Week 2 Tut

Exercise 1. ***Nearest Neighbor***

The dataset below consists of 4 examples described with 3 numeric features (a1, a2 and a3); the class has 2 values: yes and no.

What will be the prediction of 1-Nearest Neighbor (1-NN) and 3-Nearest Neighbor (3-NN) with Euclidian distance for the following new example: a1=2, a2=4, a3=2?

Assume that all attributes are measured on the same scale – no need for normalization.

	a1	a2	a3	class
1	1	3	1	yes
2	3	5	2	yes
3	3	2	2	no
4	5	2	3	no

Exercise adapted from M. Kubat, Introduction to Machine Learning, Springer, 2017

$d(1,new) = \sqrt{(2-1)^2+(4-3)^2+(2-1)^2} = \sqrt{1+1+1} = \sqrt{3}$

$d(2,new) = \sqrt{(2-3)^2+(4-5)^2+(2-2)^2} = \sqrt{1+1+0} = \sqrt{2}$

$d(3,new) = \sqrt{(2-3)^2+(4-2)^2+(2-2)^2} = \sqrt{1+4+0} = \sqrt{5}$

$d(4,new) = \sqrt{(2-5)^2+(4-2)^2+(2-3)^2} = \sqrt{9+4+1} = \sqrt{14}$

1-knn : will use d_2 $\Rightarrow$ class yes

3-knn will use d_1 d_2 d_3 $\rightarrow$ yes, yes, No $\rightarrow$ yes

vote by majority

***Exercise 2.* Nearest neighbor with nominal features**

Consider the *iPhone* dataset given below. There are 4 nominal attributes (age, income, student, and credit_rating) and the class is buys_iPhone with 2 values: yes and no.

What would be the prediction of 1-NN and 3-NN for the following new example:
age<=30, income=medium, student=yes, credit-rating=fair

If there are ties, make random selection.

Tip: As the examples are described with nominal attributes, when calculating the distance use the following rule:

difference=1 between 2 values that are not the same
difference=0 between 2 values that are the same
e.g. D(1, new)=sqrt(0+1+1+0=)=sqrt(2)

	age	income	student	credit rating	buy iPhone
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	[31,40]	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	excellent	no
6	[31,40]	low	yes	excellent	yes
7	<=30	medium	no	fair	no
8	[31,40]	medium	no	excellent	yes
9	>40	medium	no	excellent	no

new

age <=30
income = med
student = yes
cr = fair

$d(1, new) = \sqrt{0^2 + 1^2 + 1^2 + 0^2} = \sqrt{2}$

$d(2, new) = \sqrt{0^2 + 1^2 + 1^2 + 1^2} = \sqrt{3}$

$d(3, new) = \sqrt{1^2 + 1^2 + 1^2 + 0^2} = \sqrt{3}$

$d(4, new) = \sqrt{1^2 + 0^2 + 1^2 + 0^2} = \sqrt{2}$

$d(5, new) = \sqrt{1^2 + 1^2 + 0^2 + 1^2} = \sqrt{3}$

$d(6) = \sqrt{3}$

$d(7) = \sqrt{1}$

$d(8) = \sqrt{3}$

$d(9) = \sqrt{3}$

1 NN: use 7 → No

3NN: use 1, 4, 7 → No, Yes, No → No

Week 3 有 Loss Function $L=f(w)=\sum_{i=1}^{n}(W^Tx^i-y^i)^2+\lambda\sum_{j=1}^{m}w_j=\|Xw-Y\|^2+\lambda\|W\|^2$

① Linear Regression 注意 $X_0=1$

real data ↑ model check ↑ for overfitting

$$\underset{\text{Predict}}{\hat{y}}_{k\times 1}=W_0+W_1X_1+\cdots+W_nX_n=\underset{n\times k}{X^T}\underset{k\times 1}{W}=W^TX$$

- 那么怎么求上面的 W？使用 Loss Function

Method 1: Absolute loss $|y-\hat{y}|$

Method 2: Square loss $(y-\hat{y})^2$ #重点用这个因为 Mathmatic Property good

- How to minimize Square Loss Function ?

$$(\hat{y}-y)^2=\sum_{i=1}^{n}(\underbrace{\hat{y}^i}_{\text{predict}}-\underbrace{y^i}_{\text{real}})^2=\sum_{i=1}^{n}(\underbrace{W^Tx^i}_{\hat{y}^i}-y^i)^2$$

a. **Least square regression**：无法 avoid overfitting

time complexity : $O(\underbrace{nk^2}_{X^TX}+\underbrace{k^3}_{(X^TX)^{-1}})$ for building model

space complexity : $O(\underbrace{nk}_{X_{n\times k}}+\underbrace{k^2}_{(X^TX)_{k\times k}})$

我们要用 matrix form 来 minimize this Equation

$$f(w)=\min\|\underbrace{Xw}_{\hat{y}}-Y_{n\times 1}\|^2$$ #这个就是 Least square Regression

Take derivative of $f(w)$, we have

$$X^TXw - X^TY = 0 \Rightarrow W = (X^TX)^{-1}X^TY$$

↳ we want to find w that minimize the LSR

注意这里的 X 和 Y 中存储的都是 Training data

b. **Ridge Regression** : 使用了 Regulation Term avoid overfitting

$$f(w) = \min \|Xw - Y_{n\times 1}\|^2 + \lambda\|w\|^2$$

$$w = (X^TX + \lambda I_k)^{-1}X^TY$$

↑ closed form solution

- large λ : underfit
- Small λ: overfit
- L2 $\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$

c **Gradient descent** : 解决上面 time complexity 太大的问题

time complexity: $O(nk)$ per iteration

Space complexity $O(k)$

sum read to n

GD $w_{i+1} = w_i - \alpha_i \frac{df}{dw}(w_i) = w_i - \alpha_i \sum_{j=1}^{n}(w_i^T x^{(j)} - y^{(j)})x^{(j)}$

$= w_i - \alpha_i \nabla f(w_i)$

α_i: step size; $\frac{df}{dw}$: negative slope

- too small : converge slowly
- too big : overshoot, can diverge

↳ $\alpha_i = \frac{\alpha}{\sqrt{i}}$ ← hyperparameter

# of data → $\sqrt{i}$ ← iteration #

- **SGD**: $O(k)$ time complexity

  - 和 GD 相比，它只用 one observation，即省略了 sum

    $W_{i+1} = W_i - \alpha_i (W_i^T x^{(j)} - y^{(j)}) x^{(j)}$

    $= W_i - \alpha_i \nabla f_j(W_i)$ with j sampled at random

  - 问题是可能会导致 wrong direction，但总的来说还是会 slowly converge

  - Pros: 1. n times cheaper because one observation

    Cons: 1. less stable, may need more iter
    2. slower convergence

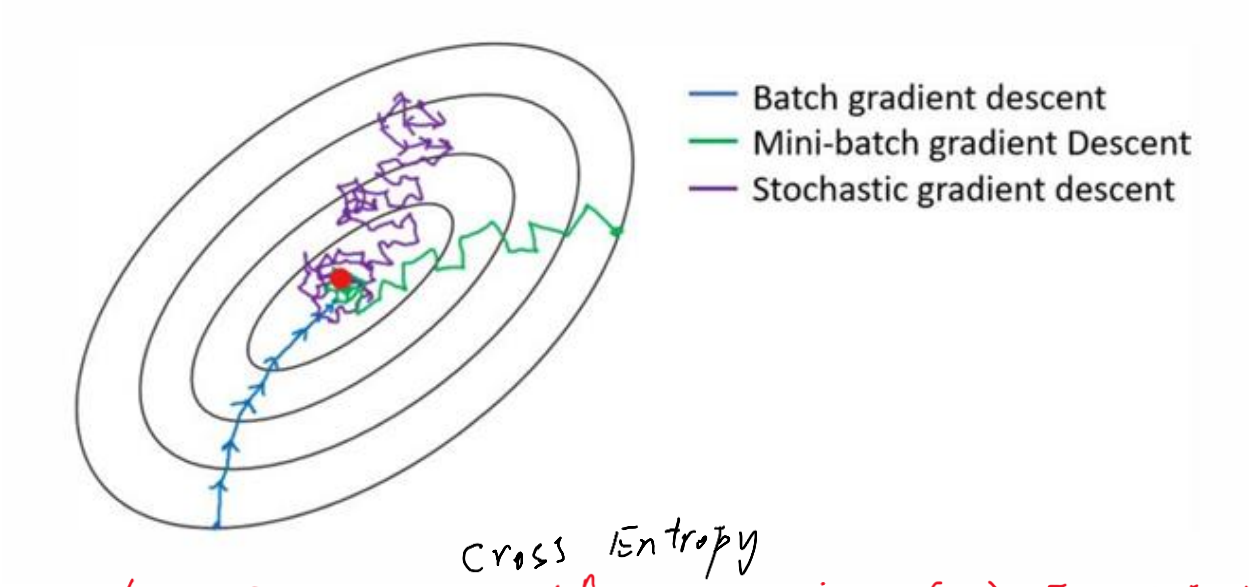
- **Mini-batch GD** Time complexity: $O(bk)$ per iter

  - 和 SGD 相比，我们不仅仅只 ob 一个，我们 ob 一小撮

  - $W_{i+1} = W_i - \alpha_i \nabla f_{B_i}(W_i)$ with $B_i \subseteq \{1, \dots, n\}$ sampled randomly

Pro: 1. 通过 adjust batch size，能在 computation 和 iter # 中找到 need more 的 trade-off

Con: 1. need to consider new hyperparameter batch size



cross entropy

有 Loss func $L = J(w) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log \sigma(w^T x^i + b) + (1-y_i)\log[1-\sigma(w^T x_i + b)]]$

② Logistic Regression

∵ Classification, ML, supervised

主要用于2分类，比如 decision bdy 大于为一类，小于为一类

- 正常 linear regression 返回 $\hat{y} = w^T x$
- 但是我们要 Probability，所以使用 sigmoid function，区间 $(0,1)$

$$P(\hat{y}|x) = \sigma(w^T x + b) = \frac{1}{1+e^{-(w^T x+b)}}$$

b是为了让decision bdy 可以不pass原点，从而增加灵活性

- 可以设置 threshold = 0.5，运算中 $P(\hat{y}|x) > 0.5 \Rightarrow \hat{y} = 1$

$\leq 0.5 \Rightarrow \hat{y} = 0$

- 又因为 $\sigma(0) = 0.5$

所以

$w^T x + b > 0 \Rightarrow \hat{y} = 1$

$w^T x + b < 0 \Rightarrow \hat{y} = 0$

- 那么如何求上面的 w？ Gradient + Cross Entropy Loss func

  - $W_{i+1} = W_i - \alpha_t \nabla f(w)$ ← 这就是 Gradient 时 Cross Entropy Loss function

  - 经过几个 iter，我们就能 get w^*

- 在给出 w^* and $\vec{x}$，我们怎么求 $P(y=1|\vec{x})$ 呢？

Suppose you see the following email:

```
CONGRATULATIONS!!  Your email address have won you the
lottery sum of US$2,500,000.00 USD to claim your prize
  contact your office agent (Athur walter) via email
  claims2155@yahoo.com.hk or call +44 704 575 1113
```

Keywords are [lottery, prize, office, email]
The given weight vector is $\mathbf{w} = [0.3, 0.3, -0.1, -0.04]^\top$

What is the probability that the email is spam?

$$\mathbf{x} = [1, 1, 1, 2]^\top$$

$$\mathbf{w}^\top \mathbf{x} = 0.3 * 1 + 0.3 * 1 - 0.1 * 1 - 0.04 * 2 = 0.42 > 0$$

$$\Pr(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-0.42}} = 0.603$$

Week 4. 没有 Loss Function

① Naive Bayes (ML, classification, supervised)

- 之所以 Naive 是因为 we assume attributes are independent and equally important

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \simeq \frac{P(E_1|H)P(E_2|H)\cdots P(E_n|H)P(H)}{P(E)}$$

- 假设 H 有两种 YES 和 NO，我们 need 分别求

$P(YES|E)$ 和 $P(NO|E)$ 再对比它们的大小.

- 使用公式的时候，分子部份可以从 Training data 中获得

分母 $P(E)$ 则可以不用求，因为在对比不同的 $P(H|E)$ 时会互相 cancel

Problem:

a. ~~ZERO-FREQUENCE~~ (某个 $P(E_i|H)=0$)

使用 Laplace correction, 即

$$P(E_i|H) = \frac{count(E_i)+1}{\#\text{ of sample} + m}$$

E_i: 只对为 0 的那个 term 使用

m: # of possible value for E_i

b. Missing Values

如果 X 中不包含某个 E_i 怎么办？

我们可以在公式中也不包含

$$P(E|H) = \frac{P(E_1|H)\,P(E_{i+1}|H)\cdots P(E_n|H)\,P(H)}{P(E)}$$

上面就是不包含 E_2 的解法

C. 之前我们解的都是 discrete Problem，现在来看看 numeric 的怎么解（i.e. attribute 包含 numeric 而不是 nominal）

· 使用 pdf

只求1个 E_i，我们还need求别的 E_i

$$\rightarrow P(H|E_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(E_i-\mu)^2}{2\sigma^2}}$$

$$\rightarrow \sigma = \sqrt{\frac{\sum_{i=1}^{n}(E_i-\mu)^2}{n-1}}$$

· 一般来说 std dev 和 mean 会给，我们要做的是区分不同的 H 对应的 μ 和 σ

Discussion：

a. 如果 Attributes 之间高度 related，我们可以在 pre-processing 中只选一部分 Attri

b. 如果 numeric Attri 不是 normal distri,
我们可以使用其实的 pdf

Examples:

Solution (2)

E1 is home owner = no, E2 is marital status = married, E3 is annual income=very high

P(yes)=5/10

P(E1|yes)=P(home owner=no|yes)=3/5
P(E2|yes)=P(marital status=married|yes)=1/5
P(E3|yes)=P(annual income=very high|yes)=1/5

$$P(yes|E) = \frac{\frac{3}{5}\cdot\frac{1}{5}\cdot\frac{1}{5}\cdot\frac{5}{10}}{P(E)} = \frac{\frac{3}{250}}{P(E)} = \frac{0.012}{P(E)}$$

