Date . . NO.

深度學習

目錄：

*：自行閱讀講義&觀看OCW影片

Date NO.

Ch1 Introduction to ML & DL

一. What is ML?

1. Data 的表示方法:

一个 dataset 可表为 $X=\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, 其中 $x^{(i)} \in R^D$, $y^{(i)} \in R^K$

$\begin{cases} x^{(i)}: 称为\ data \\ y^{(i)}: 称为\ label \end{cases}$

也有「無 label 之 dataset」 $X=\{x^{(i)}\}_{i=1}^{N}$

ex. MNIST dataset (手寫数字辨識)

$\begin{cases} x^{(1)} = \boxed{6} \in R^{32\times32} \\ y^{(1)} = e^{(6)} = (0,0,0,0,0,1,0,0,0,0) \end{cases}$

$\vdots$

2. Learning 的种類:

(1) Supervised: dataset 有 label ⇒ 切割 dataset, 一部份用來 learn, 一部份拿來 predict

(2) Unsupervised → 只有 $\{x^{(i)}\}_{i=1}^{N}$, learn X 之 pattern

(3) Reinforcement → learn from "good/bad" feedback (來自外界/人類/其他模型)

3. ML 的步驟

(1) collect data, 清資料, pre-processing, data exploration (最麻煩的一步就是(1))

⇒ 得 $X=\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ or $\{(x^{(i)})\}_{i=1}^{N}$

(2) 选擇一个由 w parametrized 之 model $\{f(\cdot;w)\}$ 与一个 cost function $C(w;X)$

⇒ $C(w;X)$ 之物理意義為「f 可以多好的解釋 training data」

(3) Training ⇒ 找一組 w^* s.t.

$$w^* = \arg\min_w C(w;X)$$

(4) Testing ⇒ 評估「經 training 後的 f」即 f^* 表現多好

(5) 拿 f^* 去解決實際生活的問題

* 狹義地說，「套模仔」指的是只會(5)的人。
廣義來說，在(2)的時候若是用別人開發的 f，也是「套模仔」。但實務表明，其實好的訓練 data 和好的訓練方法才是最重要的，因此我們更應關注 data 之品質和新的 training 方法

二. What is Deed Learning

Def. A deep learning model is a multi-layer (deep layer) ML model $f(\cdot;w)$ represent by

$$\hat{y} = f^{(L)}(\cdots f^{(2)}(f^{(1)}(x;w^{(1)});w^{(2)})\cdots;w^{(L)})$$

or by the block diagram

$$x \to \boxed{f^{(1)}(\cdot;w^{(1)})} \to \boxed{f^{(2)}(\cdot;w^{(2)})} \to \cdots \to \boxed{f^{(L)}(\cdot;w^{(L)})} \to \hat{y}$$

1. 優點：〈 多會自己 pre-processing data
可多會數段雜之 pattern ex. visual objects

2. 缺點：〈 需大量 data → 成本高！ (#, time)
computation cost → (#, time) 成本高！

→ 買 GPU
time → 可以跟教授說「我的 model 還在跑」

Ch5 Learning Theory & Regularization

DATE

一. Learning Theory.

Def. Learning (Mitchell, 1997)
一个程式在執行任務 T 時，若其表現(以 P 測量)隨經驗 E 改善，則称此程式会從 E 中学習

T → supervised, unsupervised, RL 等項目
P → accuracy, error rate, MSE, ...
E → dataset (X, y) or (X).

給定 training set of size N, f 為 data 中可能的一种对应关係，定義 empirical error/risk on N sample 為：

$$C_N(w) \text{ or } C_N[f] = \frac{1}{N}\sum_{i=1}^{N} loss\left(f(x^{(i)}; w), y^{(i)}\right)$$

令 $P(x, y)$ 為 data 实際之 pdf, 定義 generalization error / risk 為 (可由 test error 判断!)

$$C(w) \text{ or } C[f] = \int loss(f(x; w), y)\, dP(x, y)$$

⇒ C or C_N 本質是 loss function 之期望值，物理意義是 f 有多麼 generalize

令 $\mathbb{F}$ 為一泛函空間 (可以是我們挑的)，我們想找 $f^* = \arg\min_{f\in\mathbb{F}} C[f]$，不过顯然我們只能藉学習方法找出

$$f_N = \arg\min_{f\in\mathbb{F}} C_N[f]$$

Learning Theory 最主要的問題之一就是：「How to characterize $C[f_N] = \int loss(f_N(x; w), y)\, dP(x, y)$」？即，learn 到的 function f_N 有多麼 generalize?

Q: 若函数 f 之 $C_N[f]$ 值很小，則 $C[f]$ 亦很小嗎？
A: No!

NO.

DATE

Q: ML一定找的到 f^* 嗎?
A: NO!

Thm. No Free Lunch Thm
Averaged over all possible data generating distribution, every classification algorithm has the same error rate when classifying unseen points.

→ 沒有一种 ML 演算法比其他演算法更能在任意的task表現更好
→ 我們更在乎「对生成data之pmf $P(x,y)$而言，哪个演算法生成了更好的函数 f_N」?

至於如何討論 $C[f_N]$, 有 Bounding Method 与 Decomposition method 兩种.

二. Bounding Method

Def. Bayes error: $\min_f C[f] \triangleq C[f^*]$
$\Rightarrow C[f^*] > 0$ if there is randomness in $P(y|x)$

ex. 考慮 $y = \sin x + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$. 取 $\mathcal{F} = P_{10}$ (10次及以下多項式形成之空間)
$\Rightarrow$ 如何找 $f_N \in \mathcal{F}$ s.t. $C[f_N]$ 夠靠近 $C[f^*]$?

Def. Bounding method: 定義 (ε_{app}, ε_{est})
$\varepsilon = C[f_N] - C[f^*] = C[f_{\mathcal{F}}^*] - C[f^*] + C[f_N] - C[f_{\mathcal{F}}^*]$
其中 $f_{\mathcal{F}}^*$ 為 $\mathcal{F}$ 中最好的 model.
② ε_{app}: approximation error, 物理意義為「model中最好的function跟實際上真正的function之差」
③ ε_{est}: estimation error: 「從data中learn到的function和 model中最好的function之差」

NO.

DATE

Q: How to reduce $\mathcal{E}_{app}$?

A: 顯然可以藉由挑一個更複雜的 $\mathbb{F}$ 減少之

Q: How to reduce $\mathcal{E}_{est}$?

A: ① Simpler model (Complexity of $\mathbb{F}\downarrow$)

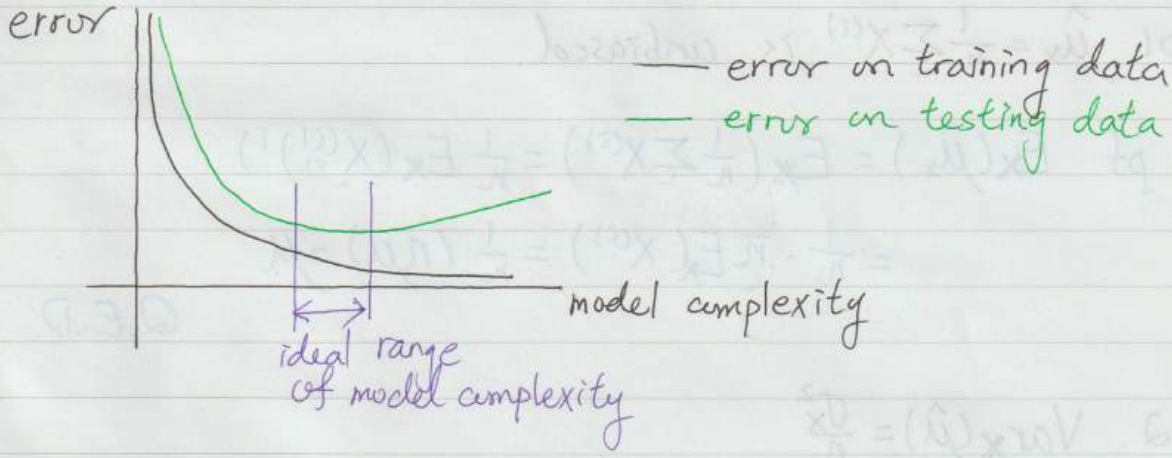
② Larger training set.

⇒ 當 model 太簡單 ⇒ high $\mathcal{E}_{app}$ ⇒ underfitting

high training error; high testing error

" 複雜 ⇒ high $\mathcal{E}_{est}$ ⇒ overfitting

low training error; high testing error



三. Decomposition Methods:

1. 前言: No (weak) assumption on data distribution is made ⇒ these bounds are too loose to quantify $C[f_N]$.

⇒ decompose $C[f_N]$ into multiple meaningful terms

假設: 〈特定之 loss function
data generating distribution $P(x,y)$ (特定)

(才能透過數學運算完成 decomposition.

⇒ 需要統計學中 point estimation 的知識

NO.

DATE

2. Point estimation.

Def. 設 dataset $\mathbb{X} = \{X^{(1)}, \dots, X^{(n)}\}$ 為 n 了 i.i.d. 樣本 of r.v. X. $X \sim$ 以 θ 為母數之分佈，則 a point estimator or statistic is $\hat{\theta}_n \triangleq g(X^{(1)}, \dots, X^{(n)})$ (function of the data). $\hat{\theta}_n$ is called the estimate of θ.

Def. The bias of an estimator $\text{bias}(\hat{\theta}_n) = E_{\mathbb{X}}(\hat{\theta}_n) - \theta$

The variance " $\text{Var}_{\mathbb{X}}(\hat{\theta}_n) = E_{\mathbb{X}}[(\hat{\theta}_n - E_{\mathbb{X}}[\hat{\theta}_n])^2]$

We say the estimator is unbiased if $\theta = E_{\mathbb{X}}(\hat{\theta}_n)$

Prop1. $\hat{\mu}_{\mathbb{X}} = \frac{1}{n}\sum_i X^{(i)}$ is unbiased.

pf. $E_{\mathbb{X}}(\hat{\mu}_{\mathbb{X}}) = E_{\mathbb{X}}\left(\frac{1}{n}\sum_i X^{(i)}\right) = \frac{1}{n} E_{\mathbb{X}}\left(\sum_i X^{(i)}\right)$

$= \frac{1}{n} \cdot \sum_i E_{\mathbb{X}}(X^{(i)}) = \frac{1}{n}(n\mu) = \mu$

Q.E.D.

Prop2. $\text{Var}_{\mathbb{X}}(\hat{\mu}) = \frac{\sigma_X^2}{n}$

pf. $\text{Var}(\hat{\mu}) = E_{\mathbb{X}}[(\hat{\mu} - E_{\mathbb{X}}[\mu])^2]$

$= E[\hat{\mu}^2 - 2\hat{\mu}\mu + \mu^2]$

$= E[\hat{\mu}^2] - \mu^2$

$= E\left[\left(E[\frac{1}{n}\sum_i X^{(i)}]\right)^2\right] - \mu^2$

$= E\left[\frac{1}{n^2}\sum_i \sum_j X^{(i)} X^{(j)}\right] - \mu^2$

$= \frac{1}{n^2}\left(\sum_{i=j} E[X^{(i)} X^{(j)}] + \sum_{i \neq j} E[X^{(i)} X^{(j)}]\right) - \mu^2$

$= \frac{1}{n^2}\left(\sum_i E[X^{(i)2}] + n(n-1) E[X^{(i)}] E[X^{(j)}]\right) - \mu^2$

$= \frac{1}{n} E[X^2] + \frac{(n-1)}{n}\mu^2 - \mu^2$

$= \frac{1}{n}(E[X^2] - \mu^2)$

$= \frac{1}{n}\sigma_X^2$

Q.E.D.

Prop3. $\hat{\sigma}_X = \frac{1}{n-1}\sum_i (X^{(i)} - \hat{\mu}_X)^2$ is the unbiased estimator

pf. $E_X[\hat{\sigma}] = E[\frac{1}{n}\sum_i (X^{(i)} - \hat{\mu})^2]$

$= E[\frac{1}{n}(\sum_i X^{(i)2} - 2\sum_i X^{(i)}\hat{\mu} + \sum_i \hat{\mu}^2)]$

$= E[\frac{1}{n}(\sum_i X^{(i)2} - n\hat{\mu}^2)]$

$= E[X^2] - E[\hat{\mu}^2]$

$= E[(X-\mu)^2 + 2X\mu - \mu^2] - E[\hat{\mu}^2]$

$= (\sigma^2 + \mu^2) - (Var[\hat{\mu}] + E[\hat{\mu}]^2)$

$= \sigma^2 + \mu^2 - \frac{1}{n}\sigma^2 - \mu^2$

$= \frac{n-1}{n}\sigma^2$

$\Rightarrow \frac{1}{n-1}\sum_i (X^{(i)} - \hat{\mu}_X)^2$ is an unbiased estimator of σ_X.

Q.E.D.

Prop4. Mean square error $E[(\hat{\theta}_n - \theta)^2] = Var_X(\hat{\theta}_n) + bias(\hat{\theta}_n)^2$

pf. $E_X[(\hat{\theta}_n - \theta)^2] = E[(\hat{\theta}_n - E[\hat{\theta}_n] + E[\hat{\theta}_n] - \theta)^2]$

$= E[(\hat{\theta}_n - E[\hat{\theta}_n])^2 + (E[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - E[\hat{\theta}_n])(E[\hat{\theta}_n] - \theta)]$

$= E[(\hat{\theta}_n - E[\hat{\theta}_n])^2] + E[(E[\hat{\theta}_n] - \theta)^2] + 2E[(\hat{\theta}_n - E[\hat{\theta}_n])(E[\hat{\theta}_n] - \theta)]$

$= E[(\hat{\theta}_n - E[\hat{\theta}_n])^2] + (E[\hat{\theta}_n] - \theta)^2 + 2 \cdot 0 \cdot (E[\hat{\theta}_n] - \theta)$

$= Var_X(\hat{\theta}_n) + bias(\hat{\theta}_n)^2$

Q.E.D.

NO.

DATE

3. Consistency

Def. An estimator is (weak consistent) $\Leftrightarrow$
$\lim_{n\to\infty} \hat{\theta}_n$ converges in probability to θ, or $\forall \varepsilon > 0$,

$$\lim_{n\to\infty} P(|\hat{\theta}_n - \theta| \geq \varepsilon) = 0 \quad (\text{or } \lim_{n\to\infty} P(|\hat{\theta}_n - \theta| < \varepsilon) = 1)$$

Thm. Weak law of Large number
The sample mean $\hat{\mu}_X = \frac{1}{n}\sum X^{(i)}$ is a consistent estimator of μ_X. i.e.

$$\lim_{n\to\infty} P(|\hat{\mu}_{X,n} - \mu_X| < \varepsilon) = 1$$

Thm. Strong law of large number
$P(\lim_{n\to\infty} \hat{\mu}_{X,n} = \mu_X) = 1$. i.e. $\hat{\mu}_X$ is a strong consistent estimator.

4. Decomposing generalization error

現考慮 $f_N = \arg\min_{f\in\mathbb{F}} C_N[f]$，我們可以分解 $C[f_N]$ 之期望值，

取 loss function 為 MSE, true label $y = f^*(x) + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$. 首先，由

$$\begin{aligned} E_X[C[f_N]] &= E_X\left[\int \text{loss}(f_N(x) - y)\, dP(x,y)\right] \\ &= E_{X,x,y}[\text{loss}(f_N(x) - y)] \\ &= E_X\left[E_{X,y}[\text{loss}(f_N(x) - y) \mid X = x]\right] \quad \text{—(*)} \end{aligned}$$

代入 MSE 知 $\mu = 0$，可得以下結果：

NO.

DATE

Thm. 在以上之假設中, $E_{x,y}[loss(f_N(x)-y)|x=x] = \sigma^2 + Var_x[f_N(x)|x] + bias[f_N(x)|x]^2$

pf. $E_{x,y}[loss(f_N(x)-y)|x] = E_{x,y}[(f_N(x)-y)^2|x]$

$= E_{x,y}[y^2 + f_N(x)^2 - 2f_N(x)y|x]$

$= E_y[y^2|x] + E_x[f_N(x)^2|x] - 2E_{x,y}[f_N(x)y|x]$

$= (Var_y[y|x] + E_y[y|x]^2) + (Var_x[f_N(x)|x] + E_x[f_N(x)|x]^2) - 2E_y[y|x]E_x[f_N(x)|x]$

$= Var_y[y|x] + Var_x[f_N(x)|x] + (E_x(f_N(x)|x) - E_y[y|x])^2$

$= Var_y[y|x] + Var_x[f_N(x)|x] + E_x[f_N(x) - f^*(x)|x]^2$

$= \sigma^2 + Var_x[f_N(x)|x] + bias[f_N(x)|x]^2$

Q.E.D.

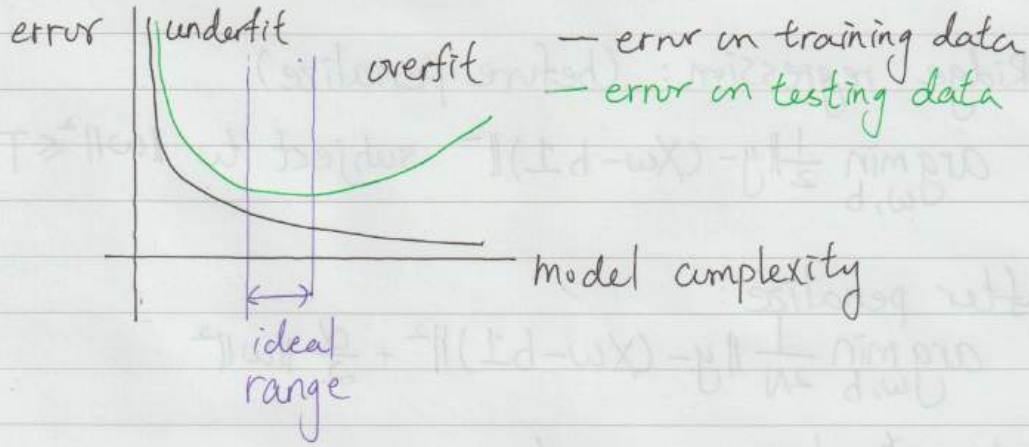
Cor. $E_x[C[f_N]] = E_x(E_{x,y}[loss(f_N(x)-y)|x])$

$= E_x(\sigma^2 + Var_x[f_N(x)|x] + bias[f_N(x)|x]^2)$

(1) σ^2 → 無法 avoid !! ∵ 源自 data 自身 randomness

(2) trade off between variance & bias ! (圖例: PPT p27)

* 結論: 到目前為止, 我們討論了無論要用哪种演算法 train model, 都要想好此 model f_N 会有哪些 trade off 以及如何挑出最「適当」的 model



NO.

DATE

四. Regularization

目前我們都是由 generalization error

$$C[f_N] = \int loss(f_N(x;\omega), y)\, dP(x,y)$$

探討 f_N 有多 generalize. 不過實際上我們只能由最小化 empirical error

$$C_N[f] = \frac{1}{N}\sum_i loss(f(x^{(i)};\omega), y^{(i)})$$

來 train 出 model f_N. 即 $f_N = \arg\min_{f\in F} C_N[f]$. 我們稱任何改善 f_N generalizability 的手段為 regularization

- on cost function : weight decay
- during training process: validation

Occam's razor: among equal-performing models, the simplest one should be selected.

1. Weight decay:

我們以 f_N 之參數 ω 之範數定義 f_N 之複雜程度

這是由於 $\|\omega\| = 0 \leftrightarrow \omega = 0$ 為隨机猜答案

⇒ penalize on norm of ω, 即在 cost function 後加上 $\|\omega\|$.

ex. Ridge regression: (before penalize)

$$\arg\min_{\omega,b} \frac{1}{2}\|y - (X\omega - b1)\|^2 \quad \text{subject to } \|\omega\|^2 \leq T.$$

after penalize:

$$\arg\min_{\omega,b} \frac{1}{2N}\|y - (X\omega - b1)\|^2 + \frac{\alpha}{2}\|\omega\|^2$$

$\alpha > 0$ 為 hyper parameter.

NO.

DATE

大α: prefer simple model
小α: care empirical error (低一点)

不去对 b regularize 之原因为 y 未必有标准化过.

Def. LASSO (least absolute shrinkage and selection operator) → penelize $\|w\|_1$

$$\arg\min_{w,b} \frac{1}{2N}\|y-(Xw-b1)\|^2+\alpha\|w\|_1$$

→ sparse w! 原因: $\|w\|_1$ 之顶点落在坐标轴上, $\|\cdot\|^2$ 之等位面为高维球面 ⇒ 两者交点易在坐标轴上.

