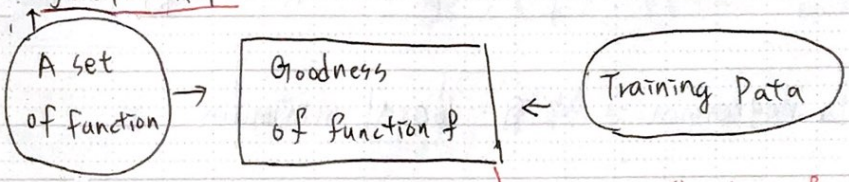
output　$\hat{y}^1$ = 999　$\hat{y}^2$ = 1420

Training Data = 10 pokemons

$(x^1, \hat{y}^1)$ $(x^2, \hat{y}^2)$ ---- $(x^{10}, \hat{y}^{10})$



$y = b + w \cdot x_{cp}$



pick the best function: gradient Descent
L need可微

Loss function L

[Input: a function.
Output: how bad it is

$L(f) = L(w,b)$ // 衡量一組參數的好壞

$$= \sum_{n=1}^{10} \left(\hat{y}^n - (b + w \cdot x_{cp}^n) \right)^2$$

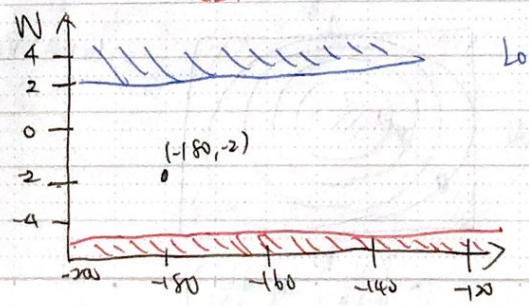
$\hat{y}^n$: 真正的數值

$(b + w \cdot x_{cp}^n)$: 現在的 function 的輸出（預測的數值）

estimation error

$f^* = \arg\min_f L(f)$

$w^*, b^* = \arg\min_{w,b} L(w,b) = \arg\min_{w,b} \Sigma(\ )$

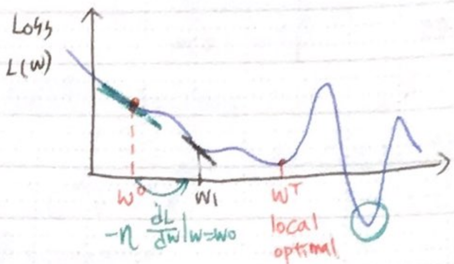


Loss 小的 function

Loss function 大 → 不好的 function

Step 3: Gradient Descent

■ Consider loss function $L(w)$ 可微



$$w^1 = w^0 - \eta \frac{dL}{dw}\Big|_{w=w^0}$$

η (eata): learning rate = 事先定好的数值

∵和 增/減 是反向的

η大 → 學習速度快

$$w^2 = w^1 - \eta \frac{dL}{dw}\Big|_{w=w^1}$$

- 找一个初始的点 (Random): w^0
- Compute $\frac{dL}{dw}\big|_{w=w^0}$ If < 0 → Increase w^0; > 0 → Decrease w^0

If $\frac{dL}{dw}\big|_{w=w^0}$ 越大 → in 越陡峭的地方 (移动的距离越大)

" 小 (" 小)

※在 linear regression：没有 local minimum

■ two parameters $w^*, b^* = \arg\min_{w,b} L(w,b)$

- 找一个初始值 w^0, b^0
- Compute $\frac{\partial L}{\partial w}\big|_{w=w^0, b=b^0}$, $\frac{\partial L}{\partial b}\big|_{w=w^0, b=b^0}$