P(no)=5/10

P(E1|no)= P(home owner=no|no)=3/5
P(E2|no)=P(marital status=married|no)=3/5
P(E3|no)=P(annual income=very high|no)=2/5

$$P(no|E) = \frac{\frac{3}{5}\cdot\frac{3}{5}\cdot\frac{2}{5}\cdot\frac{5}{10}}{P(E)} = \frac{\frac{9}{125}}{P(E)} = \frac{0.072}{P(E)}$$

P(no|E)>P(yes|E) => Naïve Bayes predicts **loan default = no** for the new example

	home owner	marital status	income	loan default
1	yes	single	very high	yes
2	no	married	high	yes
3	no	single	medium	no
4	yes	married	very high	no
5	yes	divorced	high	yes
6	no	married	low	no
7	yes	divorced	very high	no
8	no	single	high	yes
9	no	married	medium	no
10	no	single	low	yes

这里我是只有 nominal attributes 的 case

2) Calculate P(income=120|yes) and P(income=120|no) using the probability density function for normal distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$f(income = 120|yes) = \frac{1}{15.57\sqrt{2\pi}} e^{-\frac{(120-99)^2}{2*15.57^2}} = 0.01032$$

$$f(income = 120|no) = \frac{1}{66.18\sqrt{2\pi}} e^{-\frac{(120-109)^2}{2*66.18^2}} = 0.00595$$

class yes income	class no income
125	70
100	120
95	60
85	220
90	75
μ_income_yes=99	μ_income_no=109
σ_income_yes=15.57	σ_income_no=66.18

这里 Assume $\vec{X}$ 中包含了一个 numeric 的 income，其中 $E_i = 120$. 求完这个后，我们还要知道 E_i 会怎么得到总的 $P(H|E) = \frac{P(E_1|H)\,P(Income=120|H)\cdots P(H)}{P(E)}$

② Evaluate ML (评测 Accuracy)

- 使用 test data set

— Holdout Method

分为 training set | Validation Set | Testing Set

可以多次使用，再求 avg 就叫作 repeated

build the model | fune hyperparameter | evaluate accuracy

70 | 15 | 15

Holdout • 容易产生样本在每个 set 分配不均的问题，
使用 Stratification 来解决，即每个 set sample
proportion 和 original data set 一样

— Cross validation: Usually 10-folds
↳ 很多 Experiments show this is the best

- 把 original data set 分成 10 个，
- 每次用 9 个 build model，用剩下的一个测试
- 最后得到 avg (acc1, acc2, ···, acc10)

— leave-one-out 特殊的 cross-validation，fold # 是 data points 的数量
- 不适合 large data set
- No random sampling involved: the same result will be obtained every time

③ 怎么使用 Accuracy 来调参？

— Grid Search:
对于每一种 Hyperparameter 组合 我们都使用

上面的方法求 Accuracy，最后选一个 Accuracy 最高的组合，然后用整个 training set 再 train one final model

- 比如 Hyper = { x: [1, 2], y: [3, 4], z: [5, 6] }

我们要做 10-fold，只这点的 Build 次数就是 2×2×2×10 = 80 次

GridSearch CV

④ Performance Measurment

— Confusion Matrix : Not a performance Measure, just allow us to calculate performance measures

	Predict class1	Pre class2
real T	True posi	fn
real F	fp	True nega

$$Accuracy = \frac{Tp + Tn}{fp + fn + Tp + Tn}$$

- E.g. iris data classification - confusion matrix:

predict b, real c

```
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 44  6 |  b = Iris-versicolor
  0  3 47 |  c = Iris-virginica
```

- accuracy =?

- accuracy =
(50+44+47)/(50+0+0+0+44+6+0+3+47)
=141/150 =94%

$$P = \frac{tp}{tp + fp} \qquad R = \frac{tp}{tp + fn} \qquad F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

Tut:

- Gaussian NB — numeric Data
  Categorical NB — Categorical Data / nominal
  Bernoulli NB } Text Data
  Multinomial NB }

- best cross-validation score:
  we get from avg($acc_1, \cdots, acc_{10}$)
  test set is 10% of training set

- test set score:
  after we select the best hyper, we use 100% training set train the model. Then we calculate the accuracy of this model in test set

对比于 Knn, logistic Regression, 它的 model 更容易 visualize, 就是 non-linear 的

↑

Week 5

没有 loss function

Unstable classifier: small change in training set result in large change in prediction

① Decision Tree (ML, classification, supervised, divide and conquer)

easy to visualize

- Non-leaf node: Attributes
- Branch: value of Attributes
- Leaf node: label

- Construct
  (1) select the best attributes as **root**
  (2) split according to the value of current node
  (3) Repeat above steps

一个 Decision Tree 如果分到最后，leaf node 只有一种 class 是最好的

使用 Information gain + Entropy 来计算(1)中的 best attri. 其中 $\log_2 0 = 0$

best attribute is the one with highest information gain

☆ 每一个 Non-leaf node 都要用这个来

Entropy 公式为：S是当前 node 分到的 data

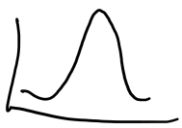
$$H(S) = -\sum_{i}^{\text{class}} P_i \log_2 P_i$$

P_i = # of sample which is class i / # of all samples

Information Gain 公式：

$$Gain = T_1 - T_2$$

Entropy = 1



Entropy 不为 1

T_2 → Current state $H(S)$

→ Sum of children state $H(S)$ with correspondent weight ☆

Example 1

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

9个YES

5个 No

14个 data

Q1 想知道选哪个 attribute 做为当前 Node.

Step1：先从 outlook 看起

a. 求 T_1

$$T1 = H(S) = I(\frac{9}{14}, \frac{5}{14}) = 0.940\ bits$$

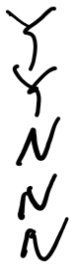
Outlook S

Sunny → S_1

overcast → S_2

Rainy → S_3

(通过table) 5个data



4个

5个

b. 求 T_2

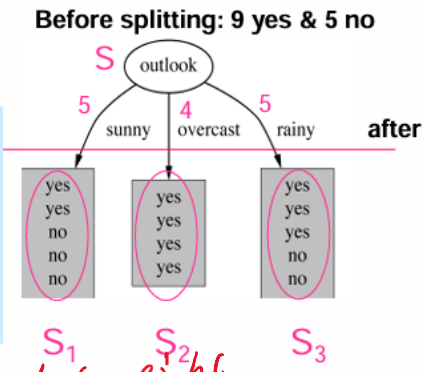
$$T2 = H(S \mid outlook) = \frac{5}{14} \cdot H(S_1) + \frac{4}{14} \cdot H(S_2) + \frac{5}{14} \cdot H(S_3)$$

$$H(S \mid outlook = sunny) = I(\frac{2}{5}, \frac{3}{5}) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971\,bits$$

$$H(S \mid outlook = overcast) = I(\frac{4}{4}, \frac{0}{4}) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0\,bits$$

$$H(S \mid outlook = rainy) = I(\frac{3}{5}, \frac{2}{5}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971\,bits$$

$$H(S \mid outlook) = \frac{5}{14} \cdot 0.971 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0.971 = 0.693\,bits$$



☆ with correspondent weight

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes

Gain for "outlook" = $T_1 - T_2 = 0.94 - 0.693 = 0.247$

Step2 求 Gain for "temp"

Gain = 0.029

Step3 求 Gain for "humidity"

Gain = 0.152

Step4 求 Gain for "windy"

Gain = 0.048

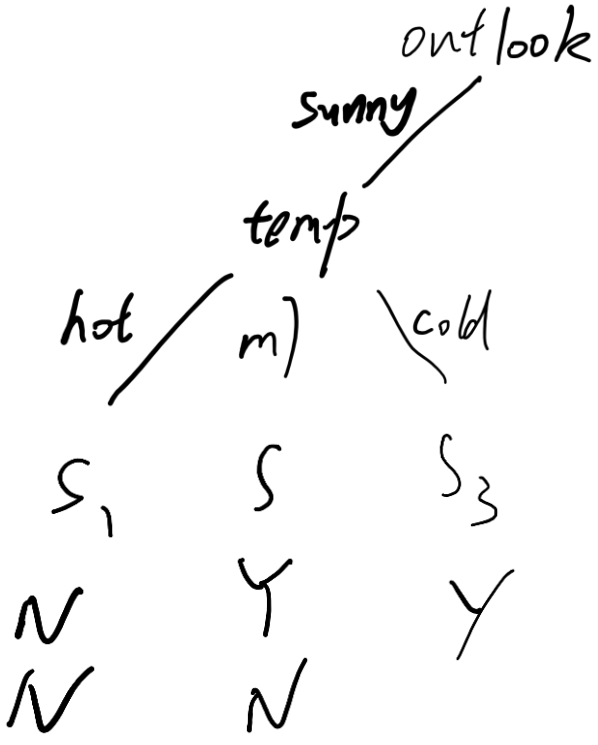
Step5 对比这些 attri 的 Gain，我们发现 outlook 的 Gain 最大，所以选它作为 root

Q2. 那么在 S_1 上我们应该选剩下3个 attr 中的哪个作为 node 呢?

we know $S = S_1 =$ Y Y N N N 有5个 Data

step1: 先从 "temp" 算起

a. $T_1 = H(S_1) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$



b $T_2 = \frac{2}{5}H(S_1) + \frac{2}{5}H(S_2) + \frac{1}{5}H(S_3)$

$H(S_1) = -\frac{0}{2}\log_2\frac{0}{2} - 1\log_2 1 = 0$

$$H(S_2) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

$$H(S_3) = 0$$

$$T_2 = \frac{2}{5}\times 0 + \frac{2}{5}\times 1 + \frac{1}{5}\times 0 = 0.4$$

Gain for Temp $= 0.971 - 0.4 = 0.571$

Step2 算 humidity

0.971

Step3 算 windy

0.020

Step4 对比发现 humidity is the largest

Ex 2：给出 $H(S)$，让我们算 Information Gain

Given is the following set of training examples:

	shape	color	class
1	circle	blue	+
2	circle	blue	+
3	square	blue	-
4	triangle	blue	-
5	square	red	+
6	square	blue	-
7	square	red	+
8	circle	red	+

You may use this table to calculate information gain:

x	y	-(x/y)* $\log_2$(x/y)	x	y	-(x/y)* $\log_2$(x/y	x	y	-(x/y)* $\log_2$(x/y	x	y	-(x/y)* $\log_2$(x/y
1	2	0.50	4	5	0.26	6	7	0.19	5	9	0.47
1	3	0.53	1	6	0.43	1	8	0.38	7	9	0.28
2	3	0.39	5	6	0.22	3	8	0.53	8	9	0.15
1	4	0.5	1	7	0.40	5	8	0.42	1	10	0.33
3	4	0.31	2	7	0.52	7	8	0.17	3	10	0.52
1	5	0.46	3	7	0.52	1	9	0.35	7	10	0.36
2	5	0.53	4	7	0.46	2	9	0.48	9	10	0.14
3	5	0.44	5	7	0.35	4	9	0.52			

a) What is the entropy of this collection of training examples?
b) What is the information gain of the attribute *shape*?

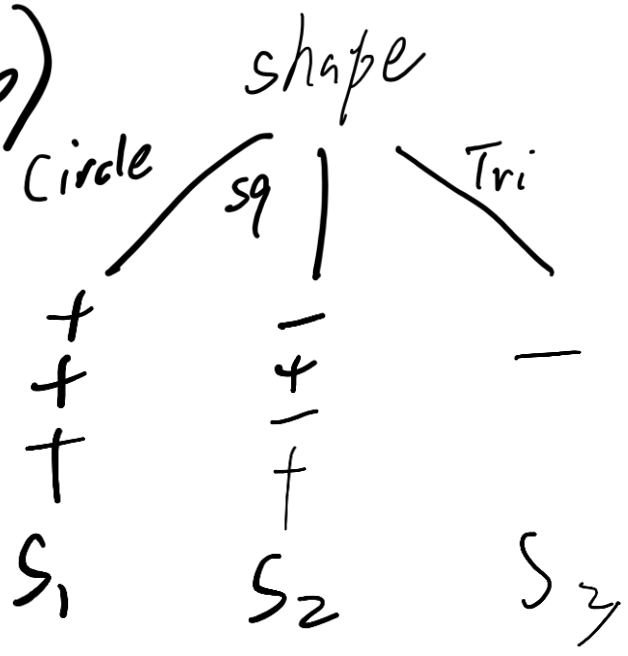
(a) $H(S) = I\left(\frac{5}{8}+, \frac{3}{8}-\right) = -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8}$

$= 0.42 + 0.53$

$= 0.95$

$T_1 = 0.95$

(b)



$T_2 = \frac{3}{8}H(S_1) + \frac{4}{8}H(S_2) + \frac{1}{8}H(S_3)$

$= 0 + \frac{4}{8}H(S_2) + 0$

因为是pure的

$$H_2(S) = I\left(\frac{2}{4}, \frac{2}{4}\right) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}$$
$$= 1$$

$$T_2 = \frac{4}{8} = \frac{1}{2}$$

$$\text{Gain} = 0.95 - \frac{1}{2} = 0.45$$

- Pruning Decision Tree — {avoid overfitting; Easier to visualize}
  - Pre-pruning: Stop growing the tree earlier
  - Post-pruning: fully grow the tree, then pruning

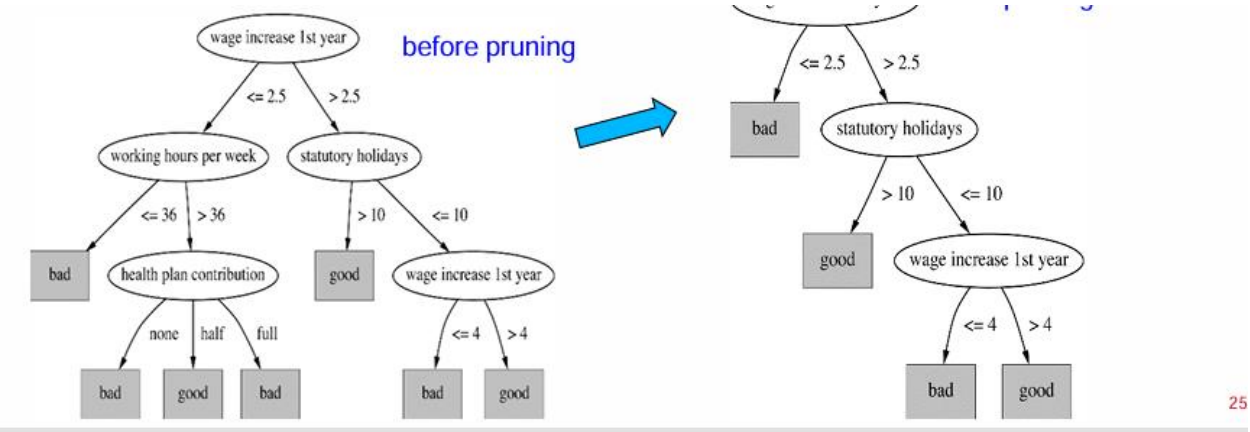
How much to prune?