Def. Elastic Net: combines Ridge and LASSO

$$\arg\min_{w,b} \frac{1}{2N}\|y-(Xw-b1)\|^2+\alpha\left(\beta\|w\|_1+\frac{1-\beta}{2}\|w\|^2\right)$$

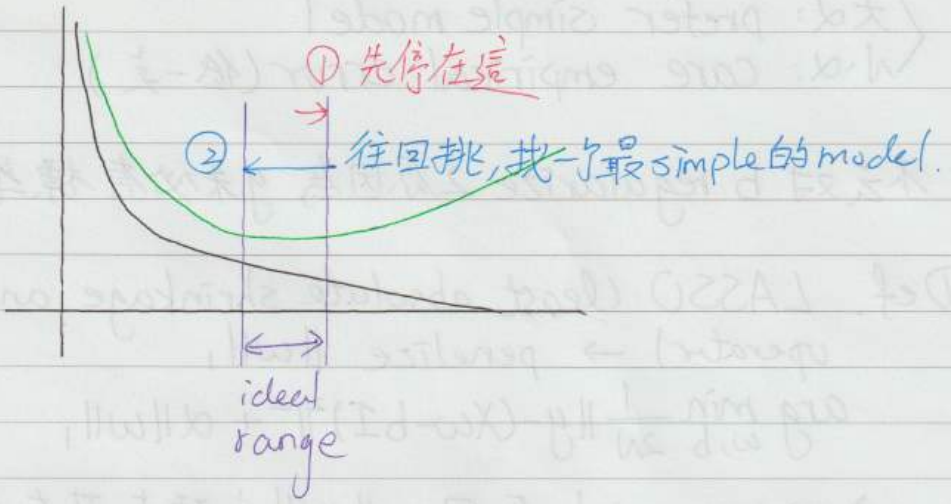
2. Validation

Def. The constants fixed in a model is called hyperparameters

ex. degree p in polynomial regression, α in LASSO...

物理意义 ⇒ 改变 hyperparameter 之值如同改变 model complexity!

⇒ 根据 Occam's razor, 我们应逐渐增加 model complexity, 直至 overfit 发生前一刻停止,



此外，「衡量是否 overfit」以及「往回挑」時，不应以 test data 測試之，否則只是讓 model 对 testing set overfit 而已.

⇒ 要從 training set 切出一个 validation set 來选 hyperparameter!

Ch6 Probablistic Models

NO.

DATE

一. Probabilistic Models

Given a dataset $\mathbb{X} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, Model $\mathbb{F}$: a collection of functions parametrized by $\textcircled{H}$

Goal: new data point x' → train a function f s.t. $\hat{y} = f(x'; \textcircled{H})$ is closet to the correct label y'

Def. Probabilistic models:

We write the above function $f(x'; \textcircled{H})$ as $P(y=y \mid X=x')$. The prediction is made by

$\hat{y} = \arg\max_{y} P(y=y \mid X=x'; \textcircled{H})$ (r.v.)

In such model, we have to find $\textcircled{H}$.

我們有兩種方法找 $\textcircled{H}$:

〈法I〉: Maximum likelihood Estimation (MLE)

設 $\mathbb{X} \sim$ 以 $\textcircled{H}$ 為母數之分佈, 則求

$\textcircled{H}_{ML} = \arg\max_{\textcircled{H}} P(\mathbb{X} \mid \textcircled{H})$

〈法II〉: Maximum Posteriori Estimation

$$\textcircled{H}_{MAP} = \arg\max_{\textcircled{H}} P(\textcircled{H} \mid \mathbb{X})$$

$$= \arg\max_{\textcircled{H}} \frac{P(\mathbb{X} \mid \textcircled{H}) P(\textcircled{H})}{P(\mathbb{X})}$$

$$= \arg\max_{\textcircled{H}} \boxed{P(\mathbb{X} \mid \textcircled{H})} \boxed{P(\textcircled{H})}$$

得一病 $\textcircled{H}$ 出現症狀 $\mathbb{X}$ 之機率?

pmf (or pdf)

ex. $\textcircled{H}$ 為病, $P(\textcircled{H})$ 表所有病分佈之pmf

二. Maximum Likelihood Estimation

NO.

DATE

Assumption (ML起手式):

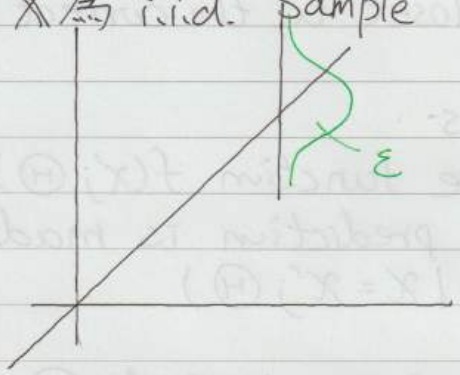
(1) $\exists$ a deterministic f^* s.t. $y = f^*(x) + \varepsilon$

(2) $\varepsilon \sim N(0, \beta^{-1})$

(3) 做線性迴歸，假設 $f^*(x; w^*) = w^{*T}x$

(4) x 已標準化

(5) X 為 i.i.d. sample



$\Rightarrow$ 目標：找出一个 w 使得 $\hat{y} = \underset{y}{\operatorname{argmax}} P(y \mid X = x; w) = w^T x$ (irr. to β)

方法：MLE of w^*：$w_{ML} = \underset{w}{\operatorname{argmax}} P(X \mid w)$

仿照統計学中的做法，

$$P(X \mid w) = \prod_{i=1}^{N} P(x^{(i)}, y^{(i)} \mid w)$$

$$= \prod_{i=1}^{N} P(y^{(i)} \mid x^{(i)}, w) P(x^{(i)} \mid w)$$

$$= \prod_{i=1}^{N} P(y^{(i)} \mid x^{(i)}, w) P(x^{(i)})$$

$$= \prod_{i=1}^{N} g(y^{(i)}; \mu = w^T x^{(i)}, \sigma^2) P(x^{(i)}) \quad (g \text{ 為 } N(w^T x^{(i)}, \sigma^2) \text{ 之 pdf})$$

$$= \prod_{i=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(y^{(i)} - w^T x^{(i)})^2\right) P(x^{(i)})$$

$$\Rightarrow \underset{w}{\operatorname{argmax}} P(X \mid w) = \underset{w}{\operatorname{argmax}} \log P(X \mid w)$$

$$= \underset{w}{\operatorname{argmax}} \log \left[\prod_{i=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(y^{(i)} - w^T x^{(i)})^2\right) P(x^{(i)}) \right]$$

NO.

DATE

$$= \arg\max_w N\sqrt{\frac{\beta}{2\pi}} - \frac{\beta}{2}\sum_i \left(y^{(i)} - w^T x^{(i)}\right)^2 + \sum_i P(x^{(i)})$$

$$= \arg\min_w \sum_i \left(y^{(i)} - w^T x^{(i)}\right)^2$$

⇒ solve analytically or by SGD.

三. MLE之应用：Logistic regression.

Assumption: $(y|x) \sim b(1, \rho)$, 即

$P(y|x;\rho) = \rho^{y'}(1-\rho)^{1-y'}$, $y' = \frac{y+1}{2} \in \{0, 1\}$

(由於是要做 binary classification, 因此y值只有0, 1兩种可能).

目標：定義 logistic function 為 $\sigma(z) = \frac{1}{1+e^{-z}}$, 且令

$z = w^T x$, 想找出一个 w s.t.

$$\hat{y} = \arg\max_y P(y|x;w) = \text{sign}(w^T x)$$

其中 $P(y|x;w) = \sigma(z)^{y'}(1-\sigma(z))^{(1-y')}$

$= \sigma(w^T x)^{y'}(1-\sigma(w^T x))^{1-y'}$

方法：MLE of w: $w_{ML} = \arg\max_w P(X|w)$

$$\Rightarrow \arg\max_w P(X|w) = \arg\max_w \log P(X|w)$$

$$= \arg\max_w \log \prod_i P(y^{(i)}|x^{(i)};w)\, P(x^{(i)}|w)$$

$$= \arg\max_w \log \prod_i \sigma(w^T x)^{y^{(i)'}}(1-\sigma(w^T x))^{1-y^{(i)'}}$$

$$= \arg\max_w \sum_i y^{(i)'} \log \sigma(w^T x) + (1-y^{(i)'}) \overset{\log}{\wedge} (1-\sigma(w^T x))$$

$$= \arg\max_w \sum_i y^{(i)'} \log \frac{1}{1+e^{-w^T x}} + (1-y^{(i)'}) \log \frac{e^{w^T x}}{1+e^{-w^T x}}$$

SEASON

NO.

DATE

$$= \arg\max_w \sum_i -y^{(i)} \log(1+e^{-w^T x}) + (1-y^{(i)})(w^T x - \log(1+e^{w^T x}))$$

$$= \arg\max_w \underbrace{\sum_i y^{(i)} w^T x^{(i)} - \log(1+e^{w^T x^{(i)}})}_{*}$$

∵ We can prove that (*) is concave of w

∴ SGD finds global optimal.

III. Maximum A Posteriori Estimation (MAP)

We solve $w_{MAP} = \arg\max_w P(w|X) = \arg\max_w P(X|w)P(w)$

pdf → 给某些 w 权重

Let's apply MAP on linear regression. We assume

1. $w \sim N(0, \beta^{-1}I)$

$$\Rightarrow \arg\max_w P(w|X) = \arg\max_w \log P(w|X)$$

$$= \arg\max_w \log P(X|w)P(w)$$

$$= \arg\max_w \log P(X|w) + \log P(w)$$

by MLE

$$= \arg\max_w \sum_i (y^{(i)} - w^T x^{(i)})^2 + \log \sqrt{\frac{1}{(2\pi)^D \det(\beta^{-1}I)}} \exp\left(-\frac{1}{2} w^T (\beta^{-1}I)^{-1} w\right)$$

$$= \arg\max_w \sum_i (y^{(i)} - w^T x^{(i)})^2 - \beta w^T w$$

$$= \arg\max_w \sum_i (y^{(i)} - w^T x^{(i)})^2 - \beta \|w\|$$

⇒ Ridge regression 之 weight decay term!

2. $w \sim \text{Laplace}(0, b)$

note that Laplace distribution 的 pdf 為

$$g(w|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

$$\Rightarrow \arg\max_w P(w|X) = \arg\max_w \log(X|w) + \log P(w)$$
$$= \arg\max_w \sum_i (y^{(i)} - w^T x^{(i)})^2 + \log \frac{1}{2b} \exp\left(-\frac{|w|}{b}\right)$$
$$= \arg\max_w \sum_i (y^{(i)} - w^T x^{(i)})^2 - \frac{1}{b}|w|$$

$\Rightarrow$ LASSO!

* some remarks

Thm. (Consistency)
The ML estimator $\hat{H}_{ML}$ is consistent.

Thm. (Cramér-Rao Lower bound)
At a fixed large number N of examples, no consistent estimator of H has a lower expected MSE than the ML estimator $\hat{H}_{ML}$.

Remark. data沒有很多時要用MAP $\Rightarrow$ introduce bias, reduce variance

Date . . NO.

Ch8 Cross Validation & Ensembling

— Cross Validation

Notation: X: training set
$X^{(i)}$: " 之第 i 个 fold
N: " size
$N^{(i)}$: " 之第 i 个 fold 之 size
C_{cv} cross validation error
f: model

1. CV 之用处 { Hyperparameter tuning
Performance reporting

Algorithm. K-fold CV
1. 将 X 分成 K 个等份之 fold $X^{(i)}$, $i=1,2,\dots,K$
2. For $i=1,\dots,K$, 以 $X-X^{(i)}$ train 出 $f_{-N^{(i)}}$
3. 令 $C_{cv}=\frac{1}{K}\sum_{i=1}^{K} C[f_{-N^{(i)}}]$, 其中 $C[\cdot]$ 为 test error 在 $X^{(i)}$ 上之

Algorithm. $K\times M$ nested CV.
1. 将 X 分成 K 等份之 fold $X^{(i)}$, $i=1,2,\dots,K$.
2. For $i=1,2,\dots,K$,
   (1) 以 $X^{(i)}$ 为 test fold, $X-X^{(i)}$ 为 training fold, 将 $X-X^{(i)}$ 再分为 M 个等份
   (2) 对所有 hyperparameter 的候选组合, 以 $X-X^{(i)}$ 做 M-fold CV
   (3) 以所找到最好的 hyperparameter 和整个 $X-X^{(i)}$ train 出 $C[f_{-N^{(i)}}]$
3. 令 $C_{cv}=\frac{1}{K}\sum_{i=1}^{K} C[f_{-N^{(i)}}]$ 为 test error. $C[\cdot]$ 为 $X^{(i)}$ 上之 test error.

Date . . NO.

2. 設 $X \sim$ 以θ为母数之分佈，$\hat{\theta}_n$为由n个来自X之sample估计之θ. 则定义 mean square error of $\hat{\theta}_n$ 为: $MSE(\hat{\theta}_n) = E_X[(\hat{\theta}_n - \theta)^2]$

Prop. $E_X[(\hat{\theta}_n - \theta)^2] = Var_X(\hat{\theta}_n) + (bias(\hat{\theta}_n))^2$

pf. $E_X[(\hat{\theta}_n - \theta)^2] = E[(\hat{\theta}_n - E[\hat{\theta}_n] + E[\hat{\theta}_n] - \theta)^2]$

$= E[(\hat{\theta}_n - E[\hat{\theta}_n])^2 + (E[\hat{\theta}_n - \theta)^2 + 2(\underbrace{\hat{\theta}_n - E[\hat{\theta}_n]}_{0})(E[\hat{\theta}_n] - \theta)]$

$= E[(\hat{\theta}_n - E[\hat{\theta}_n])^2] + (E[\hat{\theta}_n] - \theta)^2$

$= Var_X[\hat{\theta}_n] + bias(\hat{\theta}_n)^2$

Q.E.D

Cor. $\because$ Ccv is an estimator of $E_X[C[f_N]]$

($\because$ $C[f_{-N^{(i)}}]$ " $E_X[C[f_N]]$ as well)

$\therefore MSE(Ccv) = E_X[Ccv - E_X[C[f_N]]^2]$

$= Var_X[Ccv] + bias[Ccv]^2$

3. 物理意义：以5-Fold与10-Fold比较，

(1) $\begin{cases} Var: 10\text{-}Fold > 5\,Fold \\ bias: \text{"} \quad < \text{"} \end{cases}$

(2) MSE of an unbiased estimator is its variance

二. Fold 数之选择

Lemma. $bias[Ccv] = bias[C[f_{-N^{(s)}}]]$, $s = 1, 2, \ldots, K$

$Var_X[Ccv] = \frac{1}{K} Var[C[f_{-N^{(i)}}]] + \frac{2}{K^2} \sum_i \sum_{j>i} Cov[C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]]$

Date . . NO.

pf. $\text{bias}[C_{cv}] = E_X[C_{cv}] - E_X[C[f_N]]$

$= E\left[\sum_{i=1}^{K} \frac{1}{K} C[f_{-N^{(i)}}]\right] - E[C[f_N]]$

$= \frac{1}{K}\sum_i E[C[f_{-N^{(i)}}]] - E[C[f_N]]$

$= E[C[f_{-N^{(s)}}]] - E[C[f_N]], \forall s=1,\dots,K$

$= \text{bias}[C[f_{-N^{(s)}}]], \forall s=1,2,\dots,K$

$\text{Var}_X[C_{cv}] = \text{Var}\left[\sum_{i=1}^{K} \frac{1}{K} C[f_{-N^{(i)}}]\right]$

$= \frac{1}{K^2}\text{Var}\left[\sum_{i=1}^{K} C[f_{-N^{(i)}}]\right]$

$= \frac{1}{K^2}\left(\sum_{i=1}^{K}\text{Var}[C[f_{-N^{(i)}}]] + 2\sum_{i=1}^{K}\sum_{j=i}^{K}\text{Cov}_X[C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]]\right)$

= 所求

Q.E.D.

⇒ 可觀察到 ∵ $\text{bias}[C[f_{-N^{(i)}}]]$ 与 $\text{Var}[C[f_{-N^{(i)}}]]$ 之間必有 trade off

∴ $\text{bias}[C_{cv}]$ 与 $\text{Var}[C_{cv}]$ 亦有 trade off

但由 lemma 2, 只要使 $f_{-N^{(i)}}$ 与 $f_{-N^{(j)}}$ uncorrelated 就能使 $\text{Var}[C_{cv}]\downarrow$

1. K 很大時:

∵ $f_{-N^{(s)}}$ is trained on more examples

∴ $\text{bias}[C[f_{-N^{(s)}}]], \text{Var}[C[f_{-N^{(s)}}]]\downarrow$,

$\text{Cov}[C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]]\uparrow$ (∵ $X-X^{(i)}$, $X-X^{(j)}$ are similar)

2. K 很小時 ⇒ 恰相反

3. Leave-one-out CV (取 K=N)

⇒ 用於 dataset 非常小的時候

∵ 當 dataset 太小時, $bias[CL f_{-N}^{(i)}]$ & $Var[CL f_{-N}^{(i)}]$ dominate $MSE[C_{CV}]$

∴ 必 overfit

⇒ 取 K=N, 以減少 variance

三. Voting

Def. Voting is a linear combinition of the predictions of base-learners for each x:

$$\tilde{y}_k = \sum_j w_j \hat{y}_k^{(j)}, \quad w_j > 0, \quad \sum_j w_j = 1$$

ex. Sum rule: $\tilde{y}_k = \frac{1}{L} \sum_{j=1}^{L} \hat{y}_k^{(j)}$

minimum: $\tilde{y}_k = \min_j \hat{y}_k^{(j)}$

Q: Why Voting Works?

Prop. Assume that each $\hat{y}^{(j)}$ has the expected value $E_x[\hat{y}^{(j)}|x]$ and Variance $Var_x[\hat{y}^{(j)}|x]$ If $\hat{y}^{(i)}$ and $\hat{y}^{(j)}$ are uncorrelated, $\forall i \neq j$, then the variance can be reduced, but the bias is same

pf. Set $w_j = \frac{1}{L}$, then

$$\because E_x[\tilde{y}|x] = E\left(\sum_j \frac{1}{L} \hat{y}^{(j)}|x\right)$$

$$= \frac{1}{L} \sum_j E[\hat{y}^{(j)}|x]$$

$$= E[\hat{y}^{(j)}|x]$$

∴ The expected value does not change.

$\Rightarrow$ The bias doesn't change

$\because \mathrm{Var}_x(\tilde{y}|x) = \mathrm{Var}\left(\frac{\sum}{j}\frac{1}{L}\hat{y}^{(j)}|x\right)$

$= \frac{1}{L^2}\mathrm{Var}\left(\sum_j \hat{y}^{(j)}|x\right)$

$= \frac{1}{L}\mathrm{Var}(\hat{y}^{(j)}|x) + \frac{2}{L^2}\sum_i\sum_{j>i}\mathrm{Cov}(\hat{y}^{(i)}, \hat{y}^{(j)}|x)$

$\therefore$ { If $\hat{y}^{(i)}$ and $\hat{y}^{(j)}$ are uncorrelated, the variance can be reduced
" positively correlated, " increases.

Q.E.D.

Remark. 實際上 $\hat{y}^{(j)}$ 之間不太可能 i.i.d. 比如兄弟在 activate function

四. Bagging (bootstrap aggregating)

目標：使 base-learners 不同
$\Rightarrow$ 使用「稍微不同的」data 去 train

Algorithm. Bagging.
1. 自 X 中取入了 data，取後放回，重複 L 次，得 L 个 training set $X^{(j)}$, $j=1,\ldots,L$
2. 以 $X^{(j)}$ train L 个不同的 model

五. Boosting

1. 目標：train complementary base-learners
$\because$ 要 train uncorrelated 的 base-learner 可能要用很 unstable 的方法.

Date . . NO.

2. 基本的 Boosting 算法

⇒ train the next learner on the mistakes of the previous learners

比如：考虑 binary classification $d^{(j)}(x) \in \{1, -1\}$

Def. A weak learner has error prob. less than 1/2 (better than "random guess")

"strong": arbitrarily small error prob.

Algorithm. Boosting (binary classification)

Training: 1. 将 X 分成三份 $X^{(1)}, X^{(2)}, X^{(3)}$

2. 用 $X^{(1)}$ train 出 $d^{(1)}$，再拿 $d^{(1)}$ 去 predict $X^{(2)}$

3. 以 $X^{(2)}$ 中 $d^{(1)}$ 分错的点去 train $d^{(2)}$

4. 拿 $d^{(1)}, d^{(2)}$ 去 predict $X^{(3)}$

5. 用 $X^{(3)}$ 中 $d^{(1)}, d^{(2)}$ predict 出来不同的点去 train $d^{(3)}$

Testing. 1. 以 $d^{(1)}, d^{(2)}$ 判别一个 point

2. 若 $d^{(1)}(x) = d^{(2)}(x)$，则以此为 prediction；反之，则以 $d^{(3)}(x)$ 为 prediction.