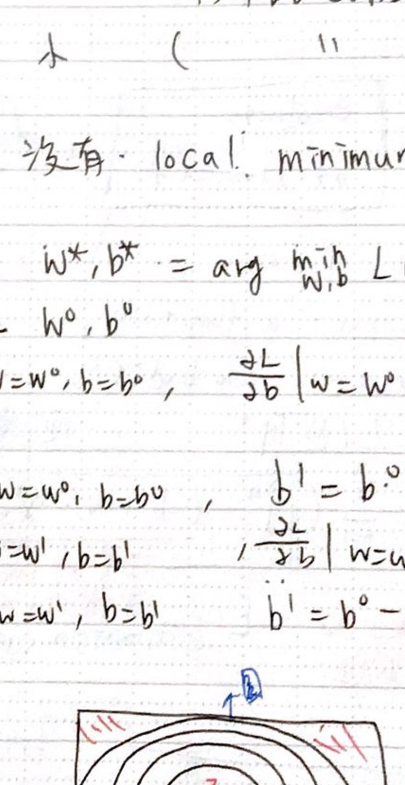
$w^1 = w^0 - \eta \frac{\partial L}{\partial w}\big|_{w=w^0, b=b^0}$, $b^1 = b^0 - \eta \frac{\partial L}{\partial b}\big|_{w=w^0, b=b^0}$

- Compute $\frac{\partial L}{\partial w}\big|_{w=w^1, b=b^1}$, $\frac{\partial L}{\partial b}\big|_{w=w^1, b=b^1}$

$w^2 = w^1 - \eta \frac{\partial L}{\partial w}\big|_{w=w^1, b=b^1}$ $b^1 = b^0 - \eta \frac{\partial L}{\partial b}\big|_{w=w^0, b=b^0}$

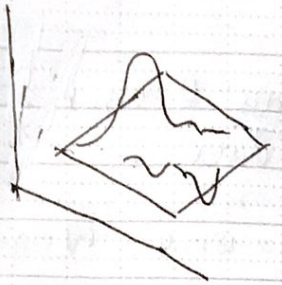
$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix}$$

↓ gradient



Color = Value of Loss $L(w,b)$: 藍 小, 紅 大

Date



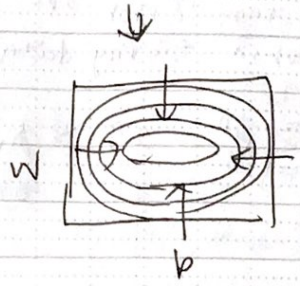
In linear regression.

the loss function L is convex.

※ Convex (是二次)

→ 沒有 local 的 optimal. 的位置

↳ 因為函數是二次



$$L(w,b) = \sum_{n=1}^{10} \left(\hat{y}^n - (b + w \cdot x_{cp}^n) \right)^2$$

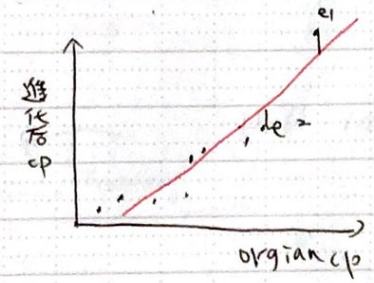
$$\frac{\partial L}{\partial w} = \sum_{n=1}^{10} 2 \cdot (-x_{cp}^n) \left(\hat{y}^n - (b + w \cdot x_{op}^n) \right)$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^{10} 2 \cdot (-1) \left(\hat{y}^n - (b + w \cdot x_{cp}^n) \right)$$

How the results?

$b = -188.4$

$w = 2.7$



Average error on Training Data

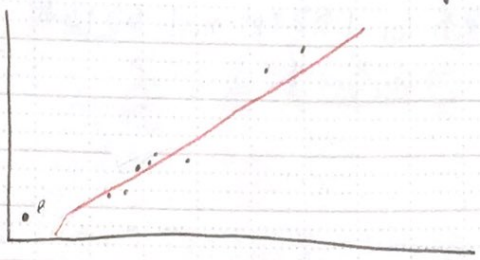
$$= \sum_{n=1}^{10} e^n = 31.9$$

Generalization

(真正关心的是，沒有看過的新的 data 其 error 是多少)

testing data

- Another 10 pokemon as testing data.



$b = -188.4 \quad w = 2.7$

Average Error on testing data

$= \sum_{n=1}^{10} e^n = 35$ > Average Error on Training D

How can we be better

→ Selecting another Model

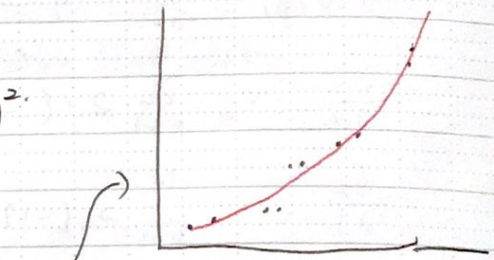
↓

$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$

Best function.