用 validation set 来 decide

Method: sub-tree replacement

把分支简化成一个 leaf node.

其中多的作为 label



(2) Ensemble

Robust to overfitting

- Combine the pred of multiple base classifier
- If base classifier $\varepsilon < 0.5$ (error), then Ensemble is better
- If ... $\varepsilon > 0.5$, then we should not use Ensemble
- 只有在 base classifier independent 的时候, Ensemble 才有效. 否则 Ensemble 的 ε 等同于 avg(base classifier 的 ε of)

Bagging (Manipulate training set)

没有 loss function

Effective for unstable classifier as weak classifier

也叫 bootstrap

- 从原来的 Dataset 中 randomly choose data points 组成新的 Dataset，新 Dataset 的 data points 数量和原先的一样，这叫做 bootstrap sample

我们知道 at least 63% original data 会出现在新的 data 中

可能重复或不包含原先的一些 data point

- 假设我们要 Ensemble m 个 basic classifier，我们就要分别创 m 个 bootstrap sample 再用这些 data 作为 training data 分别创 m 个 basic classifier (使用相同的 Algorithm)

最后把这些 base classifier 的预测结果结合起来就是 Ensemble model

Boosting

- Key idea: next classifier (when we train next weak c) should be created using data points that

were difficult for the previous classifier. → weighted data set

一定里重点说 Adaboosting

↳ 主要针对二分法

Steps: 有 loss function: $L(h(x), y) = e^{-yh(x)}$

- Initialize $W_1(n) = \frac{1}{N}$ # 每个 data point have equal weight

For $t=1, t++, t<T$:

可以看到每次 traing set 权重不同 →

- Training a new weak classifier $h_t(x)$ by using $W_t(n)$
- Caculate $\epsilon_t = \sum W_t(n) \mathbb{I}[y_n \neq h_t(x_n)]$ # 错误率 ↳ 若返回1 == 返回0 → minimizing
- Calculate weight β_t of current weak classifier

$$\beta_t = \frac{1}{2} \log_e \frac{1-\epsilon_t}{\epsilon_t}$$

- **Update weight for data points**

$$W_{t+1}(n) \propto W_t(n) e^{-\beta_t y_n h_t(x_n)}$$

即先计算 $W_{t+1} = W_t(n) e^{-\beta_t y_n h_t(x_n)}$

再 normalize it 以达到 $\sum W_{t+1}(n) = 1$

Lastly, output final classifier

$$\hat{h}(x) = \text{Sign}\left[\sum_{t=1}^{T} \beta_t h_t(x)\right]$$

注意的点是：如果某些 data point 在之前的多轮分类中已被

1. 连续正确分类，那么即使当前轮分错了也不会导致 ε_t，因为它的 weight 很低

2. Greedily minimise the exponential loss function

It works because

evaluate model performance

3. we must optimize $\varepsilon_t = \sum w_t(n) \mathbb{I}[y_n = h_t(x_n)]$ to choose $h_t(x_n)$

4. and optimize by using $\beta_t^* = \frac{1}{2} \log \frac{1-\varepsilon_t}{\varepsilon_t}$ to minimize Loss function

Bagging VS Boosting

similarity:

1. both using vote for classification
   Avg for prediction
2. same type of weak learner

difference:

1. boosting have different weight
   bagging same weight
2. bagging separately generate weak learner

boosting iteratively

Random Forest

step: iteration

- Using Bagging generate a training data
- 使用 decision tree + random choose subset of features 创建一个 weak learner

Vote by majority weak learner output.

为了减少不同 decision tree 之间的 correlation

Attribute subset 越大，树之间的 correlation 越大，single 树的 Accuracy 越高

- 对比于 single decision tree, Random forest 的 disadv:
  - loss of interpretability
  - computationally expensive

Week 4 Tut

Week 4 Tutorial exercises
Naïve Bayes

Exercise 1. ***Naïve Bayes for data with nominal features (to do in class)***
Given is the following dataset where *loan default* is the class. Predict the class of the following new example using Naïve Bayes:
home owner = no, marital status = married, annual income=very high

	home owner	marital status	income	loan default
1	yes	single	very high	yes
2	no	married	high	yes
3	no	single	medium	no
4	yes	married	very high	no
5	yes	divorced	high	yes
6	no	married	low	no
7	yes	divorced	very high	no
8	no	single	high	yes
9	no	married	medium	no
10	no	single	low	yes

$P(H|E) = \frac{P(E_1|H)\cdots P(E|H)}{P(E)}$

H: Yes

$P(\text{home owner} = \text{no} \mid \text{Yes}) = \frac{3}{5}$ $P(\text{aI} = \text{vh} \mid \text{Yes}) = \frac{1}{5}$

$P(\text{ms} = \text{married} \mid \text{Yes}) = \frac{1}{5}$ $P(H) = \frac{5}{10}$

$P(\text{Yes}|E) = \frac{\frac{3}{5} \times \frac{1}{5} \times \frac{1}{5} \times \frac{5}{10}}{P(E)} = \frac{\frac{3}{250}}{P(E)}$

H: No

$P(\text{ho} = \text{no} \mid \text{No}) = \frac{3}{5}$ $P(\text{aI} = \text{vh} \mid \text{No}) = \frac{2}{5}$

$P(\text{ms} = \text{ma} \mid \text{No}) = \frac{3}{5}$ $P(H) = \frac{5}{10}$

$P(\text{No}|E) = \frac{\frac{3}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{5}{10}}{P(E)} = \frac{\frac{9}{125}}{P(E)}$

Since $P(\text{No}|E) > P(\text{Yes}|E)$ so predict should be no

Exercise 2. ***Naïve Bayes for data with numeric features (to do in class)***

The same task as in the previous exercise but now *annual income* is a numeric feature:

	home owner	marital status	income (in K)	loan default
1	yes	single	125	yes
2	no	married	100	yes
3	no	single	70	no
4	yes	married	120	no
5	yes	divorced	95	yes
6	no	married	60	no
7	yes	divorced	220	no
8	no	single	85	yes
9	no	married	75	no
10	no	single	90	yes

$$pdf = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(E_i-\bar{E})^2}{2\sigma^2}}$$

$$\mu = \frac{\Sigma E_i}{n} \qquad \sigma = \sqrt{\frac{(E_i-\bar{E})^2}{n-1}}$$

Use Naïve Bayes to predict the class of the following new example:

home owner = no, marital status = married, annual income=120

income: $\mu = \frac{125+100+95+85+90}{5} = 99$

$$\sigma = \sqrt{\frac{970}{4}} = 15.57 \qquad \sigma^2 = \frac{970}{4} = 242.5 \qquad 15.57^2$$

$$pdf = \frac{1}{\sqrt{2\pi}\cdot 15.57} e^{-\frac{(E_i-99)^2}{2\cdot 242.5}}$$

H: Yes

$P(H) = \frac{5}{10}$

$P(ho = no \mid Yes) = \frac{3}{5}$

$P(ms = m \mid Yes) = \frac{1}{5}$

$\frac{1}{\sqrt{2\pi}\cdot 15.57} e^{-\frac{(120-99)^2}{2\cdot 15.57^2}} = 0.010318$

$P(aI = 120 \mid Yes) = 2.8025920 58 \times 10^{-3}$

$P(Y \mid E) = \frac{\frac{1}{5}\cdot\frac{3}{5}\cdot 0.010318 \times \frac{5}{10}}{P(E)} = 1.68 \times 10^{-4}$

6.19×10^{-4}

H: No: $P(H) = \frac{5}{10}$

$P(h_0 = n_0 | N_0) = \frac{3}{5}$

$P(ms = m | N_0) = \frac{3}{5}$

$P(ai = 120 | N_0) = \frac{1}{\sqrt{2\pi}\,66.181} e^{-\frac{(E_i - 109)^2}{2 \cdot 66.181^2}}$

$5.945 \times 10^{-3} = 4.9452 \times 10^{-3}$

$\mu = \frac{70412046 + 220475}{5} = 109$

$P(N_0 | E) = \frac{\frac{3}{5} \cdot \frac{3}{5} \cdot 5.94 \times 10^{-3} \cdot \frac{5}{10}}{P(E)}$

$G = \sqrt{\frac{17520}{N \cdot 4}} = 66.181$

$= \frac{1.070 \times 10^{-3}}{P(E)}$

$G^2 = 66.181^2$

$P(No | E) > P(Yes | E)$ so No

Week 5 Tut

Exercise 1. ***Decision trees and information gain (parts a) and b) – done in class; the rest in your own time)***

Consider the following set of training examples:

	shape	color	class
1	circle	blue	+
2	circle	blue	+
3	square	blue	-
4	triangle	blue	-
5	square	red	+
6	square	blue	-
7	square	red	+
8	circle	red	+

Adapted from M. Kubat, Introduction to Machine Learning, Springer, 2021

a) What is the entropy of this collection of training examples with respect to the class?
b) What is the information gain of the attribute *shape*?
c) Which attribute will be selected as root of the tree based on information gain?
d) Build the whole decision tree. Draw the tree after each selected attribute.

You may use this table to calculate information gain:

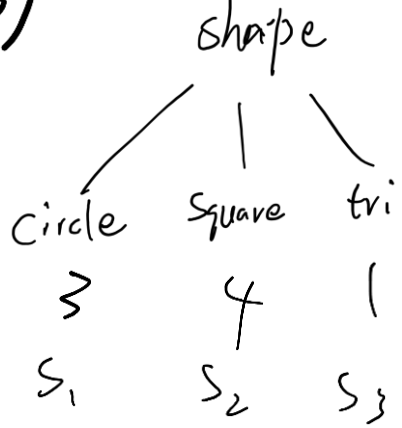
x	y	-(x/y)* $\log_2$(x/y)	x	y	-(x/y)* $\log_2$(x/y	x	y	-(x/y)* $\log_2$(x/y	x	y	-(x/y)* $\log_2$(x/y
1	2	0.50	4	5	0.26	6	7	0.19	5	9	0.47
1	3	0.53	1	6	0.43	1	8	0.38	7	9	0.28
2	3	0.39	5	6	0.22	3	8	0.53	8	9	0.15
1	4	0.50	1	7	0.40	5	8	0.42	1	10	0.33
3	4	0.31	2	7	0.52	7	8	0.17	3	10	0.52
1	5	0.46	3	7	0.52	1	9	0.35	7	10	0.36
2	5	0.53	4	7	0.46	2	9	0.48	9	10	0.14
3	5	0.44	5	7	0.35	4	9	0.52			

(a). 8 data in total { 5 +, 3 - }

$$\text{Entropy} = -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8}$$

$$= 0.95$$

(b)



$$L_1 = H(S) = 0.95$$

$$L_2 = \frac{3}{8}H(S_1) + \frac{4}{8}H(S_2) + \frac{1}{8}H(S_3)$$

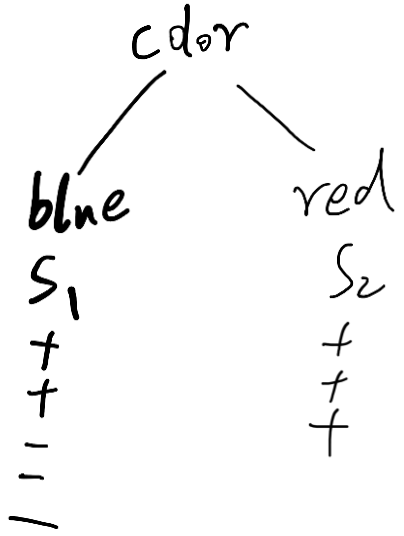
$$= \frac{3}{8}\times 0 + \frac{4}{8}\times\left[-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right] + \frac{1}{8}\times 0$$

$$= 0.5$$

+ + + − + − −

Information Gain = 0.95 × 0.5 = 0.45

(c) need calculate other Information Gain



$L_1 = 0.95$

$L_2 = \frac{5}{8} \times H(S_1) + \frac{3}{8} \times H(S_2)$

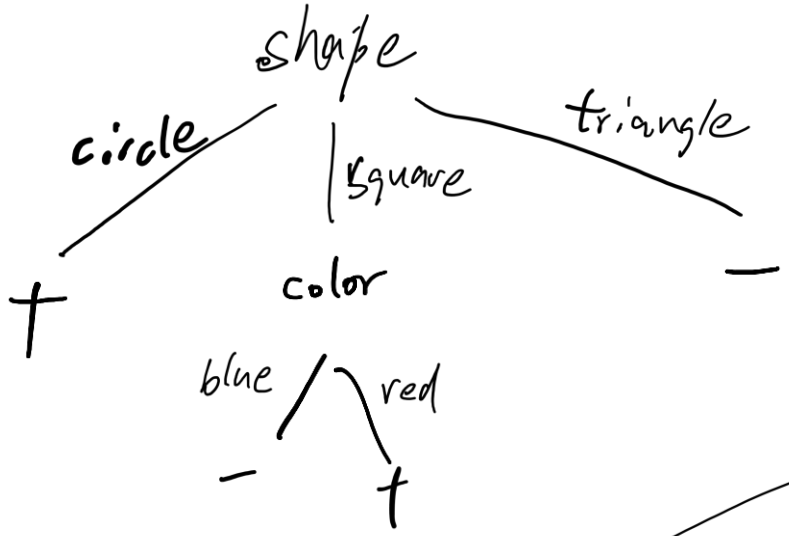
$= \frac{5}{8} \times \left[-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \right] + \frac{3}{8} \times 0$

$= \frac{5}{8} \times [0.53 + 0.44] = 0.60625$

Information Gain = 0.95 − 0.60625 = 0.34375 ≈ 0.34

Since 0.34 < 0.45 So choose ~~color~~ shape as root

(d)



overfitting → should reduce

underfitting → should increase

还有个 hyperpara, gamma, 大的表示小 width of margin, 小的表示大 width of margin

Week 6

有 Loss function: $L = f(w) = \frac{1}{2}||w^2|| + C\sum_{i=1}^{n}\xi_i$
↳ soft max