3. AdaBoost

Notation: $P_r^{(i,j)} = P((x^{(i)}, y^{(i)})$ is drawn to train the j^{th} base-learner $d^{(j)})$

$\varepsilon^{(j)}$ = error rate of $d^{(j)}$ on its training set

$\triangleq \sum_i P_r^{(i,j)} \cdot 1(y^{(i)} \neq d^{(j)}(x^{(i)}))$

(1) 概念：uses the same training set over and over again

(2). Modify the prob. of drawing the instances as a function of error rate $\varepsilon^{(j)}$.

Algorithm.

Training 1. Initialize $Pr^{(i,j)} = \frac{1}{N}$

2. For all j:

(1) 以 $Pr^{(i,j)}$ 之机率自X中抽出N个example

(2) 以此 train $d^{(j)}$

(3) 若 $\varepsilon^{(j)} \geq 1/2$, 则不再新增 base-learner

(4) 令 $\begin{cases} \alpha_j = \frac{1}{2}\log\left(\frac{1-\varepsilon^{(j)}}{\varepsilon^{(j)}}\right) > 0 \\ Pr^{(i,j+1)} = Pr^{(i,j)} \cdot \exp(-\alpha_j y^{(i)} d^{(j)}(x^{(i)})), \forall i \end{cases}$

(5) Normalize $Pr^{(i,j+1)}$, $\forall i$

$$Pr^{(i,j+1)} \leftarrow \left(\sum_i Pr^{(i,j+1)}\right)^{-1}$$

Testing. 1. Given x, 算出 $\hat{y}^{(1)}, \dots \hat{y}^{(j)}, \dots, \forall j$

2. Final prediction

$$\tilde{y} = \sum_j \alpha_j d^{(j)}(x)$$

(4) Adaboost Work 的原因.

Def. ① The margin of a prediction of an example $(x^{(i)}, y^{(i)}) \in X$ as

$$\text{margin}(x^{(i)}, y^{(i)}) \triangleq y^{(i)} f(x^{(i)}) \quad \text{越远越0越好}$$

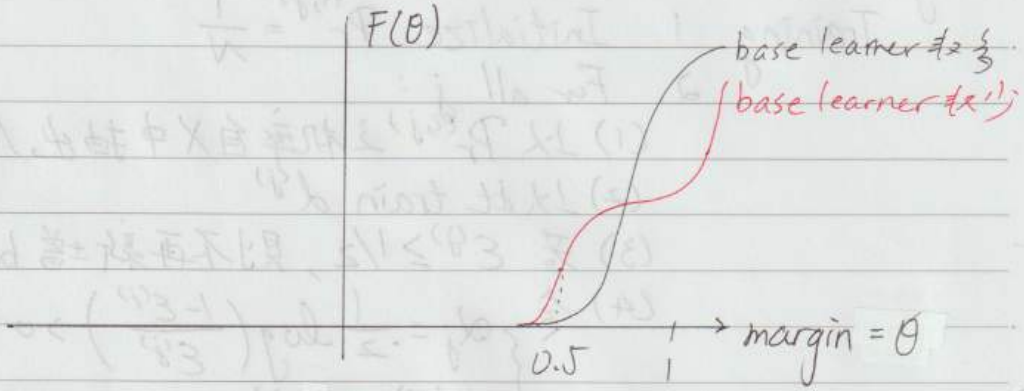
$$= \sum_{j \in \{j | y^{(i)} = d^{(j)}(x^{(i)})\}} \alpha_j - \sum_{j \in \{j | y^{(i)} \neq d^{(j)}(x^{(i)})\}} \alpha_j$$

② The confidence of $x^{(i)}$ is $f(x^{(i)})$

→ 越 decision boundary 越远, confidence 越高

則：margin之 cdf 可由

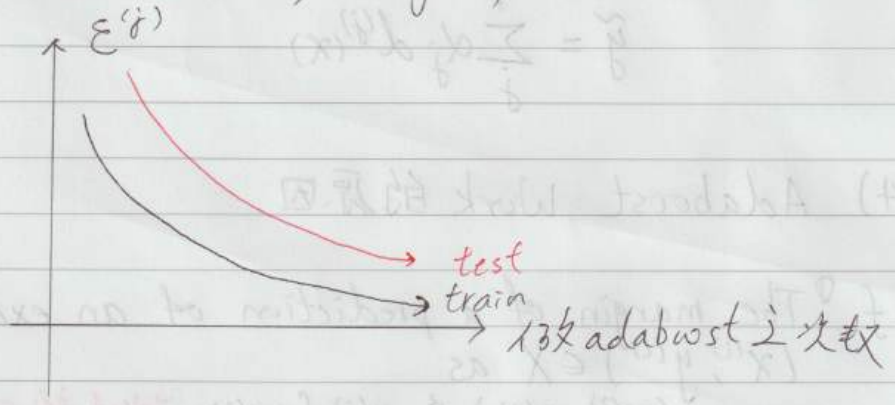
$$F(\theta) = P_X(y^{(i)} f(x^{(i)}) \le \theta) \approx \frac{|(x^{(i)}, y^{(i)}) : y^{(i)} f(x^{(i)}) \le \theta|}{|X|}$$



⇒ base learner 少，仍有一些 training data 之 margin < 0.5

" 多，幾乎沒有 "

⇒ base learner 較↑, margin ↑



⇒ 無 overfit 之問題！

Date . . NO.

Ch9 Large Scale Machine Learning

一. Big Data

1. Big Data 的特性 — 4V

(1) Volume → TB 甚至 PB

(2) Variaty → data 可能 structured or unstructured

(3) Velocity → 每一單位新增很多資料

(4) Veracity : ∵ 資料搜集自動化
∴ 用 implicit 的方法 (ex. 瀏覽器)
⇒ 可能有搜不到的情形
⇒ 每个 domain 的資料可能不是那麼健全/precise

2. Deep learning 的優勢:

(1) NN 很深時 ⇒ representation learning

Assumption. 1. Smooth assumption: If $x \approx x^{(i)} \in X$, then $f(x;w) \sim f(x^{(i)};w)$, w is the parameter of the model

2. 深層的 factor 可以 recursively 用淺層的 factor 來表示.

∵ By assumption 2 ⇒ 深的 NN 相較淺的 NN 只需要較少的 data 就能 train 出來
⇒ 消除 curse of dimension!
⇒ 其他 model 很少見!

(2) 此 factor 為 nonlinear! ⇒ 未必都一定要用 deep learning
(∵ By no free lunch thm)

⇒ When the function f to learn is
{ complex or have composite pattern ⇒ deep learning
simple or linear ⇒ other large-scale ML techniques

3. Curse of dimension. 假設 data 是散佈在 R^n 的 bin (小格子) 中
⇒ data 維數越高，則為了要 interpolate，所需的 example 數目指數增加!

二. Deep learning 的特徵

1. 可平行化:
(1) Data parallelism ⇒ 每个CPU塞一模形同時做 SGD (以 partitioned data)，再給 parameter server 去更新个 Δw

(2) model Parallelism ⇒ 每个CPU都看的到完整的 data，但只負責 train 一小部份之 model

2. model 效能受 train 的時間限制
Recall from learning theory:

Notation: f_N: The function learned from N examples X
f^*: The true function
F: The model space
f_F^*: The best model we can have in F.

Def. ① The empirical risk $C_N[f_N] = \frac{1}{N}\sum_{i=1}^{N} loss(f_N(x^{(i)}), y^{(i)})$

② The Expected risk: $C[f_N] = \int loss(f(x), y)\, dP(x, y)$
→ 無法 minimize!

③ $f_F^* \triangleq \arg\min_{f \in F} C[f]$

Date NO.

④ The excess error $\mathcal{E} = C[f_N] - C[f^*]$.

在 learning theory 中，把 $\mathcal{E}$ 拆成 approximation error 与 estimation error ($\mathcal{E}_{app}$, $\mathcal{E}_{est}$)

$$\mathcal{E} = \underbrace{C[f_F^*] - C[f^*]}_{\mathcal{E}_{app}} + \underbrace{C[f_N] - C[f_F^*]}_{\mathcal{E}_{est}}$$

⇒ 但受 training time 限制，我们 train 出来的 model 为 $\tilde{f}_N$. 则 $C_N[\tilde{f}_N] \le C_N[f_N] + \rho$.

$$\Rightarrow \mathcal{E} = \underbrace{C[f_F^*] - C[f^*]}_{\mathcal{E}_{app}} + \underbrace{C[f_N] - C[f_F^*]}_{\mathcal{E}_{est}} + \underbrace{C[\tilde{f}_N] - C[f_N]}_{\mathcal{E}_{opt}}$$

⇒ To reduce

- $\mathcal{E}_{app}$ → Chouse a larger model
- $\mathcal{E}_{est}$ → increase N or choose a smaller model
- $\mathcal{E}_{opt}$ → train 久一点 or 挑一个比较快的 optimizer.

(1) small scale ML tasks

∵ time is not an issue

∴ $\mathcal{E}_{opt}$ 可很小.

⇒ trade off between $\mathcal{E}_{app}$ and $\mathcal{E}_{est}$

(2). large scale ML tasks

∵ $\mathcal{E}_{opt}$ 要↓

∴ 用 SGD

∵ N大

∴ $\mathcal{E}_{est}$ ↓.

⇒ $\mathcal{E}_{app}$ ↓，故 prefer large model

为了使

Date . . NO.

三. NN和 Gaussian Process 之關係 ⇒ omit

這部份太複雜，有概念說很大又很宽的NN在train到很久的話会呈現 Gaussian process 就可以了.

Ch10 Neural Networks: Design

NO.

DATE

一. 什麼是神經網路

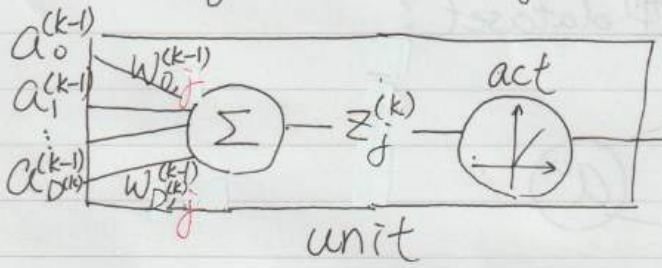
Def. Suppose parameters $\theta^{(1)}, \dots, \theta^{(L)}$ can be learn from training set X. A *feed forward neural networks* is a function composition

$$\hat{y} = f^{(L)}(\cdots f^{(2)}(f^{(1)}(x;\theta^{(1)});\theta^{(2)});\cdots);\theta^{(L)})$$

① Each function is called a *layer*

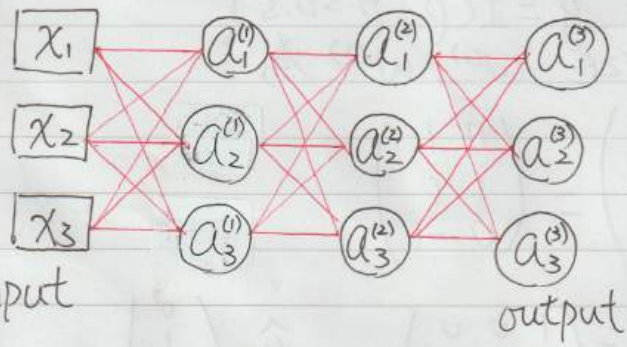
② Let $\theta^{(k)} = (w^{(k)}, b^{(k)})$, and $f^{(k)}$ be a *nonlinear function*. Suppose $a^{(k)} \in \mathbb{R}^{D^{(k)}}$ is the output of $f^{(k)}$, and $f^{(k)}$ *acts elementwisely*, then we denote $f^{(k)}$ by $act^{(k)}$, and $a^{(k)} = act^{(k)}(W^{(k)T}a^{(k-1)} + b^{(k)})$. Such $f^{(k)}$ is called an *activation function* $(\mathbb{R} \to \mathbb{R})$

③ Each $f_j^{(k)} = act^{(k)}(W_{:j}^{(k)T}a^{(k-1)})$ is called a *unit*



常直接以 "$a_j^{(k)}$" 表達 unit.

ex. A neural network $\hat{y} = f^{(3)}(f^{(2)}(f^{(1)}(x;\theta^{(1)});\theta^{(2)});\theta^{(3)})$. $x \in \mathbb{R}^3$, each output $a^{(k)} \in \mathbb{R}^3$, $\hat{y} = a^{(3)}$



* 我們常忽略 "1" 這個每層都有的 unit 以節省空間；事實上，故 "1" 是代表把 $W_{:j}^{(k)T}a^{(k-1)} + b^{(k)}$ 的 $b^{(k)}$ 就包進 $W_j^{(k)T}$ 裡.

NO.

DATE

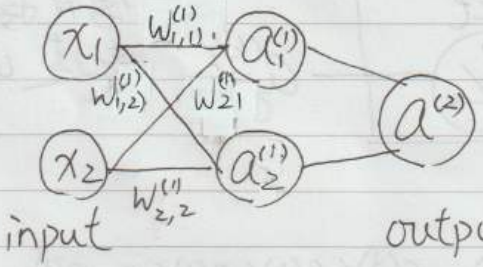
Remark. ① 並沒有規定每一層 layer 之 output $a^{(k)}$ 的維度要一樣

② output layer 做 binary classification → 可只用一個 unit，做一般 classification 可用與 class 數相同的 unit ⇒ output $\hat{y}$ 可視為一 pmf 或機率分佈

③ act 函數在 layer 數 > 3 時不可用 logistic function，要用非線性函數，否則 train 不起來

④ NN 中扣除 output layer 以外的 layer 稱為 hidden layer，功用如同做 feature engineering 一般.

ex. 給定 $\mathbb{X} = \{(x_1, x_2, y): (0,0,0), (0,1,1), (1,0,1), (1,1,0)\}$ 以及 NN: $\hat{\rho} = f^{(2)}(f^{(1)}(x; \theta^{(1)}); \theta^{(2)})$，試說明此 NN 如何學習此 XOR 型 dataset?



-sol- 取 $a^{(1)} = \max(0, W^{(1)T}x)$

$a^{(2)} = \sigma(W^{(2)T}a^{(1)}) \triangleq \hat{\rho}$

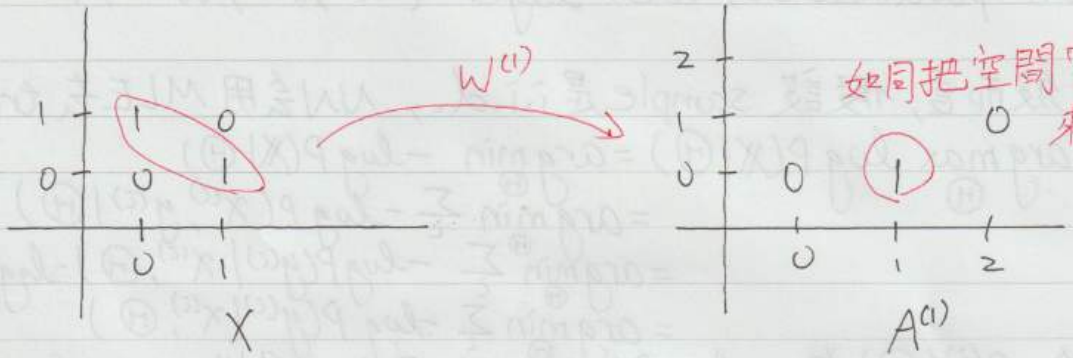
prediction: $\hat{y} = 1(\hat{\rho}; \hat{\rho} > 0.5)$

先隨机 initialize $W^{(1)}, W^{(2)}$ 為

$$W^{(1)} = \begin{pmatrix} w^{(1)}_{0,1} & w^{(1)}_{0,2} \\ w^{(1)}_{1,1} & w^{(1)}_{1,2} \\ w^{(1)}_{2,1} & w^{(1)}_{2,2} \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix},$$

$$W^{(2)} = \begin{pmatrix} -1 \\ 2 \\ -4 \end{pmatrix}, \quad \mathbb{X} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \quad \hat{y} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

則 $XW^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{pmatrix}$



令 $A^{(1)} = [1(\max(0, (XW^{(1)})_{ij}))]$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

$$\Rightarrow a^{(2)} = \sigma(A^{(1)} W^{(2)})$$

$$= \sigma\left[\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \\ -4 \end{pmatrix}\right] = \sigma\left[\begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}\right]$$

$$= \begin{pmatrix} \sigma(-1) \\ \sigma(1) \\ \sigma(1) \\ \sigma(-1) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \#$$

不过這是我們「猜到」了一个好的 $W^{(1)}, W^{(2)}$. 一般而言，如何訓練一个 NN 呢？

NO.

DATE

二. NN之訓練: Back propagatin

Given examples: $X=\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$
learn parameters of each layer $\{W^{(1)}, \dots, W^{(L)}\}$.

1. 一般而言, 假設 sample是i.i.d., NN会用 MLE去 train

$$\begin{aligned}\arg\max_{\Theta} \log P(X|\Theta) &= \arg\min_{\Theta} -\log P(X|\Theta) \\ &= \arg\min_{\Theta} \sum_i -\log P(x^{(i)}, y^{(i)}|\Theta) \\ &= \arg\min_{\Theta} \sum_i -\log P(y^{(i)}|x^{(i)}, \Theta) - \log P(x^{(i)}|\Theta) \\ &= \arg\min_{\Theta} \sum_i -\log P(y^{(i)}|x^{(i)}, \Theta)\end{aligned}$$

令 $C^{(i)}(\Theta)$ 為 $-\log P(y^{(i)}|x^{(i)}, \Theta)$, 称為 cost function

ex. 設 $P(y=1|x) \sim B(1, \rho)$, $x \in \mathbb{R}^p$, $y \in \{0, 1\}$.
各層 output $a^{(L)} = \sigma(z^{(L)})$, 則 cost function為:

$$\begin{aligned}C^{(i)}(\Theta) &= -\log P(y^{(i)}|x^{(i)}; \Theta) \\ &= -\log \left(a^{(L)y^{(i)}}(1-a^{(L)})^{1-y^{(i)}}\right) \\ &= -\log \left(\sigma(z^{(L)})^{y^{(i)}}(1-\sigma(z^{(L)}))^{1-y^{(i)}}\right) \\ &= -\log \sigma[(2y^{(i)}-1)z^{(L)}] \\ &= \zeta[(1-2y^{(i)})z^{(L)}], \ \zeta \text{為 softplus function}\end{aligned}$$

2. 大部份NN都可用SGD去解 $\arg\min_{\Theta} \sum_i C^{(i)}(\Theta)$

Algorithm: SGD
1. initialize $\Theta^{(0)}$ randomly
2. randomly partition X into minibatches of size M
3. $\Theta^{(t+1)} \leftarrow \Theta^{(t)} - \eta \nabla_{\Theta} \sum_{i=1}^{M} C^{(i)}(\Theta^{(t)})$
4. back to 2, repeat until convergence.

优点在於 收斂快, 且只要進行乘法与加法, 可以丟給 GPU, 但前提是要有一个算 $\nabla_{\Theta} \sum_i C^{(i)}(\Theta^{(i)})$ 之好方法

NO.

DATE

3. 反向傳播

總之，我們要算出 $\nabla_{\Theta}\sum_{n=1}^{M} C^{(n)}(\Theta^{(t)}) = \sum_{n=1}^{M} \nabla_{\Theta} C^{(n)}(\Theta^{(t)})$

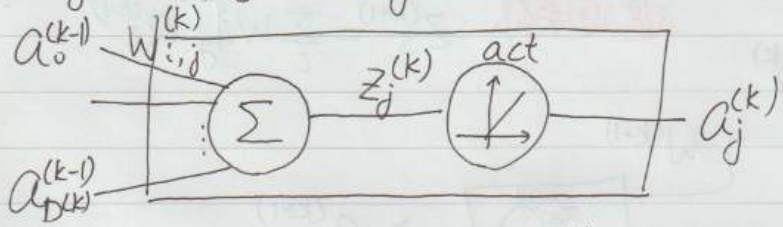
又，$\Theta = \{W^{(1)}, W^{(2)}, \cdots, W^{(L)}\}$，方便起見，

$\because W^{(k)} = [W_{ij}^{(k)}]$

$$= \begin{pmatrix} W_{0,0}^{(k)} & W_{0,1}^{(k)} & \cdots & W_{0,D^{(k)}}^{(k)} \\ W_{1,0}^{(k)} & W_{1,1}^{(k)} & \cdots & W_{1,D^{(k)}}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ W_{D^{(k)},0}^{(k)} & W_{D^{(k)},2}^{(k)} & \cdots & W_{D^{(k)},D^{(k)}}^{(k)} \end{pmatrix}$$

故，可將 $\nabla_{\Theta} C^{(n)}(\Theta^{(t)})$ 寫成 $\frac{\partial C^{(n)}}{\partial W_{ij}^{(k)}}$，$\nabla_{\Theta} C^{(n)}$ 可生成一個超長之向量，只要 $\forall i, j, k$ 都能算就好了（每一元素）。由 chain rule：

$$\frac{\partial C^{(n)}}{\partial W_{ij}^{(k)}} = \frac{\partial C^{(n)}}{\partial z_j^{(k)}} \cdot \frac{\partial z_j^{(k)}}{\partial W_{ij}^{(k)}}$$



(1) Forward pass：解 $\frac{\partial z_j^{(k)}}{\partial W_{ij}^{(k)}}$

When $k=1$, $z_j^{(1)} = \sum_i W_{ij}^{(1)} x_i^{(n)}$

$$\Rightarrow \frac{\partial z_j^{(1)}}{\partial W_{ij}^{(1)}} = x_i^{(n)}$$

When $k>1$, $z_j^{(k)} = \sum_i W_{ij}^{(k)} a_i^{(k-1)}$

$$\Rightarrow \frac{\partial z_j^{(k)}}{\partial W_{ij}^{(k)}} = a_i^{(k-1)}$$

⇒ 從最靠近 input 之層開始，逐層往後計算偏微分。

NO.