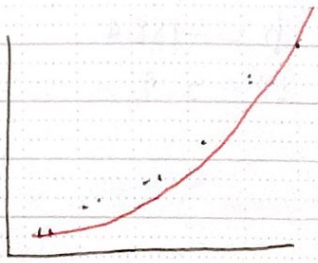
$b = -10.3 \quad w_1 = 1 \quad w_2 = 2.7 \times 10^{-3}$

Average Error = 15.4



Testing

Average Error 18.4



再更好的 model!

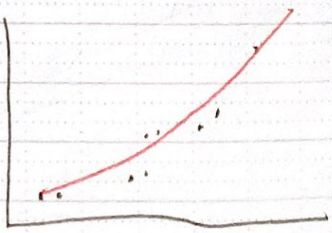
$y = b + w_1 \cdot x_{cp} + w_2 (x_{cp})^2 + w_3 \cdot (x_{cp})^3$

Best Function

$b = 6.4 \quad w_1 = 0.66 \quad w_2 = 4.3 \times 10^{-3}$

$w_3 = -1.8 \times 10^{-6}$

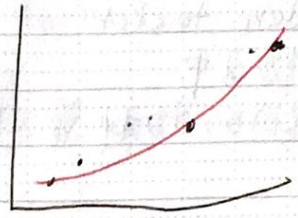
Average Error = 15.3



Date

Testing

Average Error 18.4

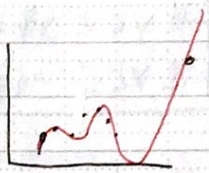


⇒ Slightly better.

$y = b + w_1 \cdot x_{cp} + \cdots + w_5 (x_{cp})^5$

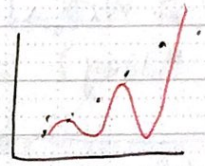
Best Function

Avenge Error = 128

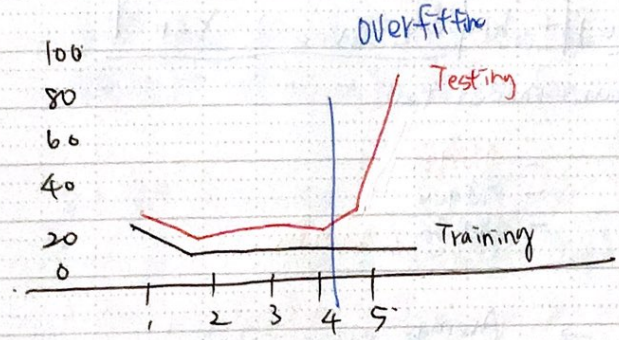


Testing

Average Error = 232.1



Model Selection



	Training	Testing
1	31.9	35
2	15.4	18.4
3	15.3	18.1
4	14.9	28.2
5	12.8	232.1

⇒ Overfitting：更複雜的model 並沒有在 testing data 有更好的表現

(在 Best function Good but 在 testing data bad)

hidden factor

⇒ 物种

→ 不同的物种算不同的 linear function

X_s = species of X

If X_s = 波波	$y = b_1 + w_1 X_{cp}$
If X_s = 獨角獸	$y = b_2 + w_2 X_{cp}$
If X_s = 綠毛蟲	$y = b_3 + w_3 X_{cp}$
If X_s = 伊布	$y = b_4 + w_4 X_{cp}$

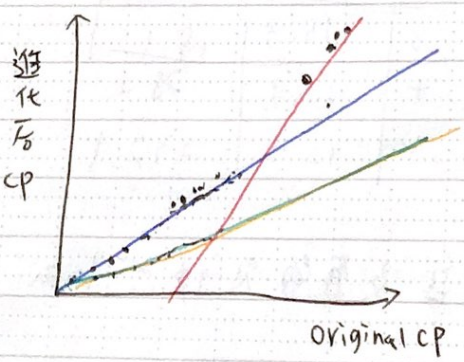
$X \rightarrow$ ⇒ y

⇒ $y = b + \sum w_i x_i$

$\delta(X_s = \text{Pidgey}) = \begin{cases} 1 & \text{if } X_s = \text{Pidgey} \\ 0 & \text{o.w.} \end{cases}$