① Support Vector Machines SVM, ML, classification

- Support Vectors: data points from different classes which lie closest to the decision boundary; (lying on hyperplane)
  ↳ in the middle of margin
- Margin: the distance between support vectors

Goal

- Max Margin hyperplane: the hyperplane with the max margin
  → 要选最大的 margin hyperplane

- small Margin is more susceptible to overfitting
  Large Margin is more robust to minor changes in data

- How to get max margin?

$$\max d \Rightarrow \min \frac{1}{2}||w||^2 \text{ where } y_i(wx_i + b) \geq 1$$

↳ 即每个 data point 都要正确分类,
↳ 可以使用 soft margin 当不能被完全分类时

↙ 使用 Lagrange multiplier ↘

$$w_k = \sum_{i=1}^{N} \lambda_i y_i x_{ik}$$

↙ 对的第 k 个 feature

$\lambda_i > 0$ means data point is support vector

$\lambda_i = 0$ means data point is not

↓

N 是 support vector 数量

$$f = w\vec{x} + b = \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i \cdot \vec{x} + b$$

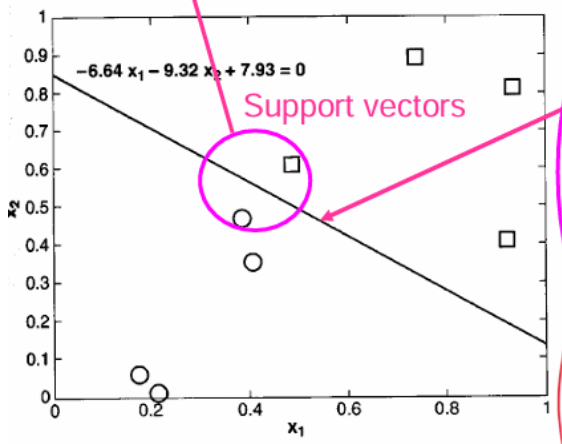
↳ dot product

只用对 support vector 的 W_i，最后结果就是 w

Linear Example

features x_1	features x_2	class y	Lagrange Multiplier
0.3858	0.4687	1	65.5261 λ_1
0.4871	0.611	−1	65.5261 λ_2
0.9218	0.4103	−1	0 λ_3
0.7382	0.8936	−1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	−1	0
0.2146	0.0099	1	0

x1, x2 — because they are support vectors



- 8 2-dim. training examples; 2 classes: -1,1
- After solving the problem with QP we find the λs
- Only 2 λs are non-zero (x1 & x2) and they correspond to the support vectors
- Using the λs , the weights (defining the decision boundary are):

$$w_1 = \sum_{i=1}^{2} \lambda_i y_i \mathbf{x}_{i1} = 65.5261(1*0.3858 - 1*0.4871) = -6.64$$

(x_1)

$$w_2 = \sum_{i=1}^{2} \lambda_i y_i \mathbf{x}_{i2} = 65.5261(1*0.4687 - 1*0.611) = -9.32$$

(x_2)

$b = 7.93$ //there is a formula for b (not shown)

- Classifying new examples:
  - above the decision boundary: class 1
  - below: class -1

Classifier = $-6.64x_1 - 9.32x_2 + 7.93$

19

Soft Margin: $C\sum\xi_i$ — solve overfitting

- 有些时候容忍一到二个 misclafify 会有更大的 margin
- Large C : more focus on minimizing training Error (more overfitting)

small C : ... on increase Margin (less overfitting)

② non-linear SVM

- 把 original data transform to high Dimensional space
- 在高维 space 把发现的 linear boundary 再映射回 original space 就会有 non-linear boundary

$$f = w \cdot z + b = \sum_{i=1}^{N} \lambda_i y_i \underline{k(x_i, z)} + b$$

↳ kernel trick

can helps us do the calculation in original space

kernel function 需要满足 Mercer's Theorem

比如 poly $(x \cdot y + 1)^p$

Radial $e^{\cdots}$

tangent $\tanh(\cdot\cdot)$

③ Dimensionality Reduction — PCA

Compression + Feature Extraction

- PCA is unsupervised
- removes redundant and highly correlated features and reduces noise

→ Principal Components z_i, new axes

z_i orthogonal with each other

$$Var(z_1) > Var(z_2) > \cdots > Var(z_m)$$

m is the # of features

Steps:

① find principal components $z_1, \cdots, z_m$

② to original data project to new axes $z_1, \cdots, z_k$

$k < m$

- Make sure $z_1, \cdots, z_k$ capture 95% of variance
- Elbow method

具体实现方法 SVD

$U_{n\times m}$

$\Lambda_{n\times m}$ $\Rightarrow$

$V_{m\times m}$

$U_{n\times k}$) use this 2 methods : the transformed data (projection)

$\Lambda_{k\times k}$: diagonal matrix

$V_{k\times k}$: define the new set of axes

$X_{n\times m} = U\Lambda V \Rightarrow X_{n\times k} = U\Lambda V$

↑ Here we eliminate weaker components (the one with lower variance)

— Compression ratio:

$$r = \frac{k(1+n+m)}{n\times m} = \frac{\text{after compression}}{\text{before compression}}$$

- 越小压缩越成功
- 小数表示压缩裁剪

Week 7：下面再都是 Deep learning

a general-purpose framework for representation learning

① Perceptron：Simplest NN

- Binary output：$f(wx^T+b) = \begin{cases} 1 & \text{if } w^Tx+b \geq 0 \\ 0 & \text{if } w^Tx+b < 0 \end{cases}$
- step function

$x_1 = 0.2 \quad x_2 = 0.3$
$w_1 = 2 \quad w_2 = -1.5$
$\Rightarrow f(-0.8) \Rightarrow 0$

- Learning rule：

$w^{new} = w^{old} + ex^T$

$b^{new} = b^{old} + e$

共 Element 只会是 0 和 ±1

e → Error = (target output − actual output)

- Steps：
  - Initialize w & b to random small value
  - While not reach stop condition：
    - Calculate a
    - Compute error $= t - a$
    - update w^{new} and b^{new}

  这里对于每个 data point 都要计算一次

  用 $data_{i+1}$ 的 w 从 $data_i$ 那边继承（继承 w_i^{new}）

  - check stop cond, such as no prediction error # check at the end of each epoch

End of epoch 1. Check if the stopping condition is satisfied:

1) All training examples are correctly classified?

current weight vector and bias:

w=[-0.7 -0.8 1.4]

b=-0.9

Ex.	input	output
1	1 0 0	0
2	1 0 1	1
3	1 1 0	0

Check Continue or not

Apply Ex.1 [1 0 0], t=1

a=step([-0.7 -0.8 1.4][1 0 0]-0.9)=step(-1.6)=0, correct

Apply Ex.2 [1 0 1], t=1

a=step([-0.7 -0.8 1.4][1 0 1]-0.9)=step(-0.2)=0, incorrect=> condition not satisfied

Stopping criterion is not satisfied (no need to check for Ex.3)

=> continue training

Start epoch 2:

training: …

End epoch 2: check stopping criterion

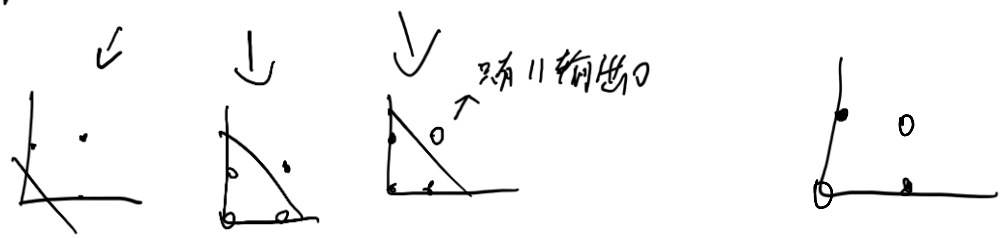
…

- Limitation :

  - Only guarantee when data is linearly separated to converge

Works for OR, AND, NAND Gate

not works for XOR



要在 input layer 先 flatten data

② MLP

有 error function：比如 MSE
有 overfitting：看 layers number

- 结构：1 input layer + multiple hidden layer + 1 output layer
- each neuron receive input only from the previous layer
- Fully connected → high training time
- Transfer function need to be differentiable, since need to calculate gradient

→ Back Propagation

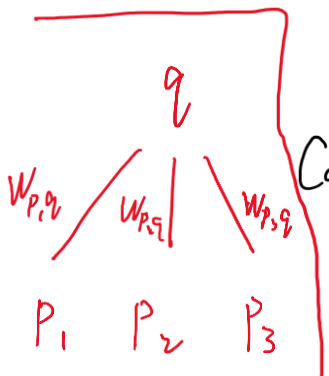
a. • 需要先正常 forward propagation 得到一个 real output
   • 再根据 real output 和 Expected output 的 Error 计算 δ for output neuron
   • 再一步步往前更新 weight，我这时就用再对比 output 了

- P_q：$Neuron_p \rightarrow neuron_q$
- W_{pq}：weight we use between p and q
- Σ_q：means the activation function for neuron q
- η：learning rate，gradient use it

b. $$W_{pq}^{new} = W_{pq}^{old} + \Delta W_{pq}$$

$$\Delta W_{pq} = \eta \cdot \delta_q \cdot O_p$$

这是每个 neuron 自己 output 完和这 output layer neuron 会直接输出结果而已
要用 input 和 output



Case 1：q is output layer neuron：
- 需要对比 {real output (true result t)} and {expected output (predicted result O)}

可以看到求 $W_{P,2}, W_{P,2}$ $W_{P,3}$ 都是用的同一个 δ_q，只有 $O_{P_1}, O_{P_2}, O_{P_3}$ 不同

$$\delta_q = (t_q - O_q) f'(z_q)$$

Case 2: q is Hidden layer neuron;

$$\delta_q = f'(z_q) \sum_i W_{qi} \delta_i$$

W_{qi} 是旧的 W_{qi} 而不是上面计算的新 W_{qi}

new δ_i

是对 q neuron 后的下一层 neuron 的 δ_i 求和

C.

$$bias_q^{new} = bias^{old} + \eta \times \delta_q$$

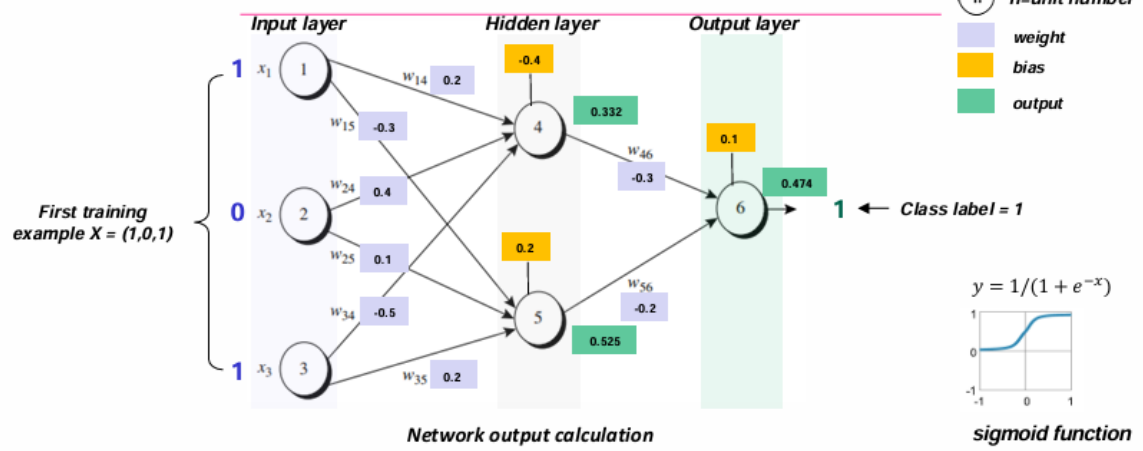
input layer neuron 没有 b，不用 update

每个 bias 也要同步 update

neuron 的

— Example

- Calculate the output of neurons 4 and 5, then the output of neuron 6



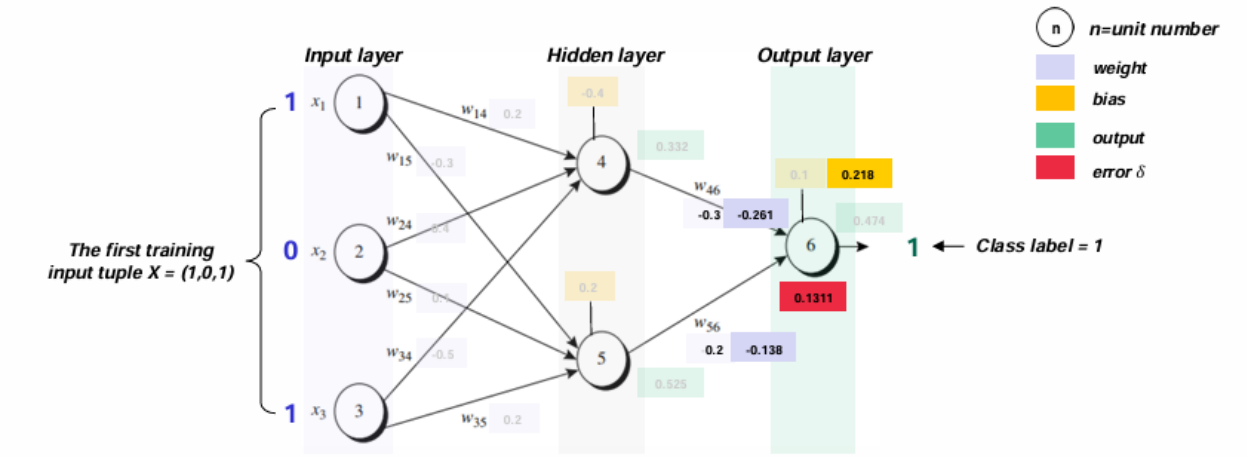
- Input to neuron 4: z_4=1*0.2+0*0.4+1*(-0.5)-0.4=-0.7, output of neuron 4: o_4=1/(1+$e^{0.7}$)=0.332
- Input to neuron 5: z_5=1*(-0.3)+0*0.1+1*0.2+0.2=0.1, output of neuron 5: o_5=1/(1+$e^{-0.1}$)=0.525 NN output
- Input to neuron 6: z_6=0.332*(-0.3)+0.525(-0.2) +0.1=-0.105, output of neuron 6: o_6=1/(1+$e^{0.105}$)= **0.474**

先根据 input layer neuron 的 output × correspond weight 得到 N_4 和 N_5 的 input

再根据 activation function (这里是 sigmoid) 计算 N_4 和 N_5 的 output

↳ 再用 N_4, N_5 的 output 计算 N_6 的 input

再 ....