DATE

(2). Backward pass：解 $\frac{\partial C^{(n)}}{\partial z_j^{(k)}} \triangleq \delta_j^{(k)}$

When $k=L \Rightarrow \delta^{(L)}$取決於最後一層如何設計！

ex. binary classification:

$$\delta^{(L)} = \frac{\partial C^{(n)}}{\partial z^{(L)}} = \frac{\partial \xi((1-2y^{(n)})z^{(L)})}{\partial z^{(L)}} = \sigma((1-2y^{(n)})z^{(L)})(1-2y^{(n)})$$

When $k<L \Rightarrow$ 考慮

$$\delta_j^{(k)} = \frac{\partial C^{(n)}}{\partial z_j^{(k)}} = \frac{\partial C^{(n)}}{\partial a_j^{(k)}} \cdot \frac{\partial a_j^{(k)}}{\partial z_j^{(k)}}$$

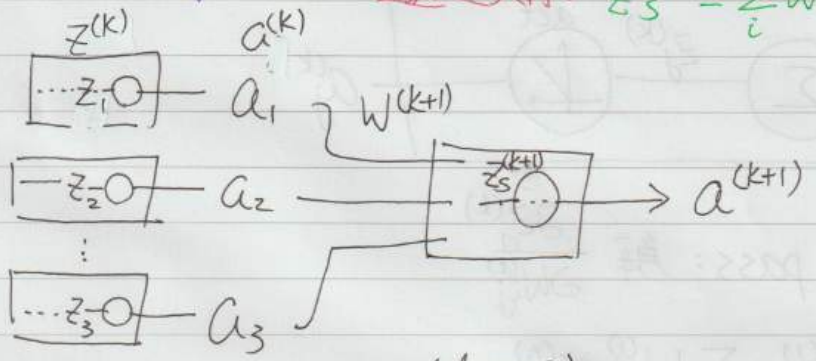
$$= \frac{\partial C^{(n)}}{\partial a_j^{(k)}} \cdot act'(z_j^{(k)})$$

by 多維之 chain rule

$$= \left(\sum_s \frac{\partial C^{(n)}}{\partial z_s^{(k+1)}} \cdot \frac{\partial z_s^{(k+1)}}{\partial a_j^{(k)}} \right) act'(z_j^{(k)})$$

$$= \left(\sum_s \delta_s^{(k+1)} \cdot w_{j,s}^{(k+1)} \right) act'(z_j^{(k)})$$

遞迴關係 $z_s^{(k+1)} = \sum_i w_{i,s}^{(k)} a_i^{(k-1)}$



(deepest)

故我們可從最靠近output那一層開始，逐層往回算 $\delta_j^{(k)}$

$$\delta_j^{(k)} = \left(\sum_s \delta_s^{(k+1)} w_{j,s}^{(k+1)} \right) act'(z_j^{(k)})$$

tensorflow會直接寫好

ex. def $(x^2)'$: return $2x$

由此一來，組合 forward 跟 backward pass，就能寫出 train function 了：

NO.

DATE

(3) Backprop with minibatch size $M=1$.

1. Input: $(x^{(n)}, y^{(n)})$ and $\Theta^{(t)}$.
2. Forward pass:
   $a^{(0)} \leftarrow [1, x^{(n)}]^T$
   for $k=1:L$
   $z^{(k)} \leftarrow W^{(k)T} a^{(k-1)}$
   $a^{(k)} \leftarrow act(z^{(k)})$
   end.
3. Backward pass:
   compute $\delta^{(L)}$
   for $k=L-1:1$
   $\delta^{(k)} \leftarrow \left(\sum_s \delta_s^{(k+1)} W_{j,s}^{(k+1)}\right) act'(z_j^{(k)})$
   end
4. Return $\frac{\partial C^{(n)}}{\partial W^{(k)}} = \left[\frac{\partial z_j^{(k)}}{\partial W_{ij}^{(k)}}\right]_{ij}^{(k)} = \left[a_i^{(k-1)} \delta^{(k)}\right]_{ij}^{(k)}$

* batch 設太大，對 training 有害！

(4) Backprop with minibatch size $M>1$
⇒ see ppT p.25.

三. Neuron Design for output layers

对於不同的任務，我們喜歡給予不同的 loss function，連帶的，最後一層 unit 之設計也要不一樣，以便於計算 $\delta^{(L)}$

1. Cross entropy.

Def. 令 $\hat{P}(y|x)$ 為 NN 之 output，則对 dataset 之分佈 $(x,y) \sim$ Empirical $(\hat{X})$ 而言，cross entropy (or KL Divergence) 為 $-E_{(x,y)\sim Emp(\hat{X})}[\log \hat{P}(x|y)]$

NO.

DATE

⇒ 我們希望NN能找到

$$\hat{p}_{NN} = \arg\min_{\hat{p}} -E_{(x,y)\sim Emp(X)}(\log \hat{P}(y|x))$$

2. Sigmoid units $\leftrightarrow$ $B(1,\rho)$.

若 $P(y=1|x) \sim B(1,\rho)$, $y\in\{0,1\}$ 且 $\rho\in(0,1)$, 則我們会把最後一層(只有一个 unit) 設計成

$$\hat{\rho} = a^{(L)} = \sigma(z^{(L)}) = \frac{e^{z^{(L)}}}{e^{z^{(L)}}+1}$$

$$\Rightarrow \delta^{(L)} = \frac{\partial C^{(n)}}{\partial z^{(L)}} = \frac{\partial}{\partial z^{(L)}} -\log \hat{P}(y^{(n)}|x^{(n)}; \textcircled{H})$$

$$= (1-2y^{(n)})\sigma[(1-2y^{(n)})z^{(L)}]$$

(1) 当 $\delta^{(L)} \to 0 \Leftrightarrow \begin{cases} y^{(n)}=1, & z^{(L)} >> 0 \\ y^{(n)}=0, & z^{(L)} << 0 \end{cases}$

(2). $C^{(n)}$ saturates only when $\hat{\rho}$ is correct

(3). Sigmoid units是專為SGD設計的

3. Softmax units $\leftrightarrow$ Catgorical output distribution

Def. 令 $\rho=(\rho_1,\dots,\rho^k)$, $\sum_i \rho^i = 1$, 則定義 catgorical output distribution 為 $Cat(\rho)$, 其 pmf 為

$$\begin{cases} P(y=e_1)=\rho_1 \\ P(y=e_2)=\rho_2 \\ \vdots \\ P(y=e_k)=\rho_k \end{cases}$$

其中 $e_i = (0,\dots,0,\underset{\uparrow}{1},0\dots0)$ (i^{th}) (onehot encoding).

NO.

DATE

設 $P(y|x) \sim Cat(\rho)$. 我們可用 softmax 放在最後一層:

$$\hat{\rho}_j = a_j^{(L)} = softmax(z^{(L)})_j = \frac{e^{z_j^{(L)}}}{\sum_{i=1}^{K} e^{z_j^{(L)}}}$$

$$\left(or = \frac{e^{z_j^{(L)}}}{\sum_{i=1}^{K-1} e^{z_i^{(L)}} + 1}\right)$$

$$\Rightarrow \delta_j^{(L)} = \frac{\partial C^{(n)}}{\partial z_j^{(L)}} = \frac{\partial}{\partial z_j^{(L)}} -\log\left(\prod_i \hat{\rho}_i^{1(y^{(n)}; y^{(n)}=i)}\right)$$

$$\begin{cases} ① y^{(n)}=j \\ \quad = -\frac{\partial \log \hat{\rho}_j}{\partial z_j^{(L)}} = -\frac{1}{\hat{\rho}_j}(\hat{\rho}_j - \hat{\rho}_j^2) = \hat{\rho}_j - 1 \\ ② y^{(n)}=i \neq j \\ \quad = -\frac{\partial \log \rho_i^{(L)}}{\partial z_j^{(L)}} = -\frac{1}{\hat{\rho}_i}(-\hat{\rho}_i \hat{\rho}_j) = \hat{\rho}_j \end{cases}$$

* $\delta_j^{(L)} \to 0$ if $\hat{\rho}_j$ is correct.

4. Linear units $\leftrightarrow$ Gaussian

Assume $P(y|x) \sim \mathcal{N}(\mu, \Sigma)$, 則我們用 linear units:

$$a^{(L)} = \hat{\mu} = z^{(L)}$$

$$\Rightarrow \delta^{(L)} = \frac{\partial C^{(n)}}{\partial z^{(L)}} = \frac{\partial}{\partial z^{(L)}} -\log g(y^{(n)}; \hat{\mu}, \Sigma)$$, 設 $\Sigma = I$,

則 $\delta^{(L)} = \frac{\partial}{\partial z^{(L)}} \| y^{(n)} - z^{(L)} \|^2$

此种 unit 不会 saturate, 但也無法 train 太举.

NO.

DATE

四. Neunn design for hidden layers

hidden layer 也可以用不同的 activation function 去設計；各种設計有不同需要考量的重点.

1. 梯度消失問題

(1) 背景：backward pass 中，$\delta_j^{(k)} = \sum_s \left(\delta_s^{(k+1)} \cdot w_{j,s}^{(k+1)}\right) act'(z_j^{(k)})$

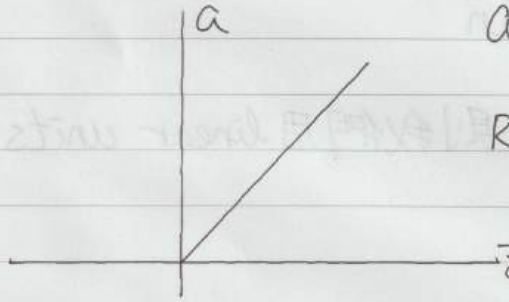
⇒ 若 $act'(\cdot) < 1$，則 $\delta_j^{(k)}$ 將隨 k ↑ 而越來越小

⇒ 超出系統精度後，直接變成 0!!

(2) 即使沒变成 0 ⇒ $\delta_j^{(k)}$ 太小，難以 train，拖累時間！

(3) logistic function 有此問題，故不应以其做为 hidden layer 的 activation function

2. ReLU



$$a = ReLU(z^{(k)}) = \begin{cases} z^{(k)}, & z^{(k)} \geq 0 \\ 0, & z^{(k)} < 0 \end{cases}$$

$$ReLU'(z^{(k)}) \triangleq \begin{cases} 1, & z^{(k)} > 0 \\ 0, & z^{(k)} < 0 \\ \text{隨机给 0 or 1}, & z^{(k)} = 0 \end{cases}$$

⇒ 解決梯度消失問題

(1) $ReLU''(\cdot) \triangleq 0$ on $\mathbb{R}$，以消除 second order effect

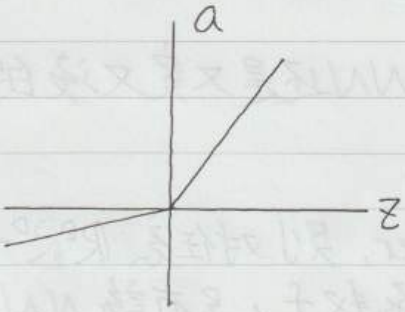
(2) 缺点：$\forall \delta_j^{(k)} = 0$，对应之 weight $w_{ij}^{(k)}$ 不被 update！

$$\left(\frac{\partial C^{(n)}}{\partial w_{ij}^{(k)}} = \delta^{(k)} \frac{\partial z_j^k}{\partial w_{ij}^{(k)}} \right.$$

NO.

DATE

(3) 改善方法：使用 Leakly ReLU

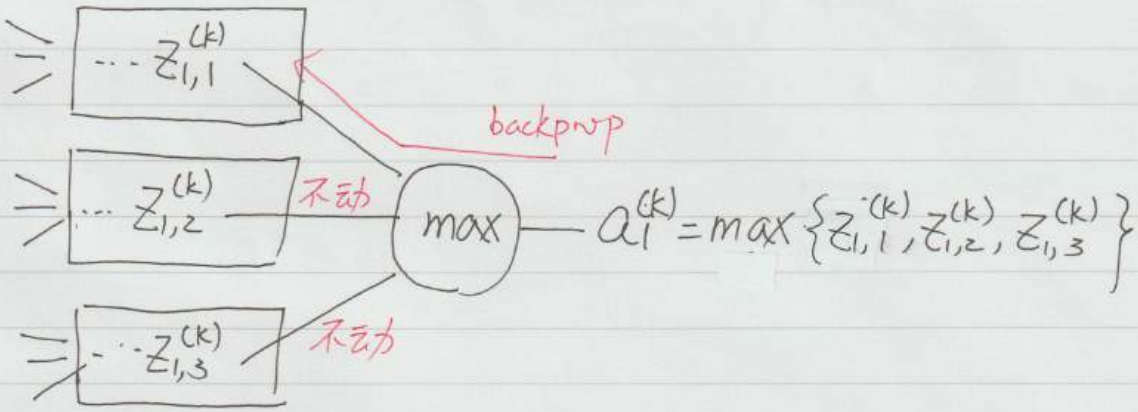


$act(z^{(k)}) = \max(\alpha z^{(k)}, z^{(k)})$,
for some $\alpha \in \mathbb{R}$

$\Rightarrow$ $\begin{cases} \alpha \text{ fixed (先給定)}: \text{Leakly Relu} \\ \alpha = -1: \text{absolute value rectification} \\ \alpha \text{ learned by GD}: \text{parametric relu} \end{cases}$

3. Maxout unit:

$$act(z^{(k)})_j = \max_s z_{j,s}$$



(1) backpwp 時，只要調最大那个就可以了

(2) 优点：避免「catastrophic forgetting」

(3) 缺点：需大量 data，否則要做 regularization

NO.

DATE / /

4. NN之結構选择

Q: 我們应該 train 一个又深又窄的 NN 还是又宽又浅的 NN?

Thm. (Universal Approximation Thm)
一个 NN 若至少有一个 hidden layer, 则对任意定義在 $\mathbb{R}^D$ 上 closed and bounded 子集的連續函数 f, 只要該 NN 足夠宽, 就能逼近 f.

問題是 data 有限, 未必做的到! 而深的 NN 可能可以把 data 分佈之 space 摺起來, 建立出很多 piecewise linear 的 region ⇒ better generalizability.

Ch11 Neural Networks: Optimization & Regularization

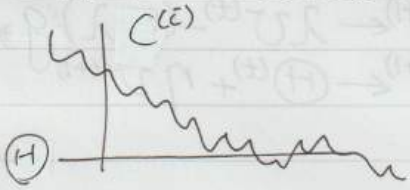
DATE

11-1 Optimization

Recall: 用 NN train - function，我們希望找出
$\arg\min_{\Theta} C(\Theta) = \sum_i C^{(i)}(\Theta)$

一. 常見的困難

1. SGD 常走不出 local minima 和 saddle point
(1) NN 之 cost function 呈鉅齒狀



(2) 小 NN 比較會遇到此問題，大 NN 還好

2. 若 $C^{(i)}$ 太 ill-conditioned，即 $\frac{\lambda_{max}}{\lambda_{min}}$ 太大，則訓練緩慢
成因：SGD 會繞遠路和抖動 (PPT p6)

3. 若 $C^{(i)}$ 跟本沒有 global minimum
⇒ SGD 會永遠往同一方向衝下去
解決方案：initialization 要做好 (做得好，一萬層的 NN 都 train 的起來

4. training 小撇步

(1) 對 input 做 standardize
⇒ prevent dominating features
improves conditioning

NO.

DATE

(2) Initialize all weights to small random values
→否則会有很多一樣的weight，unit之功能重複，使效果不彰

(3) Early stop: prevent overfitting.

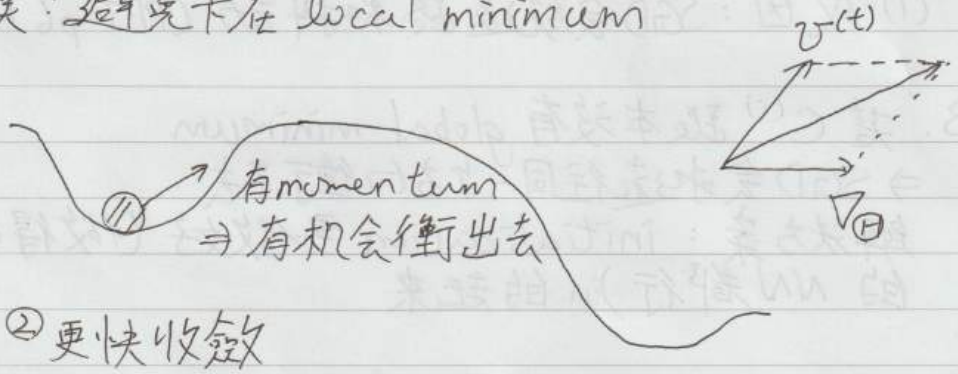
二. Momentum

1. update rule of SGD: $\textcircled{H}^{(t+1)} \leftarrow \textcircled{H}^{(t)} - \eta \nabla_{\textcircled{H}} C(\textcircled{H}^{(t)})$

⇒ 定義 momentum 為 $v^{(t+1)} \leftarrow \lambda v^{(t)} - (1-\lambda)\nabla_{\textcircled{H}} C(\textcircled{H}^{(t)})$
$\textcircled{H}^{(t+1)} \leftarrow \textcircled{H}^{(t)} + \eta v^{(t+1)}$

2. Nesterov Momentum
$\tilde{\textcircled{H}}^{(t+1)} \leftarrow \textcircled{H}^{(t)} + \eta v^{(t)}$
$v^{(t+1)} \leftarrow \lambda v^{(t)} - (1-\lambda)\nabla_{\textcircled{H}} C(\tilde{\textcircled{H}}^{(t)})$
$\textcircled{H}^{(t+1)} \leftarrow \textcircled{H}^{(t)} + \eta v^{(t+1)}$

3. 优点：①避免卡在 local minimum



②更快收斂

4. 缺点：沒有minima時沒有幫助

NO.

DATE

三. AdaGrad & RMSProp & Adam

1. 前言：卡在 { saddle point → 沒救，initialize做好一點
   { flat valley → 把 learning rate 弄得 adaptive試看

ex. 在陡峭的方向把 learning rate 調小一點 "大"
"平坦"

2. AdaGrad的 update rule：

$$r^{(t+1)} \leftarrow r^{(t)} + g^{(t)} \odot g^{(t)}$$

$$\textcircled{H}^{(t+1)} \leftarrow \textcircled{H}^{(t)} - \frac{\eta}{\sqrt{r^{(t+1)}}} \odot g^{(t)}, \quad g^{(t)} = \nabla_{\textcircled{H}} C(\textcircled{H})$$

其中 $\frac{\eta}{\sqrt{r^{(t+1)}}} = \frac{\eta}{\sqrt{t+1}} \odot \frac{1}{\sqrt{\frac{1}{t+1} r^{(t+1)}}}$

$$= \frac{\eta}{\sqrt{t+1}} \odot \frac{1}{\sqrt{\frac{1}{t+1} \sum_{i=0}^{t} g^{(i)} \odot g^{(i)}}}$$

learning rate ↓ ⇒ SGD不會亂跳

∇大，此項小；∇小，" 大

3. 缺點：實際上效果不大好 ⇒ 改良：RMSProp, Adam

4. RMSProp：

$$r^{(t+1)} \leftarrow \lambda r^{(t)} + (1-\lambda) g^{(t)} \odot g^{(t)}$$

$$\textcircled{H}^{(t+1)} \leftarrow \textcircled{H}^{(t)} - \frac{\eta}{\sqrt{r^{(t+1)}}} \odot g^{(t)}$$

NO.

