

Week 2 Int

Exercise 1. Nearest Neighbor

The dataset below consists of 4 examples described with 3 numeric features (a_1 , a_2 and a_3); 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: $a_1=2$, $a_2=4$, $a_3=2$?

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

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

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

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

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

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

3-knn will use $d_1, d_2, d_3 \rightarrow$

yes	yes	no	Vote by majority
-----	-----	----	------------------

 → yes

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:

$age \leq 30$, $income = medium$, $student = yes$, $credit_rating = fair$

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

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

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

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

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

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

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

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

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

1 NN: use 7 \rightarrow No

3NN: use 1, 4, 7 \rightarrow Yes \rightarrow No
No

Week 3 有 Loss Function $L = f(w) = \sum_{i=1}^n (\underbrace{w^T x^i - y^i}_{\hat{y}})^2 + \lambda \sum_{j=1}^m w_j = \|Xw - Y\|_F^2 + \lambda \|w\|^2$

① Linear Regression 注意 $X_0 = 1$ real data model check
 Predict $\hat{Y} = \underbrace{W_0}_{k \times 1} + W_1 X_1 + \dots + W_n X_n = \underbrace{X^T}_{n \times k} \underbrace{W}_{k \times 1} = W^T X$ for overfitting

→ 然后怎么求上面的 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 Mathematic Property good

→ How to minimize square Loss Function?

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

a. Least square regression: 无法 avoid overfitting

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

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

我们要用 matrix form 来 minimize this Equation

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

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

$$X^T X w - X^T Y = 0 \Rightarrow W = (X^T X)^{-1} X^T Y$$

↳ 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 Regulation Term avoid overfitting

$$f(w) = \min ||Xw - Y||^2 + \lambda ||w||^2$$

$$w = (X^T X + \lambda I_k)^{-1} X^T Y$$

closed form solution

large λ : underfit

Small λ : overfit

$$L_2 \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

c. **Gradient descent** : 有梯度上面 time complexity 太大导致

time complexity: $O(nk)$ per iteration

space complexity: $O(k)$

sum read to n

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

step size

negative slope

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

too small: converge slowly

too big: overshoot, can diverge

$$\hookrightarrow \alpha_i = \frac{\alpha}{\sqrt{n}} \leftarrow \text{hyperparameter}$$

of data iteration

- SGD: $O(k)$ time complexity

- 和 GD 一样，它只用 one observation, 但不看所有 sum

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

$$= W_i - \alpha_i \nabla_j (w_i) \text{ with } j \text{ sampled at random}$$

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

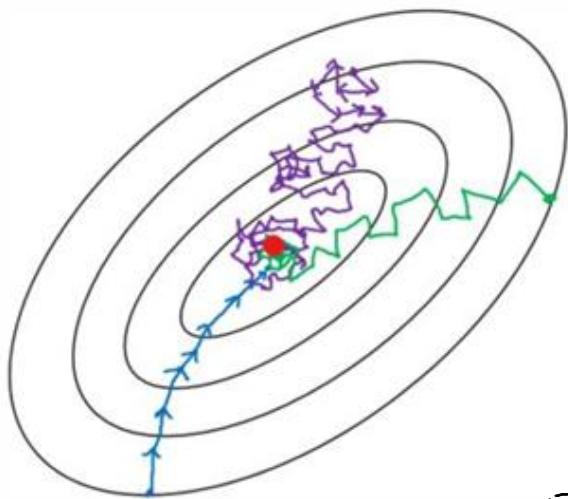
- Mini-batch GD Time complexity: $O(bk)$ per iter

和 SGD 一样, 我们不仅只用一个, 而是用一个 batch

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

- Pro: 1. 通过 adjust batch size, 可以在 computation per iter # 上 need more

- Con: 1. need to consider new hyperparameter batch size



- Batch gradient descent
- Mini-batch gradient Descent
- Stochastic gradient descent

cross Entropy

$$\text{Loss Func } L = f(w) = -\frac{1}{n} \sum_{i=1}^n [y_i \log G(w^T x_i + b) + (1-y_i) \log [1-G(w^T x_i + b)]]$$

② Logistic Regression

· 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 $(0, 1)$

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

b 是为了让 decision boundary 可以不通过原点，从而增加灵活性

· 可以设置 threshold = 0.5, 这样当 $P(y|x) > 0.5 \Rightarrow \hat{y} = 1$

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

又因为 $G(0) = 0.5$

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

- $W_{i+1} = W_i - \alpha_t \nabla f(w)$ ←
↓ This is Gradient of Cross Entropy Loss function
- 经过 n iter, fix it to get w^*

- 在给出 w^* and x^* , find $P(y=1|x)$ %

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 (Arthur walter) via email claims2155@yahoo.com.hk or call +44 704 575 1113

Keywords are [lottery, prize, office, email]

The given weight vector is $w = [0.3, 0.3, -0.1, -0.04]^\top$

What is the probability that the email is spam?