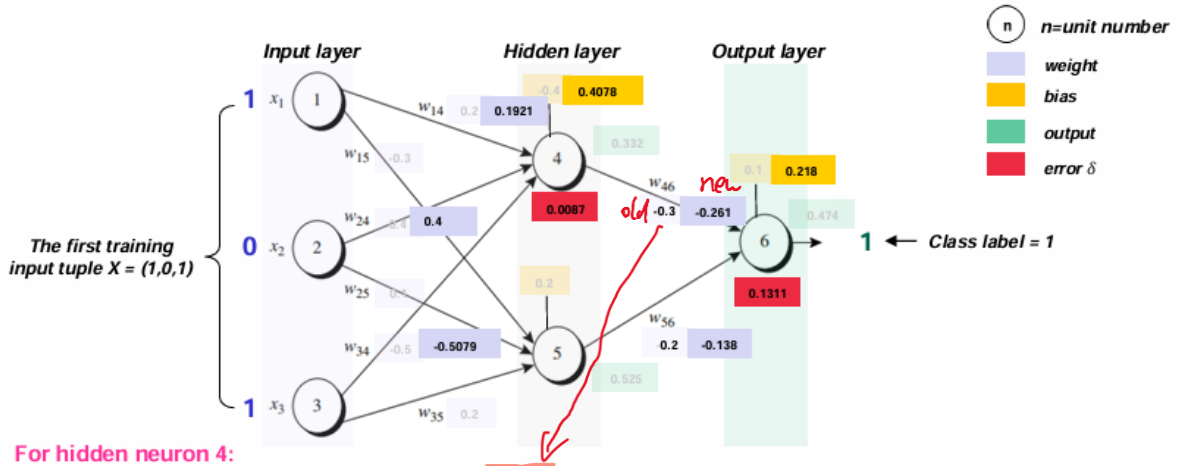


- $\delta_6 = (t_6-o_6)*o_6*(1-o_6)=(1-0.474)*0.474*(1-0.474)=0.1311$
- $\Delta w_{46}=\eta^* \delta_6*o_4= 0.9*0.1311*0.332=0.039$, $w_{46}new=w_{46}old+ \Delta w_{46}=-0.3+0.039=-0.261$
- $\Delta w_{56}=\eta^* \delta_6*o_5= 0.9*0.1311*0.525=0.0619$, $w_{56}new=w_{56}old+ \Delta w_{56}=-0.2+0.0619=-0.138$
- $\theta_6 new= \theta_6 old + \Delta\theta_6 = \theta_6 old +\eta^* \delta_6*1=0.1+0.9*0.1311*1=0.218$

① 先算 δ_q

② 再 update W_{iq}

③ 最后 update bias$_q$



For hidden neuron 4:

- $\delta_4=o_4*(1-o_4)*w_{46}*\delta_6=0.332*(1-0.332)*(-0.3)*0.1311=-0.0087$
- $\Delta w_{14}=\eta*\delta_4*o_1=0.9*(-0.0087)*1=-0.0079$, $w_{14}new=w_{14}old+\Delta w_{14}=0.2-0.0079=0.1921$
- $\Delta w_{24}=\eta*\delta_4*o_2=0.9*(-0.0087)*0=0$, $w_{24}new=w_{24}old+\Delta w_{24}=0.4+0=0.4$
- $\Delta w_{34}=\eta*\delta_4*o_3=0.9*(-0.0087)*1=-0.0079$, $w_{34}new=w_{34}old+\Delta w_{34}=-0.5-0.0079=-0.5079$
- $\theta_4new=\theta_4old+\Delta\theta_4=\theta_4old+\eta*\delta_4*1=-0.4+0.9*(-0.0087)*1=-0.4078$

可以看到，用的是 W_{46}^{old} 而不是 W_{46}^{new}

Gradient descent

review：之前在 linear regression 中说到 GD 是看所有的 Data point，再梯度下降，这里也一样，

GD: sum the error of all data points, then update weight

SGD: update the weight one-by-one

Mini batch: update the weight batch-by-batch

- One hot encoding — represent K-valued attributes with K binary numbers. Used in input layer

- Overfitting — # of hidden layer + Dropout rate

# of Hidden layer:
- Too many → overfitting
- Too few → underfitting

↳ we can gradually increase the # of layers

Dropout rate: randomly drop out neurons
- less dependent on certain neurons
- 只在 backpropagation 的階段 drop

- Learning rate η

time based

$$\eta_n = \frac{\eta_{n-1}}{1 + d \times n}$$

train step

exponential

$$\eta_n = \eta_0 e^{-dn}$$

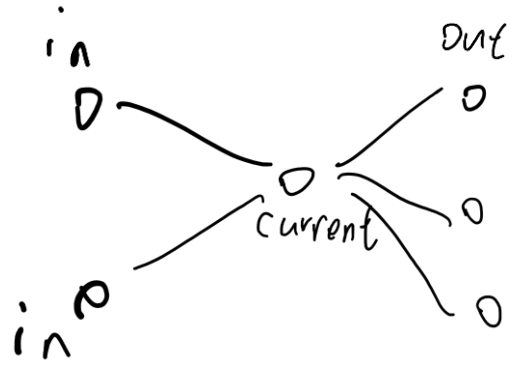
Ex. $\eta_0 = 0.3$, d: decay rate $= 0.5$,

$\eta_1 = 0.2$ $\eta_2 = 0.1$ ....

Weight initialization

Xavier 即 sampled from normal distribution,
st. centered at 0,

$$\sigma = \sqrt{\frac{2}{N_{in} + N_{out}}}$$



↓ # of input neuron : 2
# of output neuron : 3

Vanishing Gradient

啊那 sigmoid 等例，在

$$\delta = (t_q - o_q)\, o_q\, (1 - o_q)$$ 中不对准发现

当 this part 很小时，就会导到 weight 更新 slow, even does not update

→ 解决办法：用 Relu or LRelu

- **softmax for MLP**: interpret the outputs as the probabilities sum up to 1

formula: $\frac{e^{a_i}}{\sum e^{a_i}}$ #重点是等个e

- If initialize all biases and weights to 0, what will happen?

no matter how many neuron we have, they will act like one neuron per layer, since they are identical

Week 8

存 overfitting：看 pooling layer
也会用 backpropagation 更新 weight

① CNN

只用在 FC layer flatten data
有 loss function，用 optimizer 进行梯度下降

- Introduce new type of layers: convolutional and pooling
- Can recognize spatial structure 对 Image 即移动了的 Image 也能识别

Structure：

Input

Conv

RELU → non-linear activation function

Pool

Fully connected with soft max

Hyper para

optimizer：梯度下降的方法，比如 adam

Stride：滑动步长

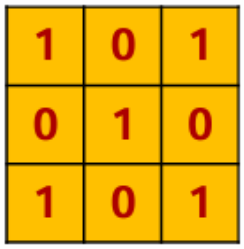
kernel size：size 对 filter window

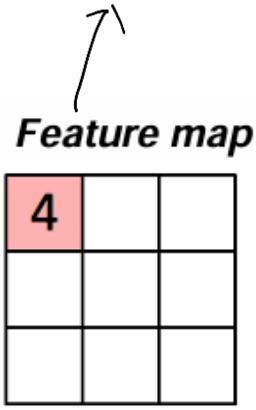
Padding：允许超长边界的大小，一般为 0

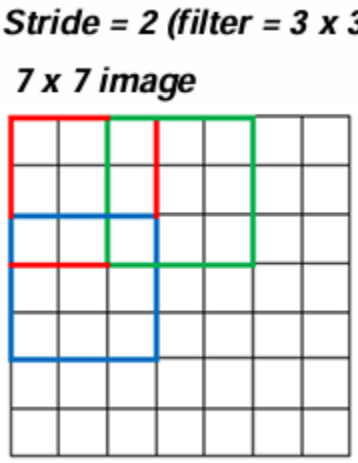
Receptive field

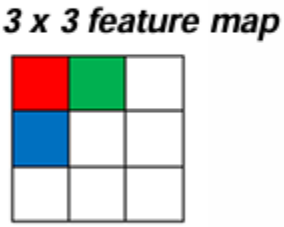
Feature map

learned during training, not pre-defined

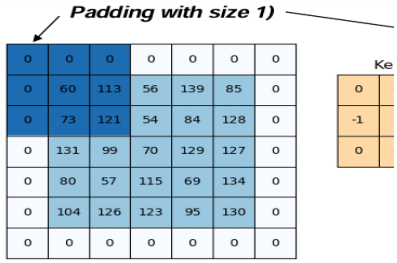








An example of padding with size 1



- **CONV layer** 捕获特征

  · The filter convolve with the image parts and "activate" when a specific feature is detected

  ↳ 在 feature map 中：某一个 Entry
  - have high value : feature detected
  - have low value : no feature detected

· 具体 step为 使用 filter 从左往右，从上往下，按照 stride 生成 feature map

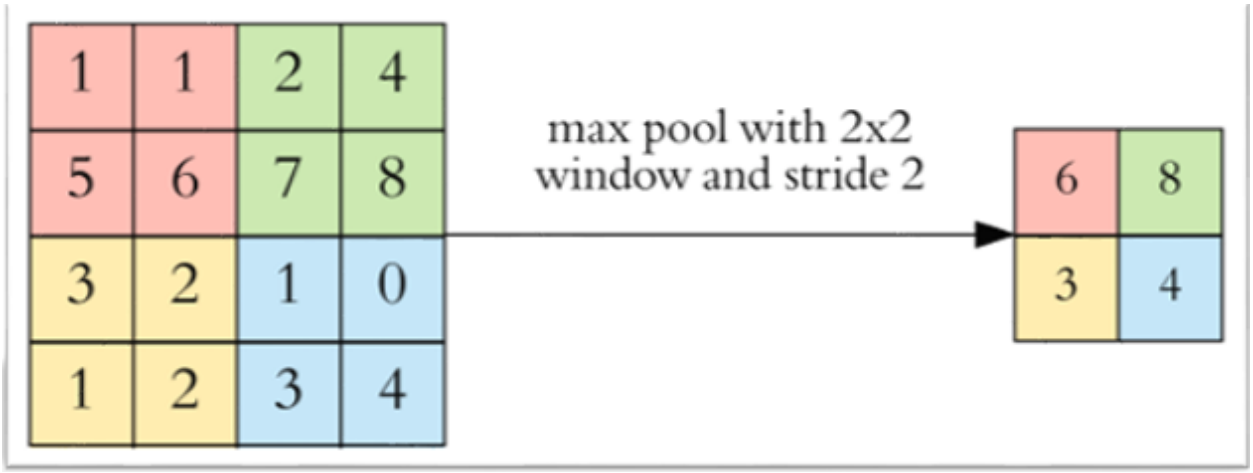
- **RELU**
  - no upper band
  - faster than tanh
  - Goal: simplifies the gradient descent calculation

- **Pooling**
  - Summarize the information from CONV layer
  - helps prevent overfitting

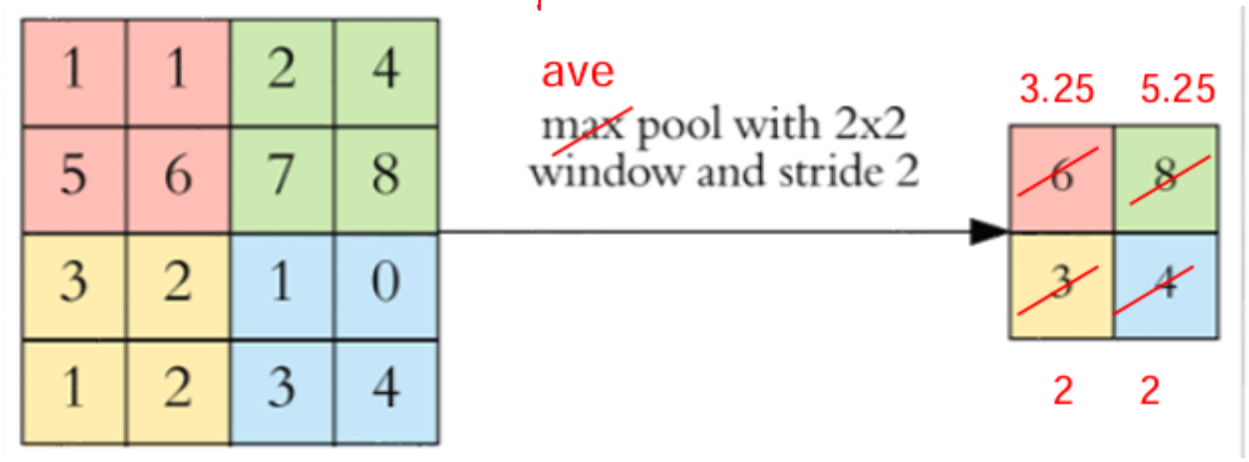
- Improve robustness to shifts in the receptive field

有两种：

a. Max MP



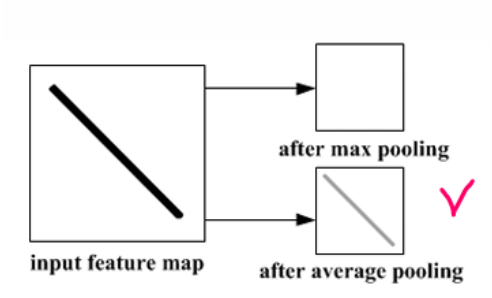
b. ave AP

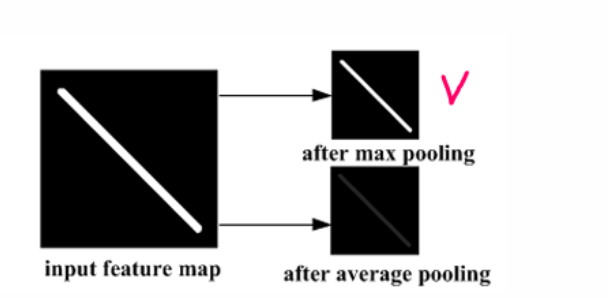


255是白色

MP: useful for images with dark background

AP: lighter background





~ **Fully connected layer** : 基本上是 MLP

- input : the first layer, need to "flatten" the output of pooling layer
- output layer : use soft max.