DATE

5. Adam

$$v^{(t+1)} \leftarrow \lambda_1 v^{(t)} - (1-\lambda_1) g^{(t)}$$
$$r^{(t+1)} \leftarrow \lambda_2 r^{(t)} + (1-\lambda_2) g^{(t)} \odot g^{(t)}$$
$$\textcircled{H}^{(t+1)} \leftarrow \textcircled{H}^{(t)} + \frac{\eta}{\sqrt{r^{(t+1)}}} \odot v^{(t+1)}$$

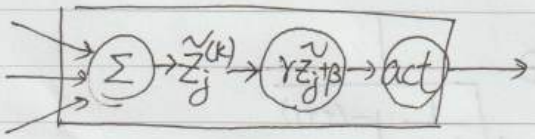
四. Batch Normalizatin

1. 前言：若 C $\textcircled{H}$ 太过 ill-anditioned，则 gradient-based 的方法（如 SGD）效率不好（PPT p19-21）

2. 考虑把 $a^{(k-1)}$ standardized，则 $g_i^{(t)} = \frac{\partial C}{\partial w^{(k)}}(\textcircled{H}^{(t)})$ 更可能让 C↓

⇒ Batch normalizatin：$\tilde{a}_i^{(k)} = \frac{a_i^{(k)} - \mu^{(k)}}{\sigma^{(k)}}$，$\forall i$

3. 也可以把 $z_i^{(k)}$ 标准化，并修改 unit 为



其中 $\tilde{z}_j^{(k)} = \frac{z_j^{(k)} - \mu^{(k)}}{\sigma^{(k)}}$，$\forall j$.

⇒ 利用 backprop 去 learn γ, β 以及 $w^{(k)}$

五. Continuatin methods & Curriculum Learning

1. 如何找比较好的 initializatin of $\textcircled{H}$？

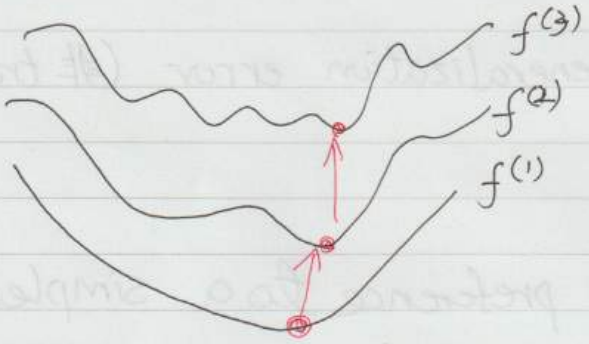
(1) train 一个 NN 很多次，每次都用不同的 random initial point ⇒ 选一个最好的

(2) 先设计一个简单的 cost function $f^{(1)}$，再以对 $f^{(1)}$ 之最佳解 $\textcircled{H}^{(1)}$ 做为稍复杂的 cost functin $f^{(2)}$ 之 initial point

NO.

DATE

⇒此方法稱、continuation methods



2. 不过 cost function 說不定不是 convex！此外，对缺乏 minima 之 NN 也沒啥用
⇒ DL 很少用

3. Curriculum learning (or shaping)
⇒ 先考簡單的問題
用簡單的問題先 train 出一个 weight
→当成下一个問題之 initial weight

ex. 要做 0☆ ~ 5☆ 評論的判斷器
→先用 0☆ V.S 5☆ train 出 model，
→当成 0☆，3☆，5☆判斷器的 initial weight
→ 生成原問題的 weight，train 出 model.

*本章中較有用的方法：
- standardize
- good initialization
- Adam
- batch normalization

11-2 Regularization

NO.

DATE

一. Regularization

1. 目的：減少 generalization error (非 training error)

2. 方法：
   (1) expressing preference to a simpler model
   (2) provide different perspective on how to explain the training data
   (3) Encoding prior knowledge

Q: data 很多時仍要 regularization 嗎？
A: 要！真實世界的 data 不是用類似 model 的東西產生的！要不然就是一個更複雜的 model 生出來的！

二. Weight Decay

Def. Weight Decay
When training an NN parameterized by $\mathcal{H}$, weight decay is to add norm penalties when solving $\arg\min_{\mathcal{H}} C(\mathcal{H}) + \alpha\Omega(\mathcal{H})$, where Ω can be such as L_1 or L_2 norm.

1. 好處：避免 hidden layer 的 feature dominant，即一顆 unit 有太大的 $z_j^{(k)}$.

2. Explicit norm penalties:
   $\arg\min_{\mathcal{H}} C(\mathcal{H})$ subject to $\Omega(\mathcal{H}) \leq R$

NO.

DATE

3. Projective SGD → 解 explicit norm penalties
⇒ 先 update $\textcircled{H}^{(t+1)}$，若 $\textcircled{H}^{(t+1)} \notin$ feasible set，則將其投影到 feasible set 上
⇒ 优点：避免 dead unit
因 learning rate 过大造成的不稳定

4. 最好一次使用 explicit constraints + reprojection + large learning rate (Hinton

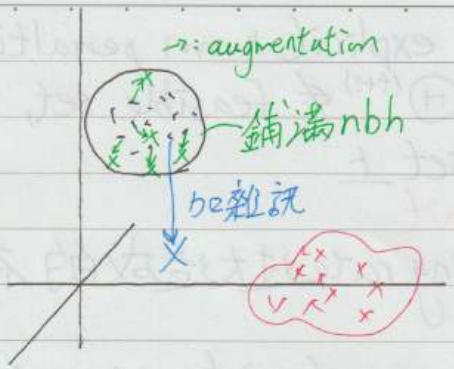
三. Data Augmentation

1. 前言：理論上有越多 data 越好，但假 data 很難製造！
⇒ 对，假設今天要分辨手寫数字，那把「5」轉 10°，还是「5」
⇒ 這种讓 dataset「膨胀」的方法称為 data augmentation

2. 在做 image classification 時，由 scaling, translating, rotating, flipping ... 生成新的 data (x, y)
** 不可 apply 会改变 $y^{(i)}$ 之 augmentation！如：把「6」転成「9」

3. 加雜訊也是一种 augmentation，但 NN 对雜訊有点太敏感 (PPT p49)
這是由於 data augmentation 的原理是把同一 class 之 data 所存在之 neighbourhood 給連起来鋪滿
(∴ dataset 相对 task 很大時可用 augmentation)

SEASON



但加了雜訊後，data跑到了「空洞」中，而NN在空洞中的表現與人類很不一樣

4. Noise injection：data $(x^{(i)}, y^{(i)})$，加入 random noise 到 $x^{(i)}$ 中，使 cost function 對 weight 的一些小變化較不敏感

* 也可加到 hidden layer

Ⅳ. Dropout

1. Ensemble method

(1) can improve generalizability by offering different explanations to ~~X~~

(2) 分類：

- Voting：使 variance ↓
- bagging：resample ~~X~~ ⇒ 讓 voter 不能串通，dependent ↓
- boosting：在不 overfitting 的前提下使預測信心 ↑

2. Drop out 的做法：SGD training 時，每次 load 一個 minibatch，而每個 unit 有 α 之機率被关掉不運作

通常 input units 取 $\alpha=0.8$，hidden units 取 $\alpha=0.5$

3. 原理：dropout 等同是一种 feature-based bagging

∵ ① resample input & latent features

② 等同 voter 間共享參数

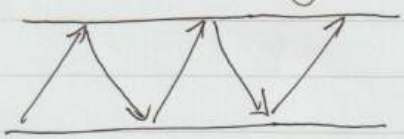
4. 优点：improve generalizatin beyond ensemble

NO.

DATE

五. 其他方法:

1. Cyclic learning rate: 使用 learning rate schedule



⇒ learning rate 在一給定形狀之範圍內振盪.

2. manifold regularization

⇒ 假設 data 是高維空間中的低維流形 K
(被鏡銷)

⇒ $\forall x \in K$, 找出其 tangent space, 令其 basis 為 $\{v^{(i,j)}\}_j$ (tangent vector).

⇒ Tangent Prop 由以下方式求 classifier f:

$$\Omega[f] = \sum_{i,j} \nabla_x f(x^{(i)})^T v^{(i,j)}$$

但求 $v^{(i,j)}$ 可能比較困難, 要依靠 domain knowledge 或一樣用 learn 的.

3. Domain-Specific Prior knowledge

⇒ 針對不同 task 去設計 NN

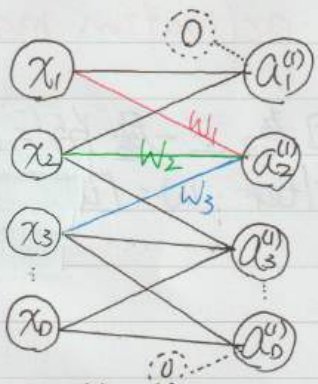
ex. Word 2 vec, CNN, ...

Ch12 Convolutional Neural Networks

DATE

一. 何謂 CNN

1. 先考慮只有一層的 NN，接成下面的形式



(1) 每一個 $a_i^{(1)}$ 都只接對應位置之上一個，相同，下一個的 unit

(2) 對 layer 1 的每一個 unit，規定所有連到其上的 weight 都要和其他人一樣

⇒ 實際上，「 」為一組

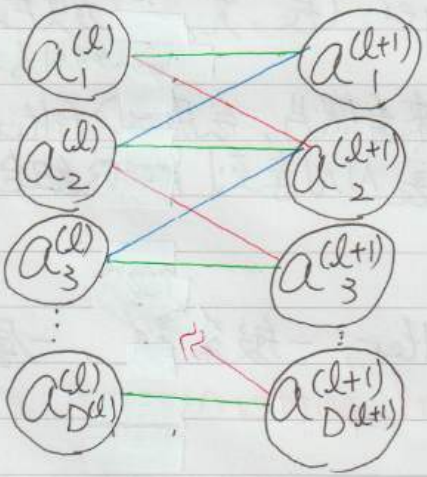
⇒ 每個第一層的 unit，拿到的是同一組 weight；對於邊緣的 unit，假裝有一個 "0" 連到上面，稱為 zero-padding

(3) 因此，$a_i^{(1)} = act^{(1)}\left([\,x_{i-1}\ x_i\ x_{i+1}]\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} + b^{(1)}\right)$，$\forall i = 1, \dots, n$. (定 $x_1 = x_{D+1} = 0$)

⇒ 要學的 weight 只有 w_1, w_2, w_3 三個！

⇒ 比 fully connected layer 簡單多了！

2. 現在，把 1 之 NN 推廣到任兩層間



(1) 每一個 $a_i^{(l+1)}$ 都接 $K^{(l)}$ 個上一 layer 之 unit，分別是 $a_{i-\frac{K^{(l)}}{2}}^{(l)}, \dots, a_i^{(l)}, \dots, a_{i+\frac{K^{(l)}}{2}}^{(l)}$

(2) 不夠接的那些（位在頭尾之）unit，進行 zero padding

(3) 每個 layer $l+1$ 的 unit 所連到的 weight 皆為 $[w_1, \dots, w_{K^{(l)}}]$

⇒ 只有 $w_1, \dots, w_{K^{(l)}}$ 這 $K^{(l)}$ 個 weight 要 learn！

同理，$a_i^{(l+1)} = act^{(l)}\left([a_{i-\frac{K^{(l)}}{2}}^{(l)}, \dots, a_i^{(l)}, \dots, a_{i+\frac{K^{(l)}}{2}}^{(l)}]\begin{bmatrix} w_1 \\ \vdots \\ w_{K^{(l)}} \end{bmatrix} + b^{(l)}\right)$

NO.

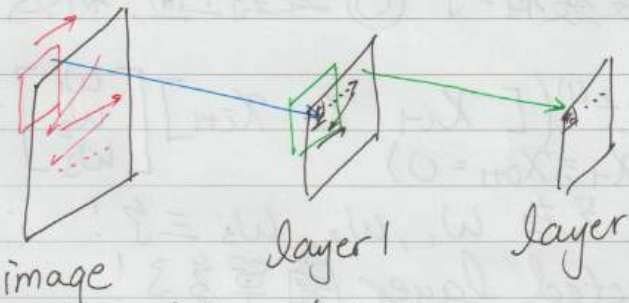
DATE

Def. 由2.所定義的layer稱為 convolution layer
$[W_1, \dots, W_{K^{(l)}}]$ 稱為 filter 或 Kernel
$act^{(l)}(\cdot)$ 稱為 feature map 或 activation map.

Remark. 會稱為convolution即是因為下一層 y_i 是上一層 $x=[x_1,\dots,x_D]$ 與 filter $w=[w_1,\dots,w_L]$ 做了 convolution

$$y_i = \sum_s x_s w_{i-s}$$

3. 2D圖片的 convolution layer ⇒ 自然也做的到



$$(x*w)_{ij} = \sum_u \sum_v x_{u,v} w_{i-u,j-v}$$

反正總有辦法連起來～

4. 3D圖片的 convolution layer (長,寬,三個顏色的 channel)
⇒ 考量人類不會「一個一個顏色」來看照片，會用3D filter 去掃這個3D input，以此讓NN學到RGB組合之 feature，據說效果較好

↗ 仍可稱為2D conv.

5. 一層也不一定只能有一個 filter，一般來說，一層中有一個 filter ⇒ detect 一個 local pattern

〈多的" ⇒ "多的〉

(1) 多了filter可以藉由增加下一層的厚度來完成
ex. 第 l 層有 $H \times W \times C$ 個 unit，上有 C 個 $K^{(l)} \times K^{(l)} \times 3$ 的 filter

NO.

DATE

⇒ 下一層有 $W^{(l+1)} \times H^{(l+1)} \times C$ 个 unit，如此继续
{ filter 大小为 $K^{(l+1)} \times K^{(l+1)} \times C^{(l+1)}$

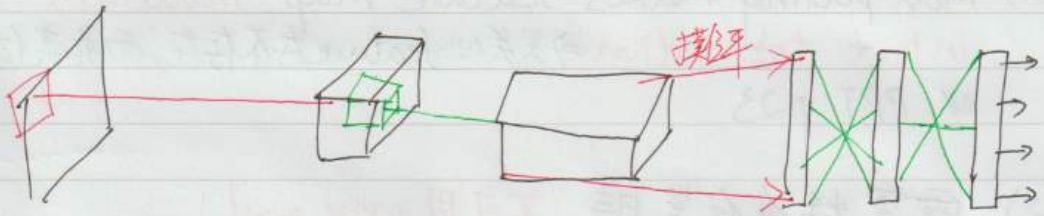
(2). 优点⇒使很多个 local pattern 被叠起来
⇒更易学到新的 pattern

(3). 按理，越深层的 convolution layer，其中一个 unit 所对应原图片的范围（称为 receptive field）也越大
⇒越深层的 layer，学到的是越 global 的 pattern.

6. 至此，可以给出一个 CNN 的定义了：

Def. A Convolutional Neural network (CNN) is an NN having at least one convolution layer.

一个常见的 CNN 架构：用很多 convolution layer 做完 pattern learning 后，将其最后一层摊平成一维的 layer (flatten layer)，接着经过若干 fully connected layer 后 output.



二. Pooling layer

Def. A pooling layer is a layer downsampling the feature map.

NO.

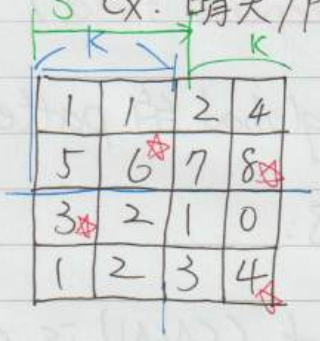
DATE

1. 原理：图片的"細節"即使解析度变差了，也不会让人覺得人脸变得不是人脸，树不是树.....

2. 方法：

(1) Max pooling：detecting edges；textures
ex. 偵測笑臉

(2) Average：detecting brightness or contrast
ex. 晴天/陰天之判別



max pool →

6	8
3	4

avg pool →

3.25	5.25
2	2

* 一次取K×K个做pooling，K称为filter的大小
做完後移动S格，再pool一次，S称为stride

3. backprop 時，在 forward pass 先記下 max 之 index（如上图所示），backward pass 時就只處理那些 index

4. 特点：

(1) max pooling makes feature map invariant to input translation ⇒只关心feature存不存在，而非其位置
ex. PPT p23

(2) 需要精準位置時，不可用 max pool！
ex. 自駕車

(3) maxpool層本身沒有需要 train 的參数.

* Exercise：試求下列 CNN 每一層共有几个 unit？共有几个要被 train 的 weight？

NO.

DATE

Input image: 256×256×3

layer	种類	參数
1	Convolution	16 filter, K=4
2	Max pooling	K=4, stride=4
3	Convolution	32 filter, K=4
4	Maxpooling	K=4, stride=4
5	flatten	
6	fully connected	10个 unit
7.	output	softmax, 10个 unit

-sol-. 1. ∵有16个 filter, 且 input為 256×256×3
∴有 256×256×16个 unit, weight共有 4×4×3×16个.

2. maxpool 沒有要 train 的 weight, 又 K=4, stride=4
⇒共有 (256/4)×(256/4)×16个 unit = 64×64×16个

3. 同1, 共有 64×64×32个 unit, 4×4×16×32个 weight

4. 共 (64/4)×(64/4)×32 = 16×16×32个 unit

5. ∵單純攤平而已
∴無 weight, unit 共 16×16×32 = 8192个

6. ∵ fully connected
∴共 8192×10 = 81920个 weight

7. ∵單純 output ∴無 weight

layer	unit	weight
1	256×256×16	4×4×3×16
2	64×64×16	0
3	64×64×32	4×4×16×32
4	16×16×32	0
5	8192	0
6	10	81920
7	10	0

#

NO.

DATE

三. 重要的CNN与其贡献

1. LeNet (LeCun, 1998)
   ⇒ 殷定至今CNN之架構基礎

2. AlexNet (2012)
   ⇒ 首次提出 dropout，第一个使用 ReLU 的 CNN，

3. VGG (2014)
   ⇒ 解決了 style transfer，也是第一个「深度」NN

4. ResNet (2016)
   ⇒ 解決了梯度消失問題

Ch13 Recurrent Neural Network

NO.

DATE

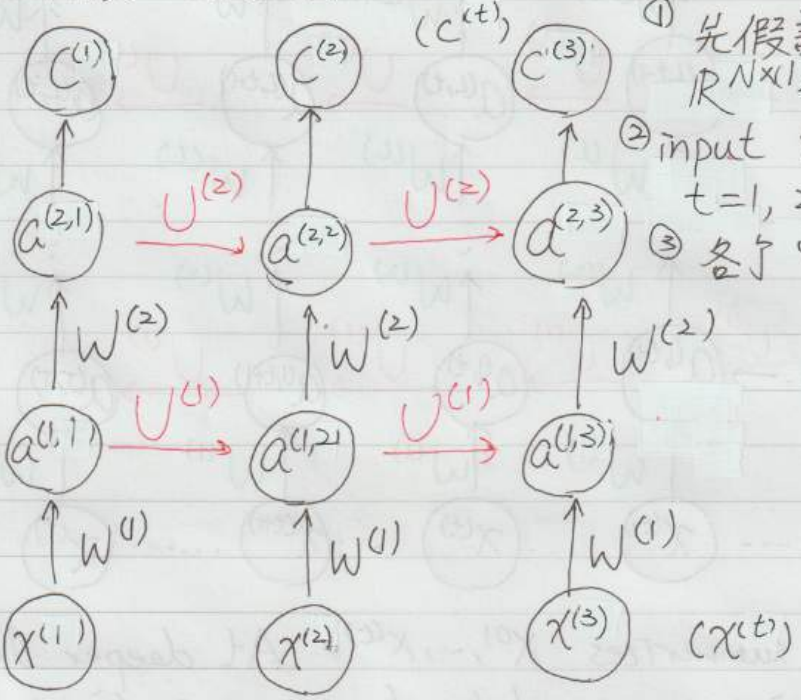
一. RNN的架構

1. Sequential Data:
data point (X, y) 不一定是 i.i.d.! 例如隨時間而變的 data：語音，文章，.... 此類 data set 記為
$X = \{X^{(n)}\}_n \subset \mathbb{R}^{N \times (D,K) \times T}$ (D: $x^{(n,t)}$之長度, K: $y^{(n,t)}$之長度)

(1) 每筆 data $X^{(n)} = \{(x^{(n,t)}, y^{(n,t)})\}$是一個 sequence，常被簡寫為 $(X^{(t)}, y^{(t)})$

(2) T 稱為 horizon，$x^{(n)}, y^{(n)}$之 T 可能不同

2. Vanilla RNN.



① 先假設 dataset 是 $\mathbb{R}^{N \times (1,1) \times 3}$, 即 $T=3$

② input $x^{(t)}, y^{(t)} \in \mathbb{R}$, $t=1, 2, 3$

③ 各了 "○" 為一 unit.