⇒ $y = b_1 \cdot \delta(X_s = \text{Pidgey}) + w_1 \cdot \delta(X_s = \text{Pidgey})\, X_{cp}$
$+ b_2 \cdot \delta(X_s = \text{Weedle}) + w_2\, \delta(X_s = \text{Weedle})\, X_{cp}$
$+ b_3 \cdot \delta(X_s = \text{Caterpie}) + w_3\, \delta(X_s = \text{Caterpie})\, X_{cp}$
$+ b_4 \cdot \delta(X_s = \text{Eevee}) + w_4\, \delta(X_s = \text{Eevee})\, X_{cp}$

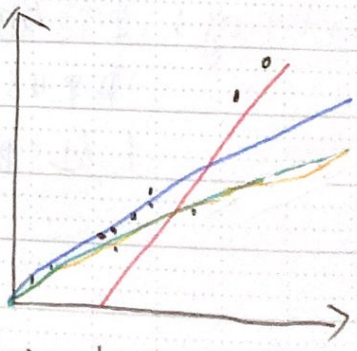
→ 可 linear model too



- 伊布
- Pidgey
- 綠毛蟲 / 獨角獸

→ Average error = 3.8

⇒ training data fit 的更好 by 不同屬性



Average error = 14.3

⇒ testing data

Date

Are there any other hidden factors?

→ Weight? Height? HP?

Back to ~~the~~ step 1. (Redesign the model again)

$X \rightarrow$

If x_s = Pidgey: $y' = b_1 + w_1 x_{cp} + w_5 (x_{cp})^2$

x_s = Weedle: $y' = b_2 + w_2 x_{cp} + w_6 (x_{cp})^2$

x_s = Caterpie: $y' = b_3 + w_3 x_{cp} + w_7 (x_{cp})^2$

x_s = Eevee: $y' = b_4 + w_4 x_{cp} + w_8 (x_{cp})^2$

$\rightarrow y$

$$y = y' + w_9 x_{hp} + w_{10} (x_{hp})^2 + w_{11} x_h + w_{12} (x_h)^2 + w_{13} x_w + w_{14} (x_w)^2$$

Training error = 1.9

Testing error = 102.3

⇒ Overfitting ⇒ (now) Regularization

Back to step 2 = Regularization

⇒ 重新 redefine 我們的 loss function，把一些 Knowledge 放入讓我們找到比較好的 function.

$$y = b + \sum w_i x_i$$

$$L = \sum_n (\hat{y}^n - (b + \sum x_i w_i))^2 + \lambda \sum (w_i)^2$$

(The function with smaller w_i is better)

⇒ 期待參數越接近 0 的 function.

∵ function 較平滑

→ 對 output 對輸入的變化是比較不敏感的

If $y = b + \sum w_i x_i$

now $y + w_i \Delta x_i = b + \sum w_i (x_i + \Delta x_i)$

(if it 愈接近 0，變化輸入)

→ if now have 雜訊，the effect 不太會受影響

Something to know

λ	Training	Testing
0	1.9	102.3
1	2.3	68.7
10	3.5	25.7
100	4.1	11.1
1000	5.6	12.8
10000	6.3	18.7
100000	8.5	26.8

λ 越大 $\quad \lambda \sum (w_i)^2$

⇒ smooth 的那个 regularization 那一项的影響力越大

⇒ 找到的 function 越平滑

⇒ training data 上的 error 越大

⇒ 越傾向考慮 w 本來的值而減少考慮 error

⇒ testing data 上的 error 可能較小

Hence we prefer smooth function

but don't be too smooth (ex 水平線)

How smooth ?

→ Select λ obtaining the "best" model

Ans: $\lambda = 100$

※調整 bias (b) 的大小和 function 的平滑程度 无关

∵ only 上下 move

Conclusion

- 原本的 cp 值物种几乎決定進化后的 cp 值 (but 仍有 hidden factors)
- Gradient descent

Next lecture: What does the error come from ?