  ex. [0.1, 0.6, 0.3]

  表示該 Image 10% 是 0

  60% 是 1

  30% 是 2

~ **multi channel**

- detect different color
- new dim should be same for all filter

- Applications

LeNet-5 : Handwriting

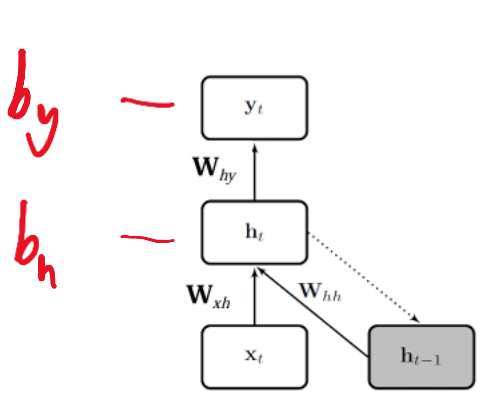
AlexNet : Image

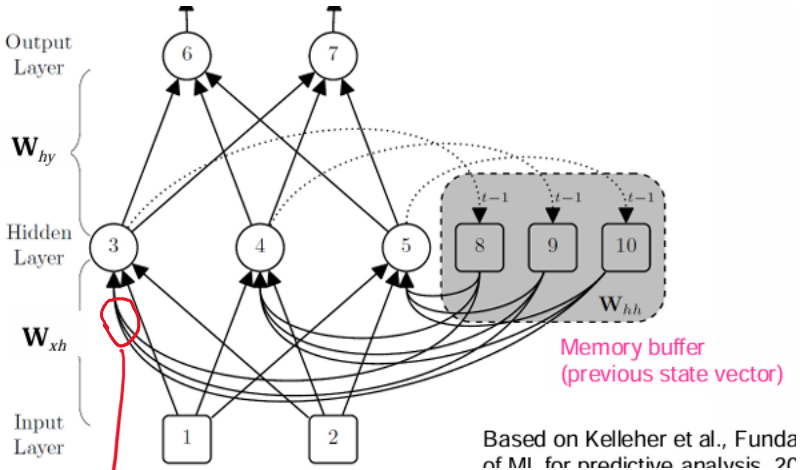
2 RNN

用 back propagation through time (BPTT)

2.1 Simple RNN

- For sequential data
- needs memory space since will check former info → long term dependency
- take 1 element from the sequence at each time
- A cyclic graph





我是这里 need refer to former info

- 一般用 tanh as transverse function:

$$h_t = \tanh\left(W_{hh} h_{t-1} + W_{xh} x_t + b_h\right)$$

$W_{hh} h_{t-1}$: old, $W_{xh} x_t$: Input, b_h: bias

$$y_t = W_{hy} h_t + b_y$$

- BPTT
  - 在整个 sequence predict 完后进行
  - 每个 t 都有对应的 ε_t，所以根据这个求 new weight W_{xy}, W_{xh}, W_{hh}

- RNN 用途

不适合：one to one

适合：one to many, many to one, many to many

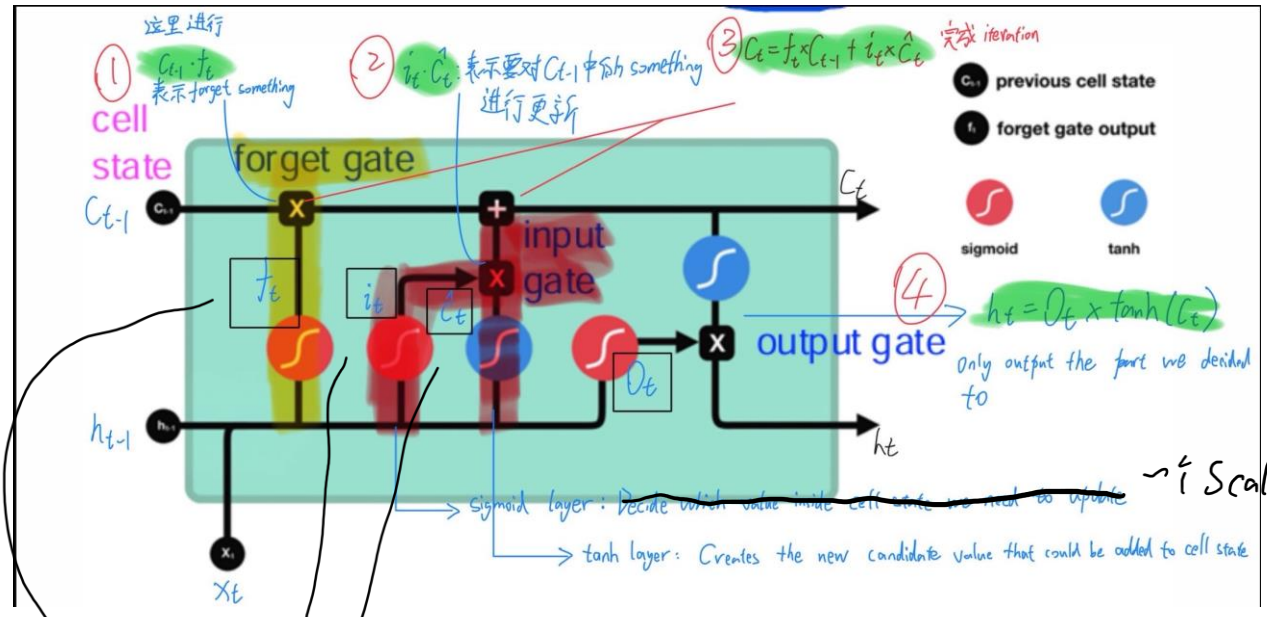
Limitation

a. 更容易产生 gradient vanishing 问题，因为 BPTT 链路一般更长

b. long distance memory 的效果不好，一用 LSTM

2 LSTM

- The model can learn what to store, what to throw, what to read from
- 不会有 Vanishing gradient problem, 因为 no repeated multiplication by W_{hh}.



~ i Scale 表明的重要性

$$\begin{cases} 6 \\ \text{or} \\ \tanh \end{cases} \left(W_? \cdot [h_{t-1}, x_t] + b_? \right)$$

Week 9

① Transformers

$O(n^2)$

Loss Func: Cross-Entropy

对比输出的 error

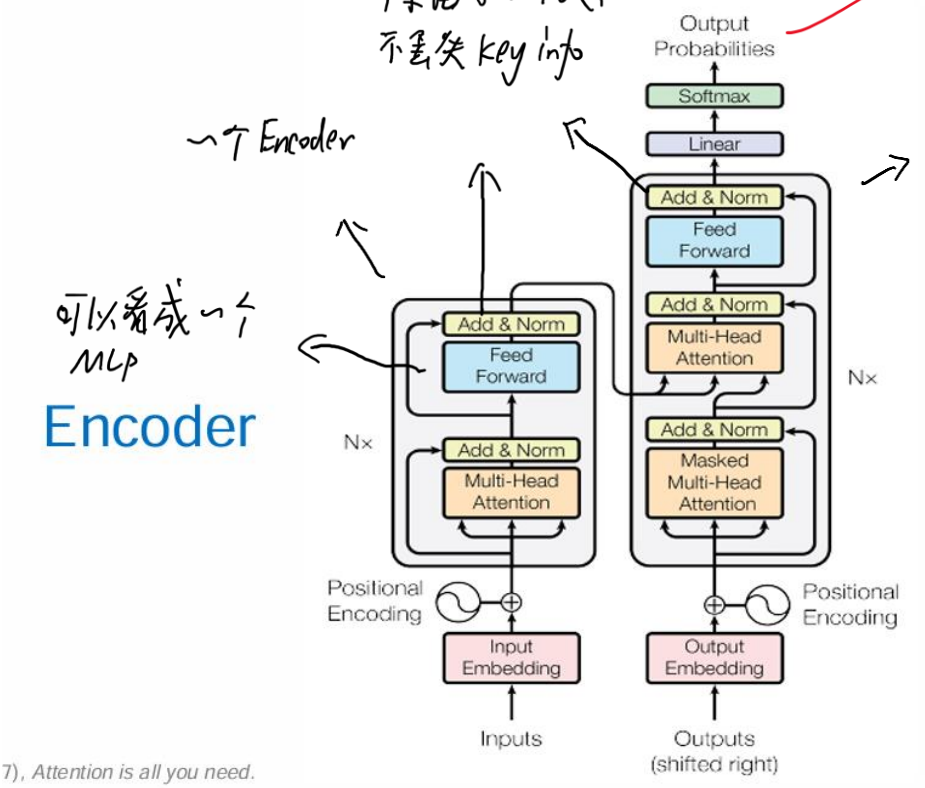
使用 optimizer

保留原始信息
不丢失 key info

一个 Encoder

一个 Decoder

可以看成一个 MLP



7), Attention is all you need.

- Decoder 和 Encoder 数量一样

Encoder

- Embedding
  - Vector size is the longest sentence in training set
  - 与 one-hot 相比，可以有效降低维度，
  - 在 Embedding space，具有相似含义的单词通常靠的更近

multi head attention

- understand a word according to its formal sentence

- Query vector Q
  Key vector K
  Value vector V
  $\Rightarrow$ $Q = XW_Q$, $K = XW_K$, $V = XW_V$

  W_Q, W_K, W_V 使用 Xavier initialize
  使用 backpropagation update

- output of self-attention layer is Z

$$Z = \mathrm{Softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \cdot V$$

- $Q \times K^T$: draw down irrelevant words
- $\sqrt{d_k}$: $\sqrt{}$ the longest vector length in embedding layer

上面都是一头，多头就是

同步执行 n 次 self-attention 计算生成，每次的 Q K V 都不同，否则达不到多头的效果

W^0 × $[Z_0 | Z_1 | \cdots]$

其中 W^o 是被 train 出来的，

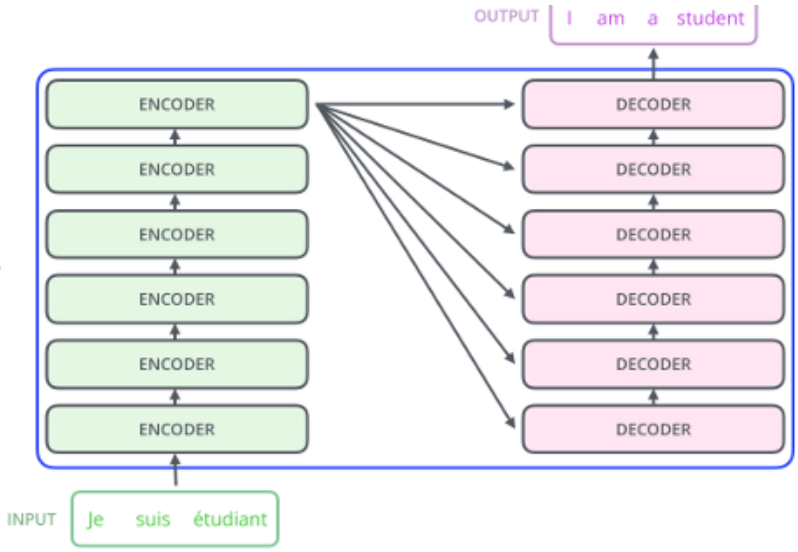
一开始也是 Xavier initial

再使用 backpropagation

Decoder

- **masked self-attention**
  - not allow to attend to earlier positions in the output sequence (不已经来的)
  - 从 Decoder embedding 获得 "X"

- **Encoder-Decoder Attention layer**



不仅从之前 Decoder 获取信息，还从总的 Encoder 获得信息

Softmax at output

- 每个 output token 都有 prob，sum 起来是 1
- 选中 argmax(token) 作为下一个可能的 token

优缺点

Down: · Computational cost is high

Benefits: · easier to parallize
· can be more deeper } 比 Down 更好

Week9 Tuj

- Encoder-only: convert text to rich numerical representation
- Decoder-only: auto-generate rest part of a sentence.
  ↳ GPT 常用

current cluster centroid

Week 10

① Clustering

$$BSS = \sum_{i=1}^{k} \sum_{i=1} |K_i| \, dist(c, c_i)$$

all points

Centroid

$$SSE = \sum_{i=1}^{k} \sum_{x \in K_i} dist(x, c_i)^2$$

- Similar to other within the cluster (High Cohesion)
- disimilar to other in other clusters (High separation)

- Centroid: Typical not an actual data point

$$C = \frac{\sum_{i=1} p_i}{N}$$

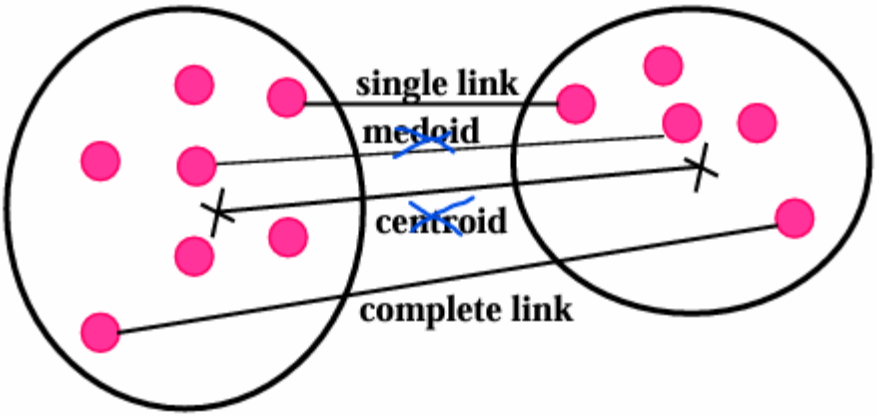
- Medoid: the centrally located data point in the cluster

- Distance between clusters:

  Single link (MIN): The smallest pairwise distance between different clusters

  Complete link (MAX): The largest pairwise distance between different clusters

  Avg link : The average pairwise distance . . .



# of data point
# of iter
$O(n\ k\ i\ d)$
# of clusters
# of attri

可以在 Data preprocessing 的步驟 remove outliers

2 K-means

- require k to be specified → the number of clusters
- Works well if data is spherical, equal density, equal size and are well separated (i.e. no outliers)

— Steps

Select K points as the initial centroids

Repeat

Form K clusters by assigning all points to the closest centroid

Recompute the centroid of each cluster

Until the centroids does not change

- 计算题

Task: Use the K-means algorithm to cluster them into 2 clusters. The initial centroids are A and B. Show the clusters after the first epoch.