(1) ∵ $y^{(t)}$ depends on $x^{(1)}, \ldots, x^{(t)}$
∴ 我們希望學到一些 data 中 time invariant 之 fecture
⇒ 橫向，直向的 weight 不隨時間而變!!

(2). 可知 $a^{(k,t)} = act(z^{(k,t)})$
$= act(U^{(k)} a^{(k,t-1)} + W^{(k)} a^{(k-1,t)})$
各 layer
k(layer) $= 1, 2, 3$, t(time) $= 1, 2, 3$

NO.

DATE

3. vanilla RNN：考虑更一般的情形

每个element为vector对，所形成长度为T之sequence

$$\begin{cases} X=\{X^{(n)}\}_n \in \mathbb{R}^{N\times(D,K)\times T} \\ X^{(n)}=(x^{(n,t)}, y^{(n,t)})_t \end{cases}, \quad x^{(n,t)}\in\mathbb{R}^D,\ y^{(n,t)}\in\mathbb{R}^T,\ X^{(n)}\in\mathbb{R}^{(D,K)\times T}$$

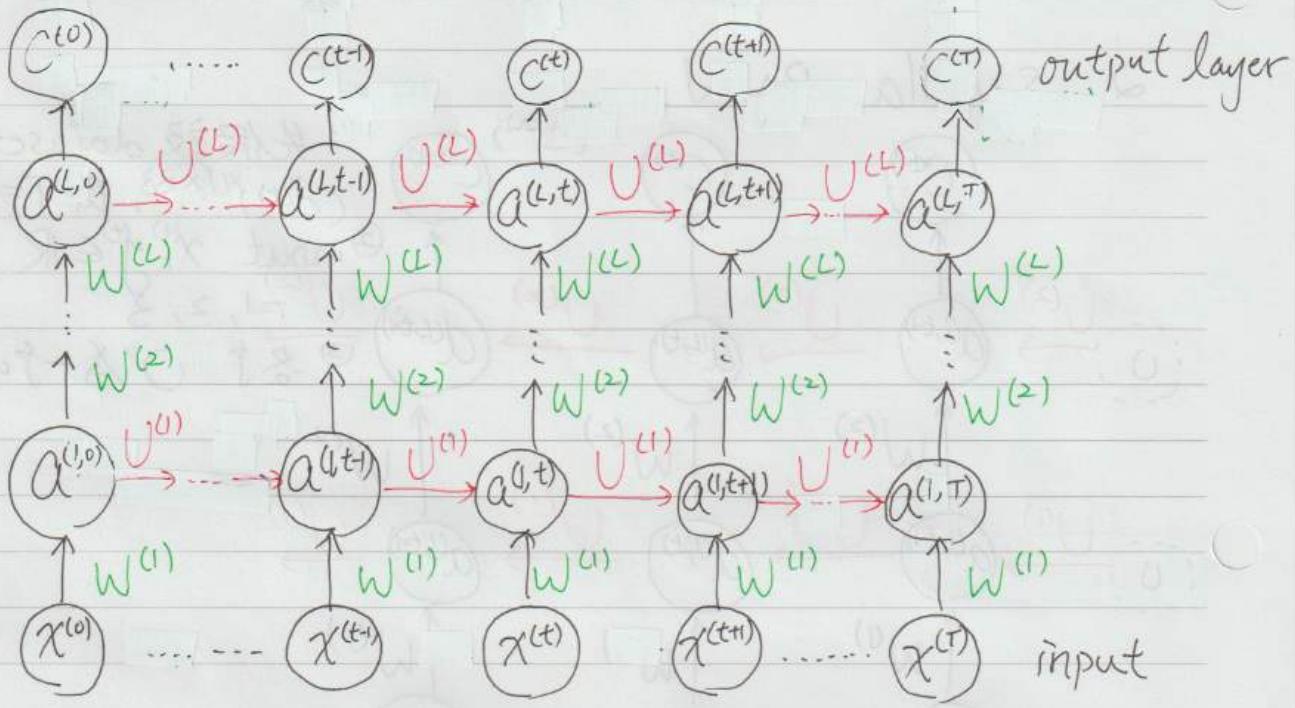
故对每一时刻 t，input $x^{(n,t)}$ 与 label $y^{(n,t)}$ 分别为 D 维及 K 维之 vector

↳ 简写为 $x^{(t)}, y^{(t)}$

⇒ model 如下：

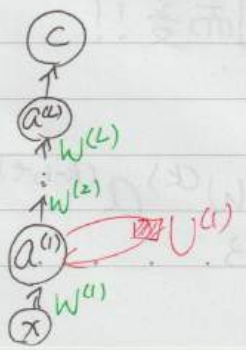
(1) $a^{(k,t)} = act(z^{(k,t)}) = act\left(U^{(k)}a^{(k,t-1)} + W^{(k)}a^{(k-1,t)}\right)$

(bias term omitted)



(2) $a^{(:,t)}$ summerizes $x^{(0)},\dots,x^{(t)}$. At deeper layers, $a^{(:,t)}$ give more abstract summarizations

(3) 为方便起见，我们会把上图之RNN fold成下图



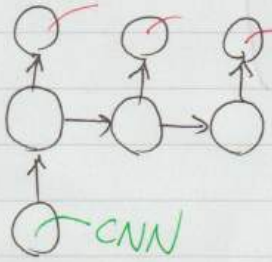
▨：memory

NO.

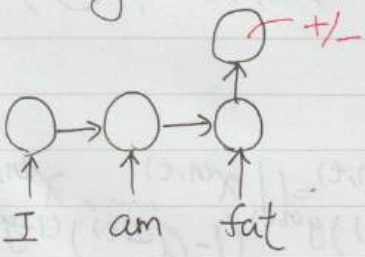
DATE

二. 不同种类的 RNN

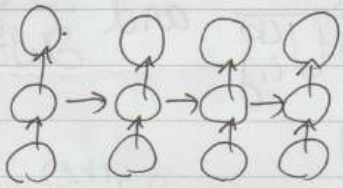
1. One to Many ex. Image Captining



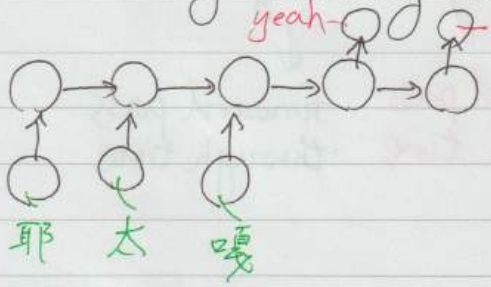
2. Many to one ex. Sentiment Analysis



3. Many 2 Many (Synced) ex. Language modeling



4. Many to Many (unsynced) ex. Machine Translatim



NO.

DATE

三. RNN之訓練

(思考 | 看到各種神經網路 → 架構、支体、訓練、應用)

1. Cost function of vanilla RNNs

by MLE,

$$\underset{\Theta}{\operatorname{argmin}} C(\Theta) = \underset{\Theta}{\operatorname{argmin}} -\log P(X|\Theta)$$

$$= \underset{\Theta}{\operatorname{argmin}} -\sum_n \sum_t \log P(y^{(n,t)} | x^{(n,t)}, \dots, x^{(n,1)}, \Theta)$$

$$= \underset{\Theta}{\operatorname{argmin}} -\sum_n \sum_t C^{(n,t)}(\Theta)$$

ex. in binary classification, $P(y^{(n,t)}=1 | x^{(n,t)}, \dots, x^{(n,1)}) \sim B(1, o^{(t)}) \Rightarrow C^{(n,t)}(\Theta) = (o^{(t,t)})^{y^{(n,t)}} (1-o^{(t,t)})^{(1-y^{(n,t)})}$

2. Using SGD, $\Theta^{(s+1)} \leftarrow \Theta^{(s)} - \eta \nabla_\Theta \sum_n \sum_t C^{(n,t)}(\Theta^{(s)})$

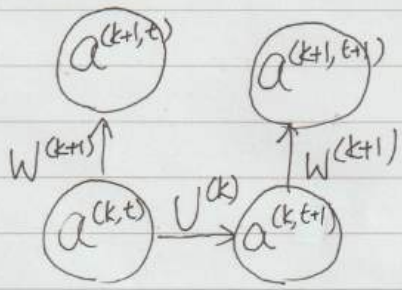
$\Rightarrow$ we need to evaluate $\frac{\partial C^{(n,t)}}{\partial U_{i,j}^{(k)}}$ and $\frac{\partial C^{(n,t)}}{\partial W_{i,j}^{(k)}}$

(1) $\frac{\partial C^{(n,t)}}{\partial W_{ij}^{(k)}} \Rightarrow$ similar to NN

(2) $\frac{\partial C^{(n,t)}}{\partial U_{ij}^{(k)}} = \frac{\partial C^{(n,t)}}{\partial z_j^{(k,t)}} \cdot \frac{\partial z_j^{(k,t)}}{\partial U_{ij}^{(k)}} \triangleq \delta_j^{(k,t)} \frac{\partial z_j^{(k,t)}}{\partial U_{ij}^{(k)}}$

$\delta_j^{(k,t)}$: backward pass through time

$\frac{\partial z_j^{(k,t)}}{\partial U_{ij}^{(k)}}$: forward pass through time



NO.

DATE

3. Forward pass through time ⇒ 解 $\frac{\partial z_j^{(k,t)}}{\partial U_{i,j}^{(k)}}$

$\because z_j^{(k,t)} = \sum_i w_{i,j}^{(k)} a_i^{(k-1,t)} + \sum_i U_{i,j}^{(k)} a_i^{(k,t-1)}$

$\therefore \frac{\partial z_j^{(k,t)}}{\partial U_{i,j}^{(k)}} = a_i^{(k,t-1)}$

⇒ 可從最 shallow (靠近 input) 以及時間最早的 unit 開始求解

4. Backward pass through time ⇒ 解 $\delta_j^{(k,t)} = \frac{\partial C^{(n,t)}}{\partial z_j^{(k,t)}}$

$\because \delta_j^{(k,t)} = \frac{\partial C^{(n,t)}}{\partial z_j^{(k,t)}} = \frac{\partial C^{(n,t)}}{\partial a_j^{(k,t)}} \cdot \frac{\partial a_j^{(k,t)}}{\partial z_j^{(k,t)}}$

$= \frac{\partial C^{(n,t)}}{\partial a_j^{(k,t)}} act'(z_j^{(k,t)})$

$= act'(z_j^{(k,t)}) \left(\sum_s \frac{\partial C^{(n,t)}}{\partial z_s^{(k+1,t)}} \cdot \frac{\partial z_s^{(k+1,t)}}{\partial a_j^{(k,t)}} + \sum_s \frac{\partial C^{(n,t)}}{\partial z_s^{(k,t+1)}} \cdot \frac{\partial z_s^{(k,t+1)}}{\partial a_j^{(k,t)}} \right)$

$= \left(\sum_s \delta_s^{(k+1,t)} w_{j,s}^{(k+1)} + \sum_s \delta_s^{(k,t+1)} U_{j,s}^{(k)} \right) act'(z_j^{(k,t)})$

⇒ 可從最 deep (靠近 output) 及時間最晚之 unit 算回去以求 $\delta_j^{(k,t)}$

5. 組合 2~4, 我們可以得出 train RNN 的方法:

(1) 使用普通的 Backprop 求 $\frac{\partial C^{(n,t)}}{\partial w_{i,j}^{(k)}}$

(2) 使用 Backprop through time 求 $\frac{\partial C^{(n,t)}}{\partial U_{i,j}^{(k)}}$ (BPTT)

- forward pass through time (single) → 求 $\frac{\partial z_j^{(k,t)}}{\partial U_{i,j}^{(k)}}$
- backward pass through time (multiple) → 求 $\delta_j^{(k,t)}$

NO.

DATE

四. 訓練RNN会遇到的問題&改善方法.

1. Exploding/Vanishing Gradient Problem

(1) 前言: 由前述Training 的討論, 可知 $a^{(k,i)}$ 与 $a^{(k,j)}$、$\delta^{(k,i)}$ 与 $\delta^{(k,j)}$ 是直接与 $(U^{(k)})^{j-i}$ 有关. 更甚者, 假設 (W, L, O, G_i) 忽略 activation function, 則

$$\begin{cases} a^{(k,j)} = (U^{(k)T})^{j-i} a^{(k,i)} \\ \delta^{(k,i)} = (U^{(k)})^{j-i} \delta^{(k,j)} \end{cases}$$

⇒ 若 $a^{(k,i)}$, $a^{(k,j)}$ 相距太遠, 即 $j-i$ 这个時間很大, 会引發一些 optimization problem! 這了原由称為 long-term dependency

(2). 若对 $U^{(k)}$ 做 eigendecomposition

$$\Rightarrow (U^{(k)})^{j-i} = Q \begin{pmatrix} \lambda_1^{j-i} & & \\ & \ddots & \\ & & \lambda_{D^{(k)}}^{j-i} \end{pmatrix} Q^T$$

⇒ When $j-i$ 很大, 則

- λ_i^{j-i} 超大 ⇒ Exploding gradient
- λ_i^{j-i} 趋小 ⇒ vanishing gradient

2. Cost surface: 通常很陡, 跟懸崖一樣

⇒ 解法:

(1) Nesterov momentum

(2) Gradient clipping: 若梯度超过一定值就把多出来的部份放掉 ⇒ effective in practice!

(3) RMSProp

(3) 解決方案: Sigmoid activation function

- bounded range in the forward pass
- $act'(\cdot) < 1$ in backward pass

但只能解決 $U^{(k)}$ 之 exploding 問題, vanishing

NO.

DATE

解决不了！还会 introduce vanishing gradient of $W^{(k)}$

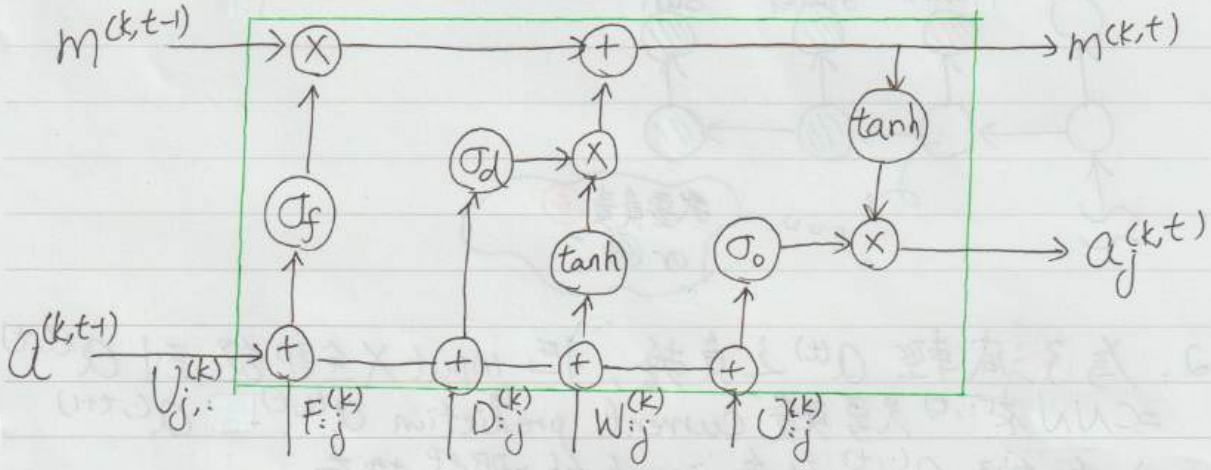
3. LSTM:

(1) idea: create shortcut in each neuron
⇒ error signals flows backward more smoothly

(2) 方法:

- tanh activation
- forget gate σ_f: 决定要不要忘掉 $m^{(k,t-1)}$
- input gate σ_d: " 储存第一个 activation 到 $m^{(k,t)}$
- output gate σ_o: " output 第一个 activation

↓ unit of LSTM



(3) equation:

(Left as an exercise!)

(4) 优点: LSTM + gradient clipping 非常有效

NO.

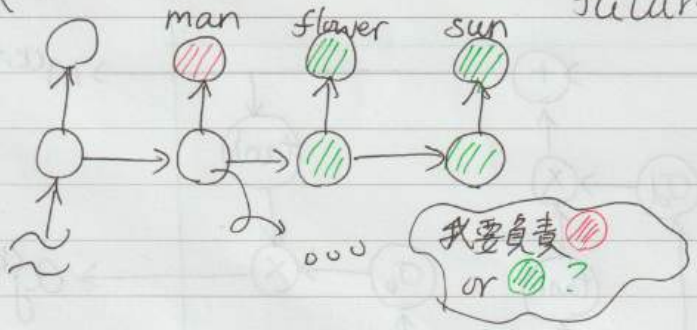
DATE

4. 平行化 train：由於要 pass through time, RNN的 training 是無法被平行化的. 但可藉由 output to hidden 或 teacher forcing 來替代

五. Attention — RNN訓練的最佳助手

1. 前言：在某些RNN中，一个 hidden representation $a^{(\cdot,t)}$ 需要負責 current prediction $a^{(L,t)}$ 以及 所有未來的 feature $a^{(L,t+1)}, \dots, a^{(L,T)}$

⇒ $a^{(t)}$ 的 trade off：究竟它該 〈 represent features for current prediction?
future " ?



2. 為了減輕 $a^{(t)}$ 之負擔，把 input X 全部餵到 $a^{(1,t)}$

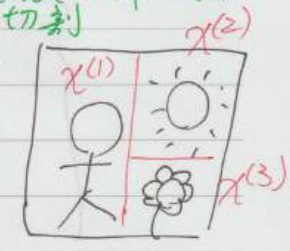
⇒ 每个 $a^{(\cdot,t)}$ 只要負責 current prediction $a^{(L,t)}$ 与 $a^{(\cdot,t+1)}$

⇒ 想了解 $a^{(\cdot,t)}$ 在看 input 的哪些地方

3. 方法：attention mechanism

實務上会用CNN的 filter 之 activation part 來做切割

(1) 假設 input 可被切成不同 part $\{x^{(i)}\}$



(2) 令 attention vector 為 $b^{(t)}$ 是由 $a^{(L+1,t-1)}$ 計算而來（方法後述）

(3) feed $a^{(1,t)}$ with the weighted input $c^{(t)} = \sum_i b_i^{(t)} x^{(i)}$，若我們實際畫出 $c^{(t)}$，会發現对应之 $x^{(i)}$ 变得較明亮

⇒ attention! (PPT p50).

NO.

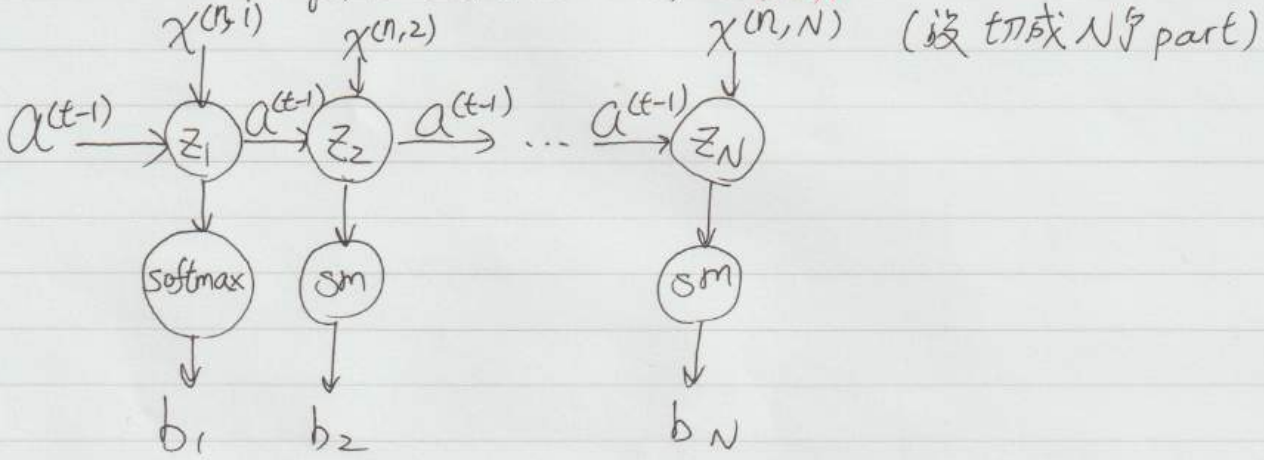
DATE

4. Computing the attention vector:

(1) 对每个 part $x^{(i)}$，以 $a^{(l-1,t-1)}$ 来算分数 (match score)。算分的方法可以是用 NN:

$$z_i = act(p^T a^{(l-1,t-1)} + q^T x^{(i)} + r)$$

* { jointly trained with the main RNN
  p, q, r 对不同的 i, t 共用



(2) 接着用 softmax normalize $\{z_i\}$

$$b_i = softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Date . . NO.