→ validation

Date 2021 / 3 / 24

Where does the error come from?

error due to
- bias
- variance

⇒ then can improve your model

Estimator

$\hat{y} = \hat{f}$ (pokemon)

→ only the company know $\hat{f}$.

→ from the training data. we can find f^*
(f^* is the estimator of $\hat{f}$)

● for example

(1) estimator the mean of variable x

→ Assume the mean $= \mu$, variance $= \sigma^2$.

(2) estimator of mean μ.

→ sample N points : $\{x^1, x^2, \cdots x^N\}$

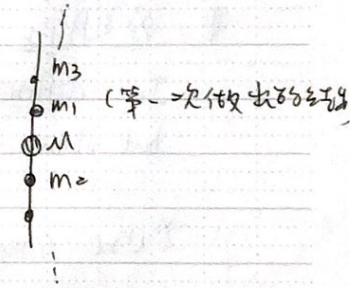
We know that $m = \frac{1}{N}\sum_{n=1}^{N} x^n \neq \mu$

but $E(m) = E\left[\frac{1}{N}\sum_{n=1}^{N} x^n\right]$

$= \frac{1}{N} E\left(\sum_{n=1}^{N} x^n\right]$

$= \mu$

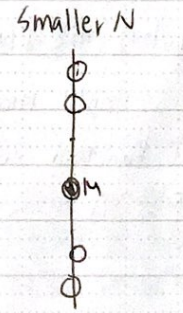
⇒ unbiased (不偏估計)

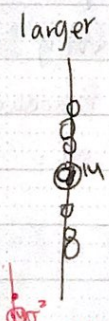


$Var(m) = \frac{\sigma^2}{N}$

↑ 分散的程度

⇒ N 越大會越集中

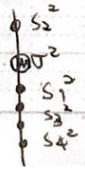




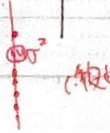
$S^2 = \frac{1}{N}\sum_{n}(x^n - m_1)^2$

$E(S^2) = \frac{N-1}{N}\sigma^2 \neq \sigma^2$

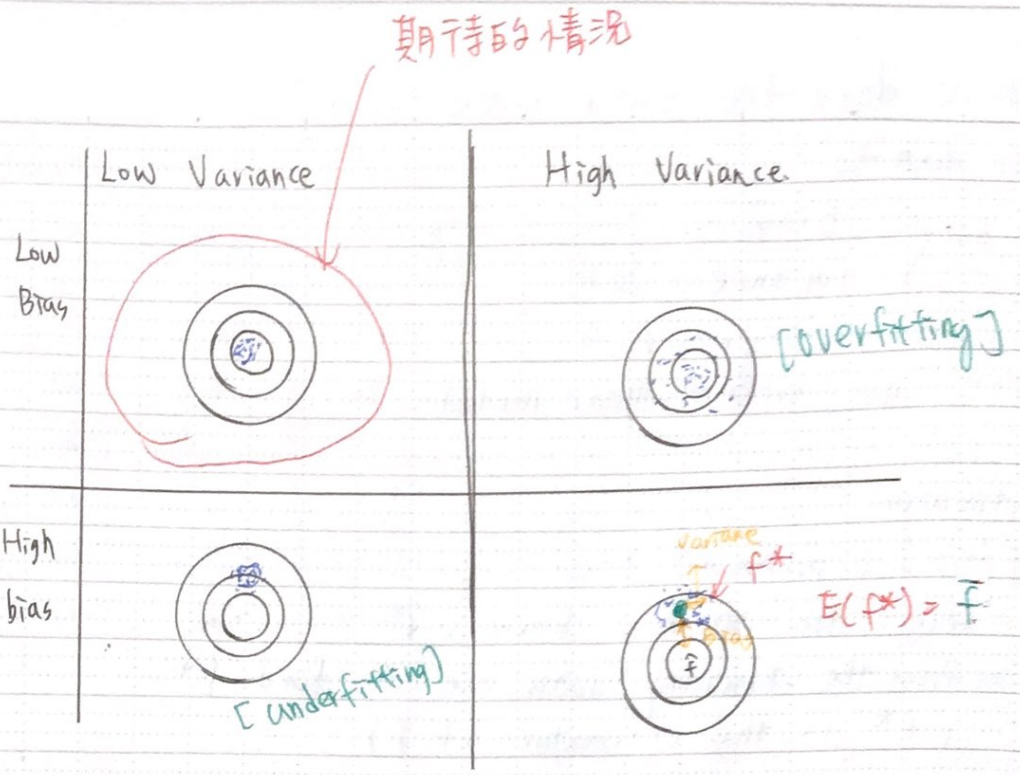
(S^2 is smaller than σ^2)