$$x = [1, 1, 1, 2]^\top$$

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

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

Week 4. 没有 Loss Function

① Naive Bayes (ML, classification, supervised)

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

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

- 假设 H 有两种 YES 和 NO, 我们 need 分别求

$P(\text{YES}|E)$ 和 $P(\text{NO}|E)$ 并对比它们的大小。

- 使用公式的时候, 分子部分可以从 Training data 中获得
分子 $P(E)$ 则不用求, 因为在对比不同的 $P(H|E)$ 时会互相 Cancel

Problem :

a. ZERO-FREQUENCY (某个 $P(E_i|H) = 0$)

使用 Laplace Correction $\rightarrow P^0$

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

只对为 0 的那个 term
使用

\rightarrow # of possible values
for E_i

b. Missing Values

如果 \mathbf{x} 中不包含某一个 E_i 怎么办？

我们可以在公式中也不包含

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

上面就是不包含 E_i 的解决方法

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

$$\rightarrow P(H|E_i) = \frac{1}{6\sqrt{2\pi}} e^{-\frac{(E_i - \mu)^2}{2\sigma^2}}$$

只用 E_i
我们
不需要
别的 E_j

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

· 一般来说 std dev 和 mean 会给, 我们要做的就是区分不同的 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 Attr

b. 如果 numeric Attr 不是 normal distri, 我们可以使用其它的 pdf

Examples:



Solution (2)

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

$$P(\text{yes})=5/10$$

$$\begin{aligned} P(E_1|\text{yes}) &= P(\text{home owner=no|yes}) = 3/5 \\ P(E_2|\text{yes}) &= P(\text{marital status=married|yes}) = 1/5 \\ P(E_3|\text{yes}) &= P(\text{annual income=very high|yes}) = 1/5 \end{aligned}$$

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

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

$P(\text{no}|E) > P(\text{yes}|E) \Rightarrow$ Naïve Bayes predicts
loan default = no for the new example

$$P(\text{no})=5/10$$

$$\begin{aligned} P(E_1|\text{no}) &= P(\text{home owner=no|no}) = 3/5 \\ P(E_2|\text{no}) &= P(\text{marital status=married|no}) = 3/5 \\ P(E_3|\text{no}) &= P(\text{annual income=very high|no}) = 2/5 \end{aligned}$$

	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

这里就是只有 Nominal attributes 的 case

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

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

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

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

class yes income	class no income
125	70
100	120
95	60
85	220
90	75
$\mu_{\text{income_yes}}=99$	$\mu_{\text{income_no}}=109$
$\sigma_{\text{income_yes}}=15.57$	$\sigma_{\text{income_no}}=66.18$

这里 Assume X 中包含了 \sim numeric E_i : income, 其中 $E_i = 120$. 完成这个后, 我们还要和 E_i 做一些衍生的 $P(H|E) = \frac{P(E_i|H) P(\text{Income}=120|H) \cdots P(H)}{P(E)}$

② Evaluate ML (以 Accuracy)

- 使用 test data set

— Holdout Method

分为 training set

Validation set

Testing set

可以多次

build the model

tune hyperparameter

evaluate accuracy

使用重采样

方法 repeated

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 ~ 平等

- **Cross Validation** : Usually 10-folds
↳ 很多 experiments show this is the best

- 把 original data set 分成 10 份
- 每次用 9 份 build model，用剩下的 1 份 test it
- 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 train one final model

$$\cdot \text{1-fold Hyper} = \left\{ \begin{array}{l} x: [1, 2] \\ y: [3, 4] \\ z: [5, 6] \end{array} \right\}$$

我们要做 10-fold, R. 的 Build 次数就是 $2 \times 2 \times 2 \times 10 = 80$ 次

GridSearch CV

④ Performance Measurement

- Confusion Matrix:

Not a performance measure,
just allow us to calculate performance measures

		Predict class 1	Predict class 2
Real T	True pos	fn	
Real F	fp	True neg	

$$\text{Accuracy} = \frac{\bar{T}_P + T_n}{f_P + f_n + \bar{T}_P + T_n}$$

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

			<i>predict b</i>	<i>real c</i>
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 =
 $(50+44+47)/(50+0+0+44+6+0+3+47)$
 $=141/150 = 94\%$

$$P = \frac{tp}{tp + fp}$$

$$R = \frac{tp}{tp + fn}$$

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

Tut:

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

- best cross-validation score:
we get from $\text{avg}(\text{acc}_1, \dots, \text{acc}_{10})$
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

KNN, logistic Regression, \hat{y} 的 model 更容易 visualize, 虽且是
non-linear fit



Week 5

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

没有 loss function

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

- 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 分割 最后, leaf node 只有 一种 class 是最好的

→ 使用 Information gain + Entropy 来判断

★ (1) 中的 best attr: $\# \text{category} \log_2 0 = 0$

best attribute is the one with highest information gain

Entropy 公式为: S 是当前 node 分割的数据

$$H(S) = - \sum_i p_i \log_2 p_i$$

class \hookrightarrow $\frac{\# \text{of sample which is class i}}{\# \text{of all samples}}$

Non-leaf node 都要用这个

Information Gain 公式:

$$\text{Gain} = \overline{T_1} - \overline{T_2}$$

Entropy =

Entropy < 0

→ 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↑ YES

5↑ NO

14↑ data

Q1 想知道选哪个 attribute 做为当前 Node.