	A	B	C	D	E
A	0	2	7	10	1
B	2	0	3	4	6
C	7	3	0	5	9
D	10	4	5	0	1
E	1	6	9	1	0

A and B are centroids

$D(A,C)=7$ ✗ $D(B,C)=3$ ✓

$D(A,D)=10$ ✗ $D(B,D)=4$ ✓

$D(A,E)=1$ ✓ $D(B,E)=6$ ✗

So clusters should be

$\{A, E\}$ $\{B, C, D\}$

- Properties

- Sensitive to initial centroids
- Most of convergence happens in the first few epoches

- How to select good initial centroids?

Method 1: run k-means several times, pick the one with the smallest SSE

$$SSE = \sum_{i=1}^{K} \sum_{x \in K_i} d(c_i, x)^2$$

Centroids of cluster k_i

data points within cluster k_i

Method 2: K-means++

- We incrementally select centroids until K centroids have been selected
- selects points that are ~~far~~ most away from the current centroids
- 可能会选到 outliers

② GMM (Gaussian Mixture Model)

没 time complexity

- A probalistic clustering
- It assume each data within each cluster is normal distributed

- steps

① Initialization, randomly set μ_i and σ_i, $\theta_i = (\mu_i, \sigma_i)$

总共有几个 cluster, 就设几个 θ_i

② repeat

②.1 Expectation step:

calculate prob that each object belongs

to each distribution by using

$$P(\text{distribution}_i \mid x_i, \theta_i) = \frac{w_i P(x_i|\theta_i)}{\sum w_i P(x_i|\theta_i)}$$

这里 $\sum w_i = 1$

要用 pdf 算，x_i 是 data point，因为知道 μ 和 σ，所以可以算

E.g. for point $x_i = 0$: $P(x_i|\theta_1) = 0.12$, $P(x_i|\theta_2) = 0.06$ (calculated using the probability density function of normal distribution)

$$\Rightarrow P(distribution\ 1|x = 0, \theta) = \frac{0.12}{0.12+0.06} = 0.66$$

$$P(distribution\ 2|x = 0, \theta) = \frac{0.06}{0.12+0.06} = 0.33$$

(2.2) Maximum Step:

- recalculate μ_i for each cluster
- σ_i does not change
- We use weighted avg points where weights are the p that the point assign to cluster i

(3) Until μ_i does not change

- Properties

  - more flexible ⇒ allow ecliptical clusters
    # 和 K-means 的不同点

3 Hierarchical Clustering

Time: $O(n^3)$
Space: $O(n^2)$
} 缺点...

- Easy to visualize → dendrogram
  - sensitive to noise, outliers
- Nested structure clusters
- A desired number of clusters can be obtained by "cutting" the dendrogram at different level
- 2种方法
  - Agglomerative (bottom-up) merge
  - Divisive (Top-down) split from one huge cluster

- Agglomerative

Steps:

Compute the proximity matrix

let each data point be a cluster

Repeat

Merge the 2 closest clusters

Update the proximity matrix

Until only a single cluster remain

使用描述的 distance function, such as Euclidean

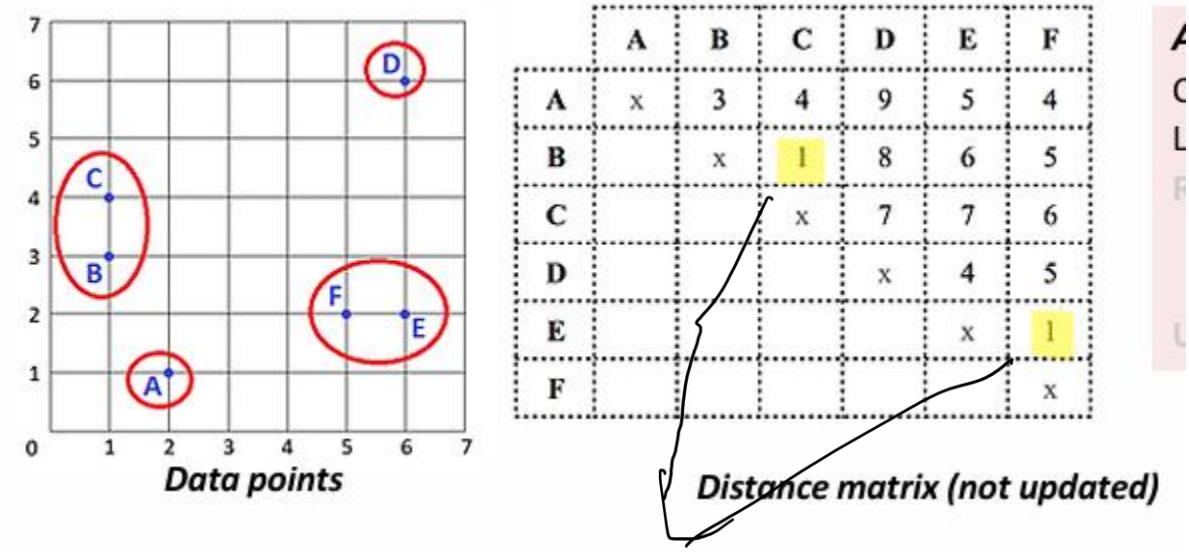
可以使用

MAX,

Min,

avg,

Ward

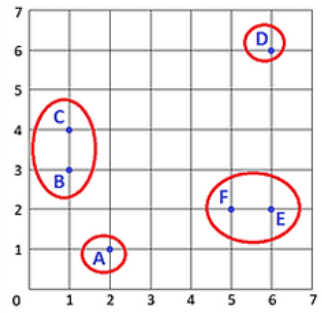


	A	B	C	D	E	F
A	x	3	4	9	5	4
B		x	1	8	6	5
C			x	7	7	6
D				x	4	5
E					x	1
F						x

pick the 2 smallest one

这里是用 Min

- Now we can update the distance matrix using the single-link distance



Data points

	A	B, C	D	E, F
A	x	3	9	4
B, C		x	7	5
D			x	4
E, F				x

Distance matrix (updated)

Algorithm:
Compute the distance matrix
Let each data point be a cluster
Repeat
Merge the two closest clusters
Update the proximity matrix
Until only a single cluster remains

Divisive method

- breaking the link correspond to the largest distance

4 DBSCAN

不能处理：密度差异较大的 data
不善长 high dimentional data
time: $O(n^2)$ (low dim will have $O(n\log n)$)
space: $O(n)$

- Density based spatial clustering of Application with Noise
- Can find clusters with arbitrary and complex shape such as S-shape

会根据 density 分 cluster

Terminology

- Neighborhood — Eps:
  the area within a radius Eps

  Eps: too small: all points will be core
  big: all points will be noise

- Density threshold — MinPts:
  the minimum number of points in the Eps

- 3 types of point: within Eps

  Core points:
  neighbour point ≥ MinPts

  border points:
  not a core point and falls within neighbour of a core point

  Noise points:
  两者都不是的点

Steps

① label points as core, border, or noise

② Discard noise points

③ cluster remain points:

- 2 core points within Eps of each other are put in the same cluster
- border points and correspondent core point are put in the same cluster
- Tie> need to be solved when border point belongs to more than one core point

- How to select Eps and MinPts?

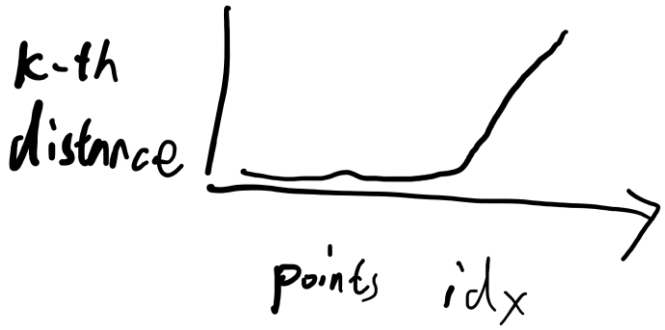
- 使用 k-dist
- 即 Examine the distance from a point to its k-th nearest neighbour

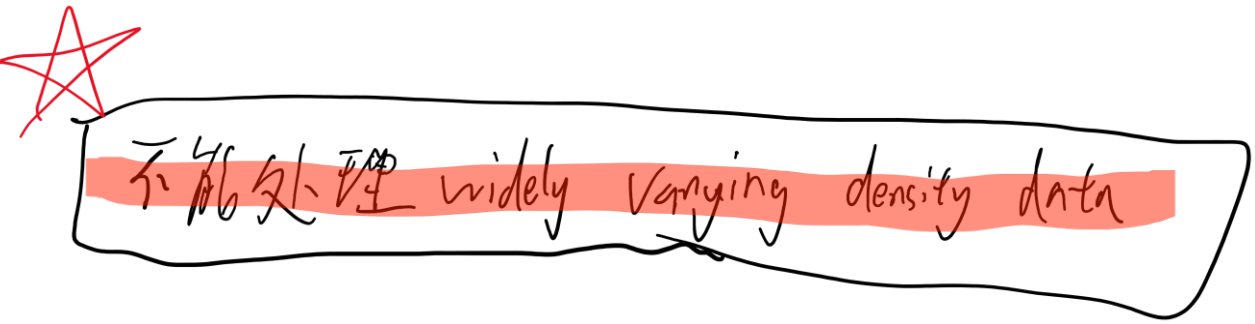
idea: 如果 K < the size of a cluster, then the distance should be very small

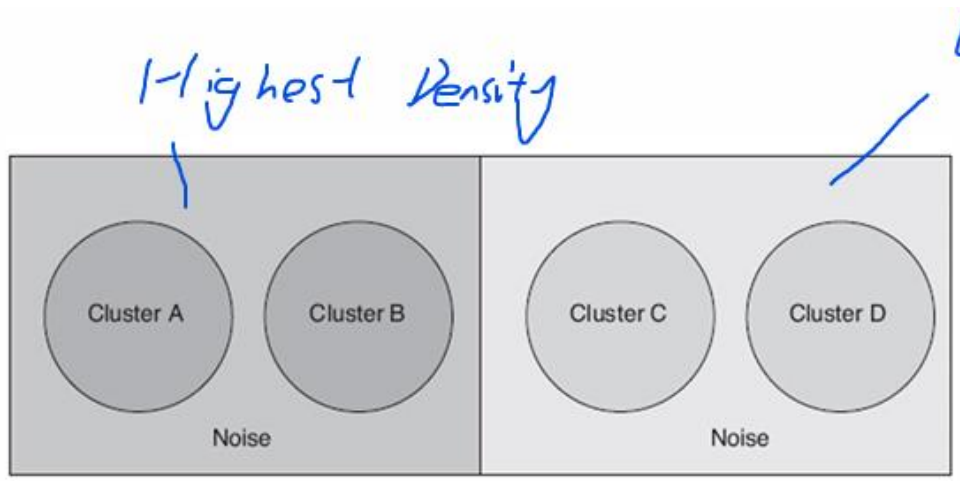
- We can plot the graph for all points

with the distance to its k-th nearest neighbour

- If there is a suddenly increase, then we can use k as MinPts and Eps







- 如果 Eps 不够大，就行成 C 和 D，因为它们的密度小。
- 如果 Eps 够大，又会导致 A 和 B 被视为一个 cluster

⑤ Evaluate clustering

主要讲 internal — Unsupervised measures

a. silhouette ☆

close to +1 or -1 表示好

b. corr → 求 M_1 和 M_2 之间的 corr

用 $Entry\_sim = \frac{1-(d_i - d_{min})}{d_{max} - d_{min}}$ 求 Entry 的值

1 data point [1 2 3 ; 1 ; 2 ; 3]

实际上每个 data point 在不在一个 cluster，在的话为 1 不在为 0

[1 2 3 ; 1 0 1 ; 2 1 ; 3 1]

C. Visual

一 重点谈谈 Silhouette for a point

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

avg

b_i: distance from x to all points in other cluster

a_i: avg distance from x to all points in current cluster

[-1, 1]

1 → 正确分到 cluster

0 → 在 cluster 的 boundary

-1 → 错误分类

the higher, the better

lower is bad bcs cohesion > separation

- For a cluster:

$$\text{avg}(s_1, s_2, \ldots) \quad s_i \in k_n$$

- For a clustering

$$\text{avg}(k_1, k_2 \ldots)$$

Tut 10

Exercise 1. DBSCAN clustering

Use the DBSCAN algorithm to cluster the items A1, A2, …, A8. The distance matrix is given below. Assume that Eps=2 and MinPts=2.

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	5	6	3.6	7	7.2	8	2.2
A2		0	6.1	4.2	5	4.1	3.2	4.5
A3			0	5	1.5	1.5	7.5	6.5
A4				0	3.6	4.1	7.2	1.5
A5					0	1.4	6.7	5
A6						0	5.4	5.5
A7							0	7.5
A8								0

在table中先找出符合条件的点

① label points

A_1 $\{A_1\}$ A_2 $\{A_2\}$ A_3 $\{A_3, A_5, A_6\}$

A_4 $\{A_4, A_8\}$ A_5 $\{A_5, A_6, A_3\}$ A_6 $\{A_6, A_3, A_5\}$

A_7 $\{A_7\}$ A_8 $\{A_8, A_4\}$

A_1, A_2, A_7 are noise

A_3 and A_5 are core and A_4 A_6

So they will merge, finally, we get a large cluster $\{A_3, A_5, A_6\}$ $\frac{1}{13}$

and a small cluster $\{A_4, A_8\}$

Exercise 2. ***Evaluating clustering quality using the silhouette coefficient***

Given are 4 items P1, P2, P3 and P4. They were clustered using a clustering algorithm. The cluster labels and the distance matrix are shown below. Evaluate the quality of the clustering by computing the silhouette coefficient for each point, each of the 2 clusters and the overall clustering.

Distance matrix:

	P1	P2	P3	P4
P1	0	0.1	0.65	0.55
P2	0.1	0	0.7	0.6
P3	0.65	0.7	0	0.3
P4	0.55	0.6	0.3	0

Cluster labels:

point	cluster label
P1	1
P2	1
P3	2
P4	2

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

bigger is better

bcs $b_i > a_i$ is o/c

$b_i < a_i$ means bad

① For each point

P_1: $a_1 = 0.1$ $\quad b_1 = \frac{0.65+0.55}{2} = 0.6$

$$S_1 = \frac{b_1 - a_1}{\max(a_1, b_1)} = \frac{0.6-0.1}{0.6} = \frac{5}{6}$$

P_2: $a_2 = 0.1$ $\quad b_2 = \frac{0.7+0.6}{2} = 0.65$

$$S_2 = \frac{0.65-0.1}{0.65} = \frac{11}{13}$$

P_3: $a_3 = 0.3$ $\quad b_3 = \frac{0.65+0.7}{2} = \frac{27}{40}$

$$S_3 = \frac{\frac{27}{40} - 0.3}{0.675} = \frac{5}{9}$$

P_4: $a_4 = 0.3$ $\quad b_4 = \frac{0.55+0.6}{2} = \frac{23}{40} = 0.575$

$$S_4 = \frac{0.575 - 0.3}{0.575} = \frac{11}{23}$$

② For cluster

K1: $\frac{\frac{5}{6} + \frac{11}{13}}{2} = \frac{131}{78}$ 忘记除2了

$= 0.8397$

K2: $\frac{\frac{5}{9} + \frac{11}{23}}{2} = \frac{214}{207}$ 忘记除2了

$= 0.5169$

③ For clustering

$0.5169 + 0.8397$

$$\frac{\frac{131}{78} + \frac{214}{207}}{2} \simeq 1.366518$$

0.678

relative good because close to 1

***Exercise 3.* Evaluating clustering quality using correlation**

For the data from the previous exercise, evaluate the clustering quality using the correlation between the similarity matrix derived from the distance matrix (given below) and the similarity matrix derived from the clustering results (i.e. the matrix whose ij entry is 1 if two objects belong to the same cluster and 0 otherwise).

The similarity matrix derived from the distance matrix is given below. It was computed from the distance matrix as $s = 1 - (d - dmin)/(dmax - dmin)$, where *dmin* and *dmax* are the minimum and maximum distances in the matrix: *dmin*=0.1 and *dmax*=9).

Similarity matrix: M_1

	P1	P2	P3	P4
P1	1	1	0.08	0.25
P2		1	0	0.17
P3			1	0.67
P4				1

M_2

	P_1	P_2	P_3	P_4
P_1	1	1	0	0
P_2		1	0	0
P_3			1	1
P_4				1

- $Corr(M_1, M_2) = \frac{A \cdot B}{\|A\| \times \|B\|}$ this is $\cos(A, B)$

- $M_1 \rightarrow [1, 0.08, 0.25, 0, 0.17, 0.67] = x$

$M_2 \rightarrow [1, 0, 0, 0, 0, 1] = y$

$$\mu_x = \frac{1+0.08+0.25+0+0.17+0.67}{6} = \frac{217}{600} = 0.36$$

$$\mu_y = \frac{2}{6} = \frac{1}{3} \simeq 0.33$$

$$\frac{71}{75} \simeq \frac{(1-\frac{217}{600})(0.08-\frac{217}{600})(0.25-\frac{217}{600})(0-\frac{217}{600})(0.17-\frac{217}{600})(0.67-\frac{217}{600})}{(1-\frac{1}{3})(0-\frac{1}{3})(0-\frac{1}{3})(0-\frac{1}{3})(0-\frac{1}{3})(1-\frac{1}{3})}$$

$$0.873 \sqrt{(1-\frac{217}{600})^2+(0.08-\frac{217}{600})^2+(0.25-\frac{217}{600})^2+(-\frac{217}{600})^2+(0.17-\frac{217}{600})^2+(0.67-\frac{217}{600})^2}$$

$$\frac{2\sqrt{3}}{3} \times \sqrt{2\times(\frac{2}{3})^2+4\times(-\frac{1}{3})^2}$$

$$\simeq 0.939$$

since it close to 1 so

x and y are highly positive corr

which means good clustering

Week 11

① Markov Model

- the p of any state only depends on the previous state

$$P(\pi_i | \pi_1, \cdots, \pi_{i-1}) = P(\pi_i | \pi_{i-1})$$

- Predict sequence of data

$$P(\pi_1, \cdots, \pi_{1C}) = P(\pi_1)(P\pi_{12}|\pi_1)(P\pi_3|\pi_2)\cdots$$

↗ from A_0

② HMM

2.1 HMM P1, what is the probability of given ob sequence?

- T : observation sequence length

N: the number of states

- Enumeration $\geq TN^T$

不好 因为 T 一般很大

- Forwarding algorithm N^2T

- a_{jk}: 从 A 或 A_0 获取 表示从 $\pi_j \to \pi_k$ 的 P

即 $a_{jk} = P(\pi_k | \pi_j)$

- $e_k(x_i)$: 从 E 获取 $e_k(x_i) = P(x_i | \pi_k)$

- x_i: observation at time i
- j, k: state number

① Initialization: 对所有的 state k 都要求

$$f_k(1) = A_0(k) e_k(x_1)$$

② Iteration: 这一步 cause $O(N^2)$, because double itera

$$f_k(i) = e_k(x_i) \sum_j f_j(i-1) a_{jk}$$

↓

上一层的所有 j

③ After iter → Termination

$$P(X) = \sum_K f_K(m)$$

2.2 Viterbi — $O(NT)$

- It is efficient because we did not do inner loop, we use max(...) instead of Σ

- Given observation sequence X, what's the most likely hidden state sequence P_{tr}

① Initialization

$$V_K(1) = A_0(K)\, e_K(x_1)$$

② Iteration

$$V_k(i) = e_k(x_i) \max_j V_j(i-1)\, a_{jk}$$

$$Ptr_k(i) = \operatorname{argmax}_j V_j(i-1)\, a_{jk}$$

③ Termination

$$\pi_m = \operatorname{argmax}_k V_k(m)$$

$$\pi_{i-1} = Ptr_k(i) \quad \text{where } i = m, \dots, 2$$

$$\pi_1 \rightarrow \cdots \rightarrow \pi_m$$

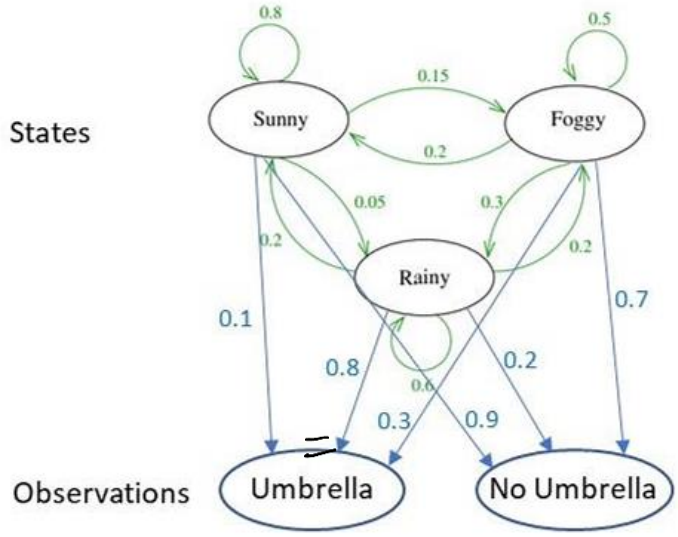
Week 11 Intro

Need to memorize formula

***Exercise 1.* Hidden Markov Models**

You were locked in a room for several days, and you were asked about the weather outside. The only piece of evidence you had is whether Alicia, who comes into the room to bring your meals, is carrying an umbrella or not.

The following Hidden Markov Model models the situation. The initial state probabilities are: A_0(Sunny)=0.4, A_0(Rainy)=0.3 and A_0(Foggy)=0.3.



Green is A

Blue is E

X_1 = ~~Um~~ No

X_2 = ~~No~~ Um

Suppose that on the first day you observed that Alicia had no umbrella, then on the second day Alicia carried an umbrella. What is the most likely sequence of hidden states?

HMM 2

Step 1: Initialization

(a) Rainy

$$f_{Rainy}(1) = a_0(Rainy)\, e_{Rainy}(X_1)$$

$$= 0.3 \times 0.8 = 0.24$$

(b) Sunny

$$f_{Sunny}(1) = a_0(Sunny)\, e_{Sunny}(X_1)$$

$$= 0.4 \times 0.1 = 0.04$$

(c) Foggy

$$f_{Foggy}(1) = a_0(Foggy)\, e_{Foggy}(x_1)$$

$$= 0.3 \times 0.3 = 0.09$$

Rainy

Step 2: $i = 2$

(a) Rainy

$$f_{Rainy}(2) = e_{Rainy}(x_2)\, Max\left\{ f_{Rainy}(1) \times a_{rainy\ rainy},\ f_{Sunny}(1) \times a_{Sunny\ rainy},\ f_{foggy}(1) \times a_{foggy\ rainy} \right\}$$

$$= 0.2 \times max\left\{ \underline{0.24 \times 0.6},\ 0.04 \times 0.05,\ 0.09 \times 0.3 \right\}$$

$$= 0.2 \times 0.24 \times 0.6$$

$$= 0.0288$$

$Ptr_{rainy}(2) = Rainy$

(b) Sunny

$$f_{Sunny}(2) = e_{Sunny}(x_2)\ max\left\{ f_{Sunny}(1)\, a_{Sunny\ Sunny},\ f_{Rainy}(1)\, a_{rainy\ Sunny},\ f_{foggy}(1)\, a_{foggy\ sunny} \right\}$$

$$= 0.9 \times \max\{0.04 \times 0.8, 0.24 \times 0.2, 0.09 \times 0.2\}$$

$$= 0.0432$$

$Ptr_{sunny}(2) = Rainy$

(c) Foggy

$$f_{Foggy}(2) = e_{Foggy}(X_2) \cdot \max\{f_{foggy}(1)\, a_{foggyfoggy}, f_{rainy}(1)\, a_{rainyfoggy}, f_{sunny}(1)\, a_{sunnyfoggy}\}$$

$$= 0.7 \times \max\{0.09 \times 0.5, 0.24 \times 0.2, 0.04 \times 0.15\}$$

$$= 0.7 \times 0.24 \times 0.2 = 0.0336$$

$Ptr_{foggy}(2) = Rainy$

$Ptr(2) = Rainy$ 应该每个 Hidden state 都求

Step 3

$$\pi_2 = \arg\max[f_S(2), f_R(2), f_F(2)]$$

$$= \text{Sunny}$$

$\pi_1 = Ptr_{sunny}(2) = rainy$

上面(b)因为 sequence 看错了，所以需要整个重做

Step 1 :

(a) Rainy: $V_R(1) = A_0(R)\, e_R(x_1)$

$= 0.3 \times 0.2 = 0.06$

(b) Sunny $V_S(1) = A_0(S)\, e_S(x_1) = 0.4 \times 0.9 = 0.36$

(c) Foggy $V_F(1) = A_0(F)\, e_F(x_1) = 0.3 \times 0.7 = 0.21$

Step 2. Iteration, only loop once

$i = 2$

(a) Rainy

$V_R(2) = e_R(x_2) \cdot \max\{V_R(1) a_{RR}, V_S(1) a_{SR}, V_F(1) a_{FR}\}$

$= 0.8 \times \max\{0.06 \times 0.6, 0.36 \times 0.05, 0.21 \times 0.3\}$

$= 0.8 \times 0.21 \times 0.3 = 0.0504$

$Ptr_R(2) = Foggy$

(b) Sunny

$$V_S(2) = e_S(x_2) \max\{V_S(1)a_{SS}, V_R(1)a_{RS}, V_F(1)a_{FS}\}$$

$$= 0.1 \times \max\{0.36 \times 0.8, \dots\}$$

$$= 0.1 \times 0.36 \times 0.8 = 0.0288$$

$$Ptr_S(2) = \text{Sunny}$$

(c) Foggy

$$V_F(2) = e_F(x_2) \cdot \max\{\dots\}$$

$$= 0.3 \times 0.2 \times 0.5 = 0.015$$

$$Ptr_F(2) = \text{Foggy}$$

Step 3 :

$$\pi_2 = \text{Argmax}\{V_F(2), V_S(2), V_R(2)\}$$

$$= \text{Rainy}$$

$$\pi_1 = Ptr_R(2) = F$$

Week 12

General framework for decision-making

① Reinforcement learning

- Goal: Obtain π^* through training
  - indicate the best step at each state

- S: set of possible state

  A: set of possible actions

  $R(S,A)$: distribution of reward of S, A

  $P(S_t | S_{t-1}, A_{t-1})$: 是 Markov, transition P. 只跟上一个 state 有关

γ: discount factor, $[0,1]$,

- high: focus on future
- low: focus on current

- **State Value Function** $V^{\pi}(s)$

$$V^{\pi}(s) = E\left[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, \pi\right]$$

↳ sum all future reward with corresponding discount factor

- It used to evaluate current state

~ **Action-Value Function** $Q^{\pi}(s,a)$

The expected cumulative reward from taking action a in state s and then following the policy.

$$Q^{\pi}(s,a) = E\left[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right]$$

↓

指定 initial 的 action 为 a_0

By definition:

- $Q^*(s,a) = \max_{\pi} E\left[\sum_{t\geq 0} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right]$

- 给出 the max Expected cumulative reward from a given pair

Bellmen equation 给出 suggestion 关于如何计算

$Q^*(s,a)$ = 给出reward + next state cummulative reward

$$= E_{s'\sim\varepsilon}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a\right]$$

$\gamma^0 = 1$

next state Action-Value function

如果已经找到可以求 $Q^*(s,a)$

- **Q-learning**

① calculate

$Q(s,a)$ before taking action a

② After taking action a, we have s' and r

$$Q^*(s',a') = R(s,a,s') + \max_{a'} \gamma Q(s',a')$$

$$\Delta Q(s,a) = \underbrace{R(s,a,s') + \max_{a'} \gamma Q(s',a')}_{\text{new}} - \underbrace{Q(s,a)}_{\text{current}}$$

↳ we want this become 0,

since it means we find the best action

- **Deep Q-Learning**

, 不需要传入 action，只用传入 state

- Recall

$$Q^*(s,a) = E\left\{ r + \gamma \max_{a'} Q(s',a') \mid s,a \right\}$$

- we have loss function

$$L = \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]^2$$

Target — current

- Then using SGD to minimize L

~ Divergence Issue

① correlations between samples

since data are time related,

∴ so data are related

∴ It will lead use repeated data and lead to diverge

Solutions:

using Experience replay

Sample experience uniformly from data set to remove correlation

② Non Stationary targets

· Cause by the change of $Q_{(s,a)}$

· lead to unstable update direction

solution: Target Network

stable the target for few steps

- Double Q-Learning

eliminate upward bias

by using

current Q to select action

older Q to evaluate action