Ch14 Unsupervised Learning

一. Unsupervised learning的觀念

1. Data mining: Dataset $X=\{x^{(i)}\}_i$, $x^{(i)} \overset{i.i.d.}{\sim} x$
   ⇒ no labels! no supervision
   ex. K-means, dimension reduction (PCA), generation

2. Self-supervised learning
   (1) 目標: learn一个model使其可以"fill the blank"
   (2) 手法: 假設 x 是一个高維度空間裡的 manifold
   ⇒ 把這个manifold学起來

   ex. Mnist dataset. $\mathbb{R}^{28\times 28}=\mathbb{R}^{784}$, 但手寫字只存在於 $\mathbb{R}^{784}$ 中一个很小的 subspace上
   ⇒ learn每个data之 n.b.h. 的 tangent space 的基底

   (3) 重要范例:
   A. word2vec ⇒ 將一句話中每个字的 semantic 以一向量h来表示
   B. Doc2vec ⇒ 將文件中每一段給予一特別之ID, 並以整篇文章為 input
   ⇒ 学到沒有用文字敘述的 context (上下文关係)
   C. Filling Image ⇒ 將照片一塊挖掉並要 model 補上
   ∵ 如果把挖掉的區域当成 label y, 則 $\dim(y)$ 太大
   ∴ 可用 RNN 解
   ⇒ 一个pixel只depend附近之label
   ⇒ 比RNN是可 train的!

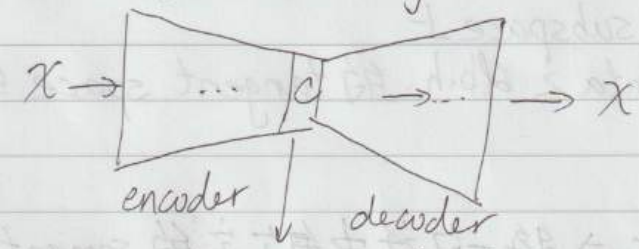
Def. A manifold is a topological space that are linear locally

⇒ 相邻之间可由 linear transform 近似，但大范讨论不行！

Remark. 对每个 manifold 上的点而言，其 tangent space 为 tangent vectors 的 span

二. Autoencoders

1. 由一个 encoder 和 decoder 组成，且标为输入 x 给 encoder 后使 decoder 能 regenerate x
   (1) 可当成 label 为 x 自身之 task
   (2) encoder 尾端和 decoder 前端被为一 bottleneck，否则不会把「有用的东西」embedd 进去 C
   ⇒ 会得到 trivially identical mapping



bottleneck（C为想要的embedding）

(3) Cost function:

$$\arg\min_{\Theta} -\log P(X|\Theta) = \arg\min_{\Theta} -\sum_{i=1}^{N} \log P(x^{(i)}|\Theta)$$

(4) 第 j 个 sigmoid output unit $a_j^L = \hat{\rho}_j$, $\forall x_j \sim B(1, \rho_j)$

（L 层之第）

$$\Rightarrow P(x_j^{(i)}|\Theta) = (a_j^{(L)})^{x_j^{(i)}} (1-a_j^{(L)})^{1-x_j^{(i)}}$$

(5) 第 L 层之 output $a^{(L)} \triangleq \hat{\mu}$, $x \sim N(\mu, \Sigma)$

（$\hat{\mu}$ 为 μ 之估计）

$$\Rightarrow -\log P(x^{(i)}|\Theta) = \|x^{(n)} - a^{(L)}\|^2$$

ex. 以 MNIST train — autoencoder，并将各数字学到的 embedding 称为 $C_0, C_1, \dots, C_9$

Date . . NO.

⇒ 將 train 好的 decoder 拆下，並以數字 1 的某 data $x_1^{(i)}$ 的 embedding $C_1^{(i)}$ 為 decoder 之輸入，預期為：

$$decoder(C_1^{(i)}) = \boxed{1} = x_1^{(i)}$$

同理，以數字 2 的某 data $x_2^{(j)}$ (embedding $C_2^{(j)}$) "

$$decoder(C_2^{(j)}) = \boxed{2} = x_2^{(j)}$$

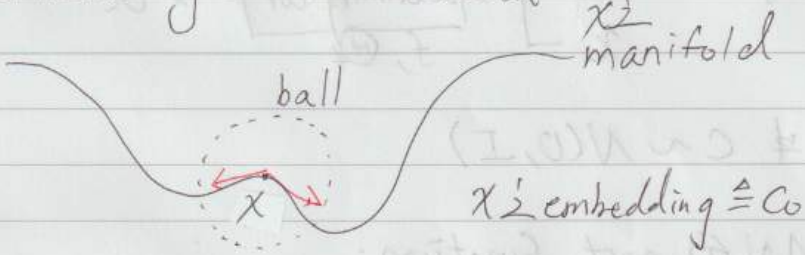
若以 $\frac{C_1^{(i)}+C_2^{(j)}}{2}$ 為 decoder 之 input，竟得到

$$decoder\left(\frac{C_1^{(i)}+C_2^{(j)}}{2}\right) = \boxed{?}$$

⇒ 介於 1、2 之間的圖形

⇒ 此 autoencoder 確實學到了 manifold!

2. embedding C 的物理意義：



設 $x \in$ 某 manifold，則希望 input 在該 ball 內的點，encoder 之 output 為 Co ∩manifold

⇒ C 為 manifold 之一種 coordinate

⇒ 可 regularize on 「local change of x 對應到的 change of C」

⇒ 以 Jacobian $J(x) = \frac{\partial C}{\partial x}$ 來建模，regularize on

$$\Omega(c) \triangleq \sum_{i=1}^{N} \left\| \frac{\partial C^{(i)}}{\partial x^{(i)}} \right\|_F^2$$

⇒ 對 $J(x)$ 做 SVD，得 $J(x) = UDV^T$，則可以最大的 singular values in D 所對應之 V 的 row 為 manifold 在 x 之 tangent vector 在x上

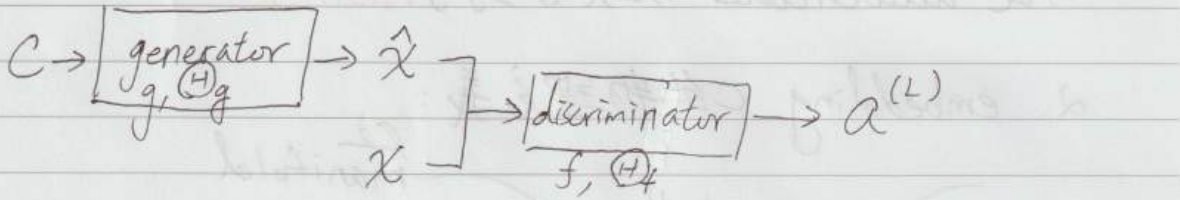
⇒ 此 tangent vector 即能使 C 變化最大的方向，且延此方向稍微移動仍在 manifold 上

⇒ 將 tangent space 聯集起來，可視為 manifold 之近似

三. Generative Adversarial Networks

1. 目標：generate data points from random codes

2. 構造：
   (1) Generator g：生成 image
   (2) Discriminator：分辨 real image 和 generator 生成之 image 之不同



其中 $C \sim N(0, I)$

3. GAN 的 cost function：

$$\arg\min_{\Theta_g} \max_{\Theta_f} \log P(X \mid \Theta_g, \Theta_f)$$ ⇒使 $\hat{x}$ 盡可能 $\approx x$，令 discriminator 分辨不出來

$$= \arg\min_{\Theta_g} \max_{\Theta_f} \sum_{i=1}^{N} \log f(x^{(i)}) + \sum_{j=1}^{N} \log(1 - f(g(c^{(j)})))$$

$$= \arg\min_{\Theta_g} \max_{\Theta_f} \sum_{i=1}^{N} \log \hat{\rho}^{(i)} + \sum_{j=1}^{N} \log(1 - \hat{\rho}^{(j)})$$

⇒ $\begin{cases} \rho^{(i)} \text{ depends on } \Theta_f \text{ only} \\ \rho^{(j)} \text{ "} \end{cases}$ both Θ_f and Θ_g

4. 如何 train GAN？

Date NO.

Algorithm. train GAN by SGD

1. Initialize θ_g, θ_f
2. At each iteration:
   (1) 將 θ_g 固定，並重複K次以下步驟：
      (i) Sample N个 real point $\{x^{(i)}\}_i$ from X
      (ii) " $c \sim N(0,1)$
      (iii) update θ_f

$$\theta_f \leftarrow \theta_f + \eta \nabla_{\theta_f} \left(\sum_{i=1}^{N} \log f(x^{(i)}) + \sum_{j=1}^{N} \log(1 - f(g(c^{(j)}))) \right)$$

   (2) 將 θ_f 固定，執行一次以下步驟：
      (i) Sample N个 $c \sim N(0, I)$
      (ii) update θ_g

$$\theta_g \leftarrow \theta_g - \eta \nabla_{\theta_g} \left(\sum_{i=1}^{N} \log(1 - f(g(c^{(j)}))) \right)$$

① 過允 overfit
② " g update 是錯的

5. GAN 很難 train!
   (1) 很難 converge ∵ The update of θ_g and θ_f may cancel each other's progress
   (2) mode collapsing ⇒ train 不起來
      ∵ K 很小時，alternate SGD 不知道 $\min_{\theta_g} \max_{\theta_f}$，$\max_{\theta_f} \min_{\theta_g}$ 之差別
      ∴ g 可能生成「騙得過 f，但騙不過人類」的 output
      ⇒ solutions: { minibatch discrimination / unrolled GANs

   (3). Ballance between g and f:
      { Too large K: f overfit data ⇒ ① g update for 「wrong」 target f ② ∇_{θ_g} vanishes
      Too small K: g updated for "meanless" f
      ⇒ solution: Wasserstein GAN

Date . . NO.

6. Information theory 下的 GAN

Thm. Let P_{data} be the distribution of x, P_g be the distribution of $\hat{x}=g(c)$. Then the max term in the cost function of GAN actually measures the JS divergence $\triangleq C^*$

$$D_{JS}(P_{data} \| P_g) = \frac{1}{2} D_{KL}(P_{data} \| Q) + \frac{1}{2} D_{KL}(P_g \| Q),$$

where $Q = \frac{1}{2}(P_g + P_{data})$

lemma1 For each x, assuming that f has infinite capacity, then we can find f maximizing

$$P_{data}(x) \log f(x) + P_g(x) \log(1-f(x)) \quad \text{——(1)}$$

to have C^*

pf.
$$C^* = \max_{\text{H}f} \sum_{i=1}^{N} \log f(x^{(i)}) + \sum_{j=1}^{N} \log(1-f(g(c^{(j)})))$$

$$= \max_{\text{H}f} \frac{1}{N} \sum_{i=1}^{N} \log f(x^{(i)}) + \frac{1}{N} \sum_{j=1}^{N} \log(1-f(\hat{x}^{(j)}))$$

$$\doteq \max_{\text{H}f} E_{x\sim P_{data}}[\log f(x)] + E_{x\sim P_g}[\log(1-f(x))]$$

$$= \max_{\text{H}f} \int_x P_{data}(x) \log f(x)\, dx + \int_x P_g(x) \log(1-f(x))\, dx$$

$$= \max_{\text{H}f} \int_x \left(P_{data}(x) \log f(x) + P_g(x) \log(1-f(x)) \right) dx$$

Q.E.D.

lemma 2 The f maximize (1) is

$$f^*(x) = \frac{P_{data}(x)}{P_{data}(x)+P_g(x)} \in [0,1]$$

pf. differentiate (1) w.r.t. x

Date . . NO.

pf of Thm: By lemma 1 and 2,

$$C^* = \max_{f} \int_x \left(P_{data}(x) \log f(x) + P_g(x) \log(1 - f(x)) \right) dx$$

$$= \int_x P_{data}(x) \log \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} dx + \int_x P_g(x) \log \left(1 - \frac{P_g(x)}{P_{data}(x) + P_g(x)}\right) dx$$

$$= -2 \log 2 + \int_x P_{data}(x) \log \frac{P_{data}(x)}{(P_{data}(x) + P_g(x))/2} dx$$

$$+ \int_x P_g(x) \log \left(1 - \frac{P_g(x)}{(P_{data}(x) + P_g(x))/2}\right) dx$$

$$= -2 \log 2 + 2 D_{JS}(P_{data} \| P_g)$$

Q.E.D.

Cor. The cost function of GAN is

$$\arg\min_{g} D_{JS}(P_{data} \| P_g)$$

⇒ 若 data 維數高, $x = g(z) \in$ low dimensional manifold,
則 P_g and P_{data} 只在 manifold 上定義
∴ $\{x \mid P_g(x) \neq 0 \text{ and } P_{data}(x) \neq 0\}$ 可能很小、
⇒ GAN 難 train!

四. GAN 之變體

1. Wasserstein GAN (W-GAN)
⇒ 以 modify 的 cost function 輔以 training 時的 gradient clip 來找 $P_g(x)$ 與 $P_{data}(x)$ 皆 $\neq 0$ 之集合

2. Improved W-GAN
⇒ 加了 gradient panelty 在 W-GAN 的 cost function 上, 使 convergence 加速

3. CapsuleNet ⇒ discriminator detects the existence of patterns more than their relative positions
⇒ 讓生成的image不要那麼「怪」

4. DC-GAN ⇒ 找一C對、畫出來的東西包含事先畫好的几筆

5. Conditional GAN → text to image synthesis
→ 已多被 diffusion model 取代

6. 其他GAN可做的事:
(1) 增加 image之 resolution
(2) image to image translation. ex 空拍圖→地圖

五. Concluding Remarks: Ch12 ~ Ch14所介紹的NN, 在今日(2024)几乎都被更先進的 model 取代掉了.

Past & current 之比較 → (voice/music)

model \ domain	sound	image	text
CNN	transformer	transformer	無
RNN	少見(*)	diffusion model	LLM
GAN	LLM(**)	diffusion model	LLM

*: 通常 sound 都用CNN以時頻譜做訓練
**: text to music

今日這些更 powerful 的商用 model 往往不公開其架構
⇒ 懂得如何操作 (prompt engineering) 更重要!

Ch16 Reinforcement Learning

NO.

DATE

一. 何謂強化學習

Def. Given an agent who sees states $S^{(t)}$, take actions $a^{(t)}$, and receives reward $R^{(t)}$, a reinforcement learning algorithm learns the best policy $\pi^*(S^{(t)}) = a^{(t)}$ that maximizes the total reward $\sum_t R^{(t)}$

1. Environment: 定義 S 為 environment 之集合, $S^{(t)}$ 為時刻 t agent 所在之 state.
   * S 不隨時間改變!

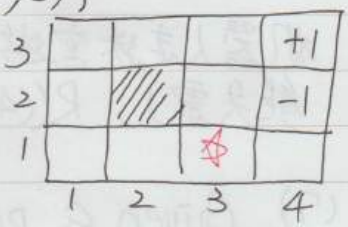
2. Reward $R^{(t)}$ 取決於 $S^{(t+1)}, S^{(t)}, \dots$ 以及 $a^{(t)}, a^{(t-1)}, \dots$ 把所有 action 所形成之集合稱為 action space A.

ex. 机器人走迷宮遊戲

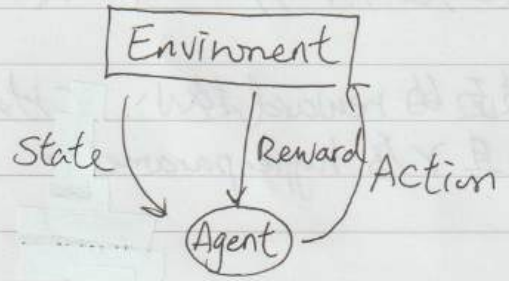
$S = \{(1,1), (1,2), (1,3), (2,1), (2,3), \dots, (4,3)\}$

$A = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$

若机器人一開始在 $(3,1)$, 接著依序經過 $(3,2)$, $(3,3)$, $(4,3)$, 則:



$$\begin{cases} S^{(0)} = (3,1), a^{(0)} = \uparrow, R^{(0)} = 0 \\ S^{(1)} = (3,2), a^{(1)} = \uparrow, R^{(1)} = 0 \\ S^{(2)} = (3,3), a^{(2)} = \rightarrow, R^{(2)} = 0 \\ S^{(3)} = (4,3), R^{(3)} = 1 \end{cases}$$



NO.

DATE

二. Markov Process

Def. A random process is a collection of time-indexed random variables $\{S^{(t)}\}_t$

Def. A Markov process $\{S^{(t)}\}$ is a random process satisfy the Markov property:
$P(S^{(t+1)} | S^{(t)}, S^{(t-1)}, \dots) = P(S^{(t+1)} | S^{(t)})$

Def. A Markov decision process (MDP) is defined by
(1) the state space S and the action space A
(2) Start state $s^{(0)}$
(3) The transition distribution $P(s'|s;a)$ (fixed over time t)
(4) the deterministic reward function $R(s,a,s') \in \mathbb{R}$
(5) A discount factor $\gamma \in [0,1]$
(6) The horizon $H \in \mathbb{N}$ (can be infinite)

ex. 机器人走迷宫游戏中，可定义 $P((3,2)|(3,1);\uparrow) = 0.8$ (马达可能失灵)，$R((4,1),\uparrow,(4,2)) = -1$

(7). Given a policy $\pi(s) = a$, an MDP proceeds as
$s^{(0)} \xrightarrow{a^{(0)}} s^{(1)} \xrightarrow{a^{(1)}} \cdots \xrightarrow{a^{(H-1)}} s^{(H)}$

(8) The accumulative reward
$= R(s^{(0)}, a^{(0)}, s^{(1)}) + \gamma R(s^{(1)}, a^{(1)}, s^{(2)}) + \cdots + \gamma^{H-1} R(s^{(H-1)}, a^{(H-1)}, s^{(H)})$

Remark. $\because 0 < \gamma < 1 \therefore$ 越後面的 reward 越小，γ 決定演算法要前瞻or短視，且 γ 為 hyperparameter

NO.

DATE

Def. Given a policy π, the expected accumulative reward collected by taking actions following π can be expressed by

$$V_\pi = E_{S^{(0)},\dots,S^{(H)}}\left[\sum_{t=0}^{H} \gamma^t R(S^{(t)}, \pi(S^{(t)}), S^{(t+1)}) \,;\pi\right]$$

Goal: find the optimal policy $\pi^* = \arg\max_\pi V_\pi$

二. Value iteration

Def. The optimal value function is the maximum expected accumulative reward when starting from state s and acting optimally for h steps

$$V^{*(h)}(S) = \max_\pi E_{S^{(1)},\dots,S^{(h)}}\left[\sum_{t=0}^{h} \gamma^t R(S^{(t)}, \pi(S^{(t)}), S^{(t+1)}) \,|\, S^{(0)}=S;\pi\right]$$

1. Value iteration of finite horizon

Having $V^{*(H-1)}(S)$ for each s, we can solve π^* by

$$\pi^* = \arg\max_\pi E_{S^{(0)},\dots,S^{(H)}}\left[\sum_{t=0}^{H} \gamma^t R(S^{(t)}, \pi(S^{(t)}), S^{(t+1)}) \,;\pi\right]$$

$$= \arg\max_a \sum_{S'} P(S'|S;a)\left[R(S,a,S') + \gamma V^{*(H-1)}(S')\right], \forall S.$$

於是有 recurrance ⇒ 必有 sub-optimal

⇒ 想到 dynamic programming

$$\Rightarrow \begin{cases} h=H-1, & V^{*(H-1)}(S) = \dots \\ h=H-2, & V^{*(H-2)}(S) = \dots \\ \vdots & \vdots \\ h=0, & V^{*(0)}(S) = \dots \\ h=-1, & V^{*(-1)}(S) = 0 \end{cases}$$

故可寫下下列 sudo code:

NO.