⇒ Increas N



(彼此間差距越小)



error 取決兩件事

1、 瞄準的位置在哪裡 [bias 有多大]

→ estimator 是不是 bias （有沒有瞄準）

2、 瞄準的這個位置，子彈射出后仍有偏移 [variance 有多大]

→ f^* 和平中間的距離 = variance

平行宇宙 （做很多次實驗）

In different universes, we use same model.
but obtain different f^*

※ Use simple model, variance is small

↓

Why? why 複雜性 (complex) model, variance 回 bigger.?

→ Because simple model 受到不同 data 的 impact 是小的

eg: 整个 function set. 某个一个 function $f(x) = c$

，不管抓 n 次，model 都 same, variance = 0

Date

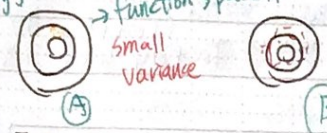
Bias

↳ If we average all f^*, is it close to $\hat{f}$.

$E(f^*) = \bar{f}$

Large Bias　　small Bias

[underfitting] → function space 小



[overfitting]

Large variance

function space 大 = 有包含 target

不同 function 間的 bias 有多大？

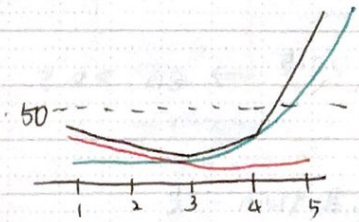
⇒ we can't evaluate it, because we don't known $\hat{f}$

⇒ 假設 $\hat{f}$

- $\bar{f}$
- $\hat{f}$
- 5000 次 f^*

※ 比較 simple 的 model : bias 大 [A]

complex 〃 : 〃 小 [B]

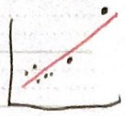


error from variance.

error from bias.

future work. 要 improve model 要走什麼.

• how is bias 大？ variance 大？



bias 大 : model 沒有辦法 fit 你的 training 的 examples

Variance 大 : you can fit the training data., but in testing data. have large error.

↳ need redesign your model ∵ model 沒有包含 target

↳ add more feature. in input ② more complex model

① More data. (generate 假的 data ex 手寫辨識 角度轉15度，水平圖与左右顛倒語言，use變聲器男→女

② Regularization ⇒ 曲線会更平滑

Model selection

- 在 bias & variance 之間做一些 trade-off
- select a model, its variance is small, bias is small
  ⇒ 合起來給我們最小的 testing data 的 error
- We should (Not) do

Training set → Testing set (your) public set → Testing set (real) private set

- model 1 → Err = 0.9
- model 2 → Err = 0.7 —
- model 3 → Err = 0.5 → Err > 0.5)

⇒ you can do is

Training set ⟨ Training set (找最好的 function f^*)
Validation set (選 model)

Training set	validation set

public Testing set | private Testing set

Model 1 → err = 0.9

Model 2 → err = 0.7

Model 3 → err = 0.5 → err > 0.5

定要用 Model 3,
但用全部的 data 在 model 3 上再 train 一次

don't recommend

→ because you will 把 public testing set 的 bias 考慮進入

→ 在 public set 上所看到的 performance 無法反應你在 private set 的 performance

※ public set 不是真正的結果

N-fold Cross Validation

Training set

Train	Train	Val
Train	Val	Train
Val	Train	Train

Model 1	Model 2	Model 3
Err = 0.2	Err = 0.4	Err = 0.4
Err = 0.4	Err = 0.5	Err = 0.3
Err = 0.3	Err = 0.6	Err = 0.3
Avg Err = 0.3	Avg Err = 0.5	Avg Err = 0.4