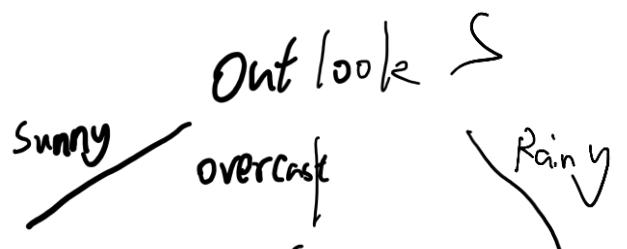
Step1: 先从 outlook 看起

a. 找 $\overline{T_1}$

$$T_1 = H(S) = I\left(\frac{9}{14}, \frac{5}{14}\right) = 0.940 \text{ bits}$$

(通过table) 5↑ data

S₁
Y
N
N
N



S₂
Y
Y
Y
Y

S₃
Y
Y
Y
N
N

b. \bar{T}_1

$$T_2 = H(S | \text{outlook}) = \frac{5}{14} H(S_1) + \frac{4}{14} H(S_2) + \frac{5}{14} H(S_3)$$

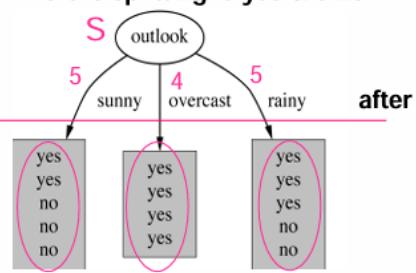
$$H(S | \text{outlook} = \text{sunny}) = I\left(\frac{2}{5}, \frac{3}{5}\right) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971 \text{ bits}$$

$$H(S | \text{outlook} = \text{overcast}) = I\left(\frac{4}{4}, \frac{0}{4}\right) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0 \text{ bits}$$

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

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

Before splitting: 9 yes & 5 no



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

Gain for "outlook" = $\bar{T}_1 - \bar{T}_2 = 0.94 - 0.693 = 0.247$

Step 2 \nexists Gain for "temp"

$$\text{Gain} = 0.029$$

Step 3 \nexists Gain for "humidity"

$$\text{Gain} = 0.152$$

Step 4 \nexists Gain for "windy"

$$\text{Gain} = 0.048$$

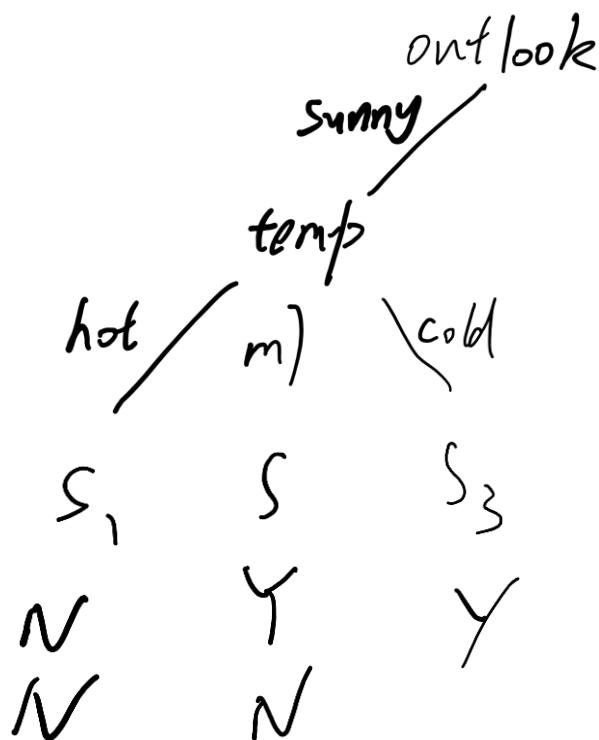
Step 5 对于 每个 attri 的 Gain, 找出最大的 outlook 的 Gain 最大, 所以把它作为 Root

Q2. 那么在 S_1 上我们应该选择剩下3个 attr 中的哪个作为 node 0/6?

we know $S = S_1 = \begin{cases} Y \\ N \\ N \\ N \end{cases}$ 有 5 个 Data

step1 : 先从 "temp" 算起

$$a. \bar{T}_1 = H(S_1) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$$



$$b. \bar{T}_2 = \frac{2}{5}H(S_1) + \frac{2}{5}\bar{T}_1(S_2) + \frac{1}{5}\bar{T}_1(S_3)$$

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

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

$$H(S_3) = 0$$

$$T_2 = \frac{2}{5}x_0 + \frac{2}{5}x_1 + \frac