DATE

Algorithm: Value iteration (Finite Horizon)

Input: S, A, P, R, γ, H

step 1. $\forall s \in S$, initialize $V^*(s) \leftarrow 0$

step 2. For $h=0$ to $H-1$

For $s \in S$

$$V^*(s) \leftarrow \max_a \sum_{s'} P(s'|s;a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

step 3. For $s \in S$ do

$$\pi^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s;a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

step 4. Return $\pi^*(s)$, $\forall s \in S$

ex. 机器人走迷宫游戏中，設 $P(s'|s:a)$ 的 noise 为 0.2，即指令为"↑"時，真的往上之机率为 0.8，但往左 or 右之机率各为 0.1；往下之机率为 0，依此類推。若任一方向有障礙物，則「留在原地」之机率为這些有障礙之方向的机率和

給定 $\gamma=0.9$，求 $h=2$ 時，$V^{*(2)}(3,3)=?$

-sol- 当 $h=0$，只有 (2,3), (3,2), (3,3), (4,3) 会影响

$$\Rightarrow V^{*(1)}(3,3) = \max\{\uparrow: 0.9\times(0.8\times0+0\times0+0.1\times0+0.1\times1),$$
$$\downarrow: 0.9\times(0\times0+0.8\times0+0.1\times0+0.1\times1),$$
$$\rightarrow: 0.9\times(0.1\times0+0.1\times0+0\times0+0.8\times1),$$
$$\leftarrow: 0.9\times(0.1\times0+0.1\times0+0\times0+0.8\times1)\}$$
$$=0.72$$

$\Rightarrow$ 当 $h=2$，有

$$V^{*(2)}(3,3) = \max\{\uparrow: 0.9\times(0.8\times0.72+0\times0+0.1\times0+0.1\times1)$$
$$\downarrow: 0.9\times(0\times0.72+0.8\times0+0.1\times0+0.1\times1)$$
$$\leftarrow: 0.9\times(0.1\times0.72$$

SEASON

理論上，6步以內就可以 cover 所有格子，但也可以把 H 設到 ∞，PPT p17 是 $H=\infty$ 之結果

2. Bellman optimality equation:

当 $h\to\infty$，$V^*(s)=\max_a \sum_{s'} P(s'|s;a)[R(s,a,s')+\gamma V^*(s')]$，$\forall s$

由此可知，optimal policy $\pi^*(s)=\arg\max_a V^*(s)$，$\forall s$ 在 $h\to\infty$ 時

(1) stationary：optimal action 的方向不再变化

(2) memoryless：与 $S^{(0)}$ 無关

Algorithm: Value Iteration (Infinite Horizon)

Input: $S, A, P, R, \gamma, H=\inf$

step 1. $\forall s\in S$, $V^*(s)\leftarrow 0$

step 2. Do

For $s\in S$

$V^*(s)\leftarrow \max_a \sum_{s'} P(s'|s;a)[R(s,a,s')+\gamma V^*(s')]$

until $V^*(s)$ converge

step3. For $s\in S$

$\pi^*(s)\leftarrow \arg\max_a \sum_{s'} P(s'|s;a)[R(s,a,s')+\gamma V^*(s')]$

step4. Return $\pi^*(s)$, $\forall s\in S$.

3. The convergence of infinite horizon.

Thm. When $H\to\infty$, Value iteration converges and gives the optimal policy π^*

NO.

DATE

pf. Note that the reward can be negative.

$$\Rightarrow V^*(s) - V^{*(H)}(s) = \gamma^{H+1} R(s^{(H+1)}, a^{(H+1)}, s^{(H+2)}) + \gamma^{H+2} R(s^{(H+2)}, a^{(H+2)}, s^{(H+2)}) + \cdots$$

$$\leq \gamma^{H+1} |R(s^{(H+1)}, a^{(H+1)}, s^{(H+2)})| + \cdots + \gamma^0 |R(s^{(0)}, a^{(0)}, s^{(1)})|$$

Define $R_{max} = \max\{|R(s^{(i)}, a^{(i)}, s^{(i-1)})| : i \in \mathbb{N}\}$

$$\Rightarrow V^*(s) - V^{*(H)}(s) \leq \gamma^{H+1} R_{max} + \gamma^{H+2} R_{max} + \cdots = \frac{\gamma^{H+1}}{1-\gamma} R_{max}$$

$$\Rightarrow \lim_{H \to 0} V^*(s) - V^{*(H)}(s) = 0$$

$$\Rightarrow V^*(s) = \lim_{H \to 0} V^{*(H)}(s)$$

Q.E.D.

喜欢远目标　比较会坏 ⇒ 喜保险

4. γ值的选择：（PPT p21）

- γ大, noise大 → prefere distant exit, avoiding risk.
- γ大, noise小 → " distant ", risking.
- γ小, noise大 → " close ", avoiding risk.
- γ小, noise小 → " close ", risking.

γ小 ⇒ 短视 ⇒ 喜较近目标

比较不会坏 ⇒ 喜冒险

四. Policy iteration

Def. For a policy π, we define its value function

$$V_\pi(s) = E_{s^{(1)}, \cdots,} \left[\sum_{t=0}^{\infty} \gamma^t R(s^{(t)}, \pi(s^{(t)}), s^{(t+1)}) \mid s^{(0)} = s; \pi \right]$$

即照著π行事所得的 expected accumulated reward.

NO.

DATE

1. 如果要用這种方式去找 $\pi^* = \arg\max_{\pi} V_\pi$，那演算法理应長得像：

Input: $S, A, P, R, \gamma, H \to \infty$

step 1. $\forall s \in S$, initialize $\pi(s)$ randomly

step 2. Do

Evaluate $V_\pi(s)$, $\forall s$

Improve π s.t. $\forall s \in S$, $V_\pi(s)$ beames better

until $\pi(s)$ converge

step 3. Return $\pi(s)$, $\forall s \in S$.

2. 考虑 Bellman expectation equation (policy iteration 版)

$$V_\pi(s) = \sum_{s'} P(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')], \forall s \in S$$

(infinite!)

故，要計算 $V_\pi(s)$ 有 2 种方式：

(1) $V_\pi(s)$ 是 S 之方程式，又 $s \in S$，故解 $|S|$ 條联立方程式 ($O(|S|^3)$)

(2) 用 dynamic programming: (numercial friendly)

initialize $V_\pi(s) = 0$, $\forall s$

$$V_\pi(s) \leftarrow \sum_{s'} P(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')], \forall s$$

3. How to improve π.

⇒ We want to find $\hat{\pi}$ s.t. $V_{\hat{\pi}}(s) \geq V_\pi(s)$.

Prop. We should update π by the rule

$$\hat{\pi}(s) \leftarrow \arg\max_{a} \sum_{s'} P(s'|s;a)[R(s,a,s') + \gamma V_\pi(s')]$$

↓

下一步往哪更好？

NO.

DATE

pf. Consider $\forall s \in S$,

$$V_\pi(s) = \sum_{s'} P(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')]$$

$$\leq \sum_{s'} P(s'|s;\hat{\pi}(s))[R(s,\hat{\pi}(s),s') + \gamma V_\pi(s')]$$

$$\leq \sum_{s'} P(s'|s;\hat{\pi}(s))[R(s,\hat{\pi}(s),s') + \gamma \sum_{s''} P(s''|s';\hat{\pi}(s'))[$$
$$R(s',\hat{\pi}(s'),s'') + \gamma V_\pi(s'')]$$

$$\leq \cdots$$

$$\leq E_{s',s'',\ldots}[R(s,\hat{\pi}(s),s') + \gamma R(s',\hat{\pi}(s'),s'') + \cdots | s^{(0)} = s;\hat{\pi}]$$

$$= V_{\hat{\pi}}(s)$$

Q.E.D.

4. 由以上討論，可寫出 Policy Iteration 演算法：

Input: $S, A, P, R, \gamma, H \to \infty$

step1. Initialize $\pi(s)$, $\forall s \in S$

step2. Do

  For $s \in S$, $V_\pi(s) \leftarrow 0$

  Do

    For $s \in S$

      $V_\pi(s) \leftarrow \sum_{s'} P(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')]$

  until $V_\pi(s)$ converge

  For $s \in S$

    $\pi(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s;a)[R(s,a,s') + \gamma V_\pi(s')]$

until $\pi(s)$ converge

step3. Return $\pi(s)$.

Thm. When $H \to \infty$, policy iteration converges and gives the optimal policy π^*.

Date . . NO.

五. Reinforcement Learning 的概念-

1. 因[2]討論的為 MDP ⇒
   (1) 可能無法 model environment as an MDP
   ⇒ $P(s'|s;a)$ 可能未知!
   (2) Reward $R(s,a,s')$ 可能未知!

2. 对策:
   (1) Explore: perform actions randomly
   ⇒ estimate $P(s'|s;a)$ and $R(s;a,s')$ by collecting samples
   (2) Exploit: learn $P(s'|s;a)$ and $R(s,a,s')$
   ⇒ π^* can be computed using value/policy iteration

3. Model-based 与 Model free RL 之差異
   假設要估算 $E[f(x)|y] = \sum_x P(x|y) f(x)$ given samples
   $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, ..., 要怎麼做呢?
   (1) Model-based 的方法会直接 estimate
   $$\hat{P}(x|y) = \frac{\#(x^{(i)}=x \text{ and } y^{(i)}=y)}{\#(y^{(i)}=y)}$$
   接著算 $\hat{E}[f(x)|y] = \sum_x \hat{P}(x|y) f(x)$
   ⇒ 先把要用到的參数準備好
   (2) Model-free ⇒ 直接算 $\hat{E}[f(x)|y]$
   $$\hat{E}[f(x)|y] = \frac{1}{\#(y^{(i)}=y)} \sum_{i \in \{i | x^{(i)}=x,\, y^{(i)}=y\}} f(x^{(i)})$$
   ⇒ 由 sampling theory, 会work!

4. Model-free RL 之挑戰
   (1) Value Iteration 的 Algorithm 中 $V^*(s)$ 的 update

$\Rightarrow$① $V^*(s) = \max_\pi E\left[\sum_t \gamma^t R^{(t)} \mid s^{(0)} = s; \pi\right]$

除非真的 sample 到 true max, 否則無法 estimate!

② $\pi^*(s) \leftarrow \arg\min_a \sum_{s'} P(s'|s;a)[R(s,a,s') + \gamma V_\pi(s')]$

→ need model to solve

(2) Policy iteration:

① 和 Value iteration 相比, $V_\pi(s)$ 能以 MC estimation 去估計

② $\pi(s)$ 仍需 model 去 solve

六. Model-based RL

1. 演算法:

Algorithm. Model-based RL using MC estimation

1. Use some exploration policy π' to perform one or more episodes / trails

→ Each sample records samples of $P(s'|s;a)$, $R(s,a,s')$ from start to terminal state

(1) $s^{(0)} \xrightarrow[\pi'(s^{(0)})]{} s^{(1)} \xrightarrow[\pi'(s^{(1)})]{} \cdots \xrightarrow[\pi'(s^{(H-1)})]{} s^{(H)}$

(2) $R(s^{(0)}, \pi'(s^{(0)}), s^{(1)}) \rightarrow \cdots \rightarrow R(s^{(H-1)}, \pi(s^{(H-1)}), s^{(H)})$

2. Estimate $P(s'|s;a)$ and $R(s,a,s')$

(1) $\hat{P}(s'|s;a) = \dfrac{\#\text{times the action } a \text{ takes state } s \text{ to } s'}{\#\text{times action } a \text{ is taken in state } s}$

(2) $R(s,a,s')$ = Avg. of reward values received when a takes s to s'

3. Update the exploitation policy π

4. Repeat 1-3, but gradually mix π into π'

Date . . NO.

2. 困難点:
(1) $P(s'|s,a)$ and $R(s,a,s')$ 要估的量可能很多!
→ s, a, 甚至是連續的.
(2) 假如 $P(s'|s,a)$ 太低 ⇒ 都被 sample 到
⇒ 可能找不到最好的 policy
⇒ 若 $R(s,a,s')$ depends on s, 則 $R(s,a,s')$ 可能估的很爛!

3. 實務上 model-based 很難 work, 故常直接以 model-free 之方法去算 $V^*(s)$ or $V_\pi(s)$.

4. SARSA 演算法. (model free)
1. Q function:

Def. Q function for π:
$$Q_\pi(s,a) = E_{s^{(1)},\dots}\left[R(s,a,s^{(1)}) + \sum_{t=1}^{\infty} \gamma^t R(s^{(t)}, \pi(s^{(t)}), s^{(t+1)}); s, a, \pi\right]$$
s.t. $V_\pi(s) = Q_\pi(s, \pi(s))$ with recurrence
$$Q_\pi(s,a) = \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma Q_\pi(s', \pi(s'))\right],$$
$\forall s, a$.

Algorithm. Policy iteration based on Q_π
Input: MPP$(S, A, P, R, \gamma, H\to\infty)$
Output: $\pi(s)$'s, $\forall s$
1. For each state s, initialize $\pi(s)$ randomly
2. Repeat until $\pi(s)$ converge
(1) Initilize $Q_\pi(s,a)=0$, $\forall s, a$
(2) repeat until $Q_\pi(s,a)$ is converged:
$$Q_\pi(s,a) \leftarrow \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma Q_\pi(s', \pi(s'))\right]$$
(3) $\pi(s) \leftarrow \arg\max_a Q_\pi(s,a)$

Date . . NO.

(1) 此 Algorithm 之好處:

① 2-(2) 中，可用 MC estimation 估 $Q_\pi(S,a)$

$Q_\pi(S,a) = E[R^{(0)} + \sum_{t=1} \gamma^t R^{(t)}]$

② 2-(3) 中，$\pi(S)$ 之 update 不需要 model.

(2) 缺点：花费很多 trials，才 improve 一点 $\Rightarrow$ 太慢!

2. Temporal Difference Estimation: (TDE)

Algorithm. TDE of $Q_\pi(S,a)$:

① $\hat{Q}_\pi(S,a) \leftarrow$ random value, $\forall S, a$

② Repeat until conv., $\forall a^{(t)}$

$\hat{Q}_\pi(S^{(t)}, a^{(t)}) \leftarrow \hat{Q}_\pi(S^{(t)}, a^{(t)}) + \eta[(R(S^{(t)}, a^{(t)}, S^{(t+1)}) + \gamma \hat{Q}_\pi(S^{(t+1)}, \pi(S^{(t+1)}))) - \hat{Q}_\pi(S^{(t)}, a^{(t)})]$ —(*)

(1) (*) 可进一步改寫為

$\hat{Q}_\pi(S,\pi) \leftarrow \eta(R(S,a,S') + \gamma \hat{Q}_\pi(S', \pi(S'))) + (1-\eta)\hat{Q}_\pi(S,a)$

$\Rightarrow \eta \in [0,1]$ 之物理意義為 forget rate

(2) $\eta \to 0$ 時，$\hat{Q}_\pi(S,a)$ degenerates to the avg. of accumulative reward 且 $\hat{Q}_\pi(S,a) \to Q_\pi(S,a)$

3. SARSA 的步驟

Algorithm. SARSA

Input: S, A, γ

Output. $\pi^*(S)$, $\forall S$

1. Initialize $Q_\pi(S,a)$ arbitrarily, $\forall S, a$
2. For each episode, do:
   (1) Set s to initial state
   (2) repeat until S is terminal state
      ① $a \leftarrow \arg\max_{a'} Q_\pi(S,a')$ (先做动作)
      ② Observe S' and reward $R(S,a,S')$
      ③ $Q_\pi(S,a) \leftarrow Q_\pi(S,a) + \eta[(R(S,a,S') + \gamma Q_\pi(S', \pi(S'))$

$- \hat{Q}_n(s^{(t)}, a^{(t)})]$ (再做更新)

④ $s \leftarrow s'$

(1) 每次決定下一动作 a 時，policy 都会改进

(2) 將 V 改成 Q-function 是一大跳進，因為不僅走過了 value & policy iteration，且增快速度.

(3). SARSA 的收斂性：

Thm. SARSA converges and gives the optimal policy π^* almost surely if

(1) π is greedy in the limit with infinite exploration

(2) η is small enough eventually, but not decreasing too fast. Furthermore, η should satisfy $\sum \eta^{(t)} = \infty$ and $\sum \eta^{(t)2} < \infty$.

Remark. (1) Greedy in the limit: the policy $\pi \longrightarrow$ exploitation/greedy policy (in the limit)

(2) Infinite exploration: all (s, a) pairs are visited infinite times

→ SARSA 的第 2(2)① 無法保證！

4. Exploration Strategies:

現希望 mix-in / exploration 的策略使 infinite exploration with π' 可達成

(1). ε-Greedy Strategy

⇒ At every time step.

{ 以 ε 之机率隨机移动 (explore)

"$(1-\varepsilon)$" update exploitation policy 並隨之 act (exploit)

(2) Softmax Strategy

① 如果 a 曾经获得更多 accumulative reward，则就更有可能使用 a

⇒ 在 SARSA 中，可由此 distribution 来 sample a

$$P(a|s) = \frac{\exp(Q_\pi(s,a)/t)}{\sum_{a'} \exp(Q_\pi(s,a'))/t}, \forall a$$

t → temperature

② t 高：exploration

t 低：exploitation

(3) Exploration Function

⇒ to explore areas with fewest samples.

⇒ 比如：在 SARSA 中，定义 exploration function

$$f(q,n) = q + \frac{K}{n}$$

其中 { q 为 Q-value 之 estimate
{ n 为 the number of samples for the estimate
{ K 为某正数

⇒ 将 $Q_\pi(s,a)$ 的 update 改用

$$Q_\pi(s,a) \leftarrow Q_\pi(s,a) + \eta([R(s,a,s') + \gamma f(Q_\pi(s',a'), count(s',a'))] - Q_\pi(s,a)).$$

⇒ Infinite exploration; exploit once exploring enough.

八. Q-learning (model free)

Def. (Optimal Q function)

$$Q^*(s,a) = \max_\pi E_{s^{(1)},\ldots}\left[R(s,a,s^{(1)}) + \sum_{t=1}^{\infty} \gamma^t R(s^{(t)}, \pi(s^{(t)}), s^{(t+1)}); s, a, \pi\right]$$

s.t. $V^*(s) = \max_a Q^*(s,a)$ with the recurrance

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')], \forall s.$$

Def. Optimal value function.

$$V^*(s) = \max_{\pi} E_{s^{(0)},\dots}\left[\sum_{t=0}^{\infty} \gamma^t R(s^{(t)}, \pi(s^{(t)}), s^{(t+1)}) \,\middle|\, s^{(0)}=s; \pi\right]$$

with recurrence

$$V^*(s) = \max_a \sum_{s'} P(s'|s;a)[R(s,a,s') + \gamma V^*(s')]$$

Remark. ① OVF之物理-1-1：当初始 state 是 s 並 acting optimally onward 時，其 accumulative reward 之期望值最大值

② OQF: "
且 taking action a 時，"

⇒ 將 value iteration 改寫為 Q-value iteration

1. Q-value iteration

Algorithm. Q-value iteration.

Input: MDP $(S, A, P, R, \gamma, H \to \infty)$

Output: $\pi^*(s)$ $\forall s$

1. Initialize $Q^*(s,a) = 0$, $\forall s, a$
2. Repeat until $Q^*(s,a)$ converge
   $Q^*(s,a) \leftarrow \sum_{s'} P(s'|s;a)[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')]$
3. For each s do
   $\pi^*(s) \leftarrow \arg\max_a Q^*(s,a)$

2. Temporal Difference Estimation for exploitation policy π:

(1) $\hat{Q}^*(s,a) \leftarrow$ random value, $\forall s, a$

(2) Repeat until converge, $\forall a^{(t)}$

$$\hat{Q}^*(s^{(t)}, a^{(t)}) \leftarrow \hat{Q}^*(s^{(t)}, a^{(t)}) + \eta[R(s^{(t)}, a^{(t)}, s^{(t+1)}) + \gamma \max_a \hat{Q}^*(s^{(t+1)}, a)) - \hat{Q}^*(s^{(t)}, a^{(t)})]$$

Date . . NO.

3. Q-Learning的步驟

Algorithm. Q-Learning

Input: S, A, γ

Output: $\pi^*(s), \forall s$

1. Initialize $Q^*(s,a)$ arbitrarily
2. For each episode do
   (1) Set s to initial state
   (2) Repeat until s is terminal state
      ① Taking action a from s using some exploration policy π' derived from Q^* (ex. ε-greedy)
      ② Observe s' and reward $R(s,a,s')$
      ③ $Q^*(s,a) \leftarrow Q^*(s,a) + \eta((R(s,a,s') + \gamma \max_{a'} Q^*(s',a')) - Q^*(s,a))$
      ④ $s \leftarrow s'$

4. Q-learning之收斂性

Thm. Q-learning converges and gives the optimal policy π^* if

(1) All states and actions are visited infinitely often (π' has explored enough)

(2) η satisfies $\sum_t \eta^{(t)} = \infty$, $\sum_t \eta^{(t)^2} < \infty$.

5. SARSA与Q-learning之比較

Def. ① Off-policy: π updated toward a greedy policy indep. with π'

② On-policy: π uptated to improve (and depend on) π'

Date . . NO.

ex. off-policy: Q-learning
on- : SARSA

1. 實務上：SARSA 較保守，不願犯錯，但 Q-learning 較願意冒險，∵在 exploration policy 改變時可持續學習 ∴死的是別人(其他元)，願意冒險

2. 空間複雜度：SARSA 與 Q-learning 皆為 $O(|S||A|)$
⇒ memory usage 可能很大，要注意

*. Concluding remark: RL 除了在遊戲 AI 方面之應用外，今日之 LLM 亦將其應用於訓練中，其重要性可見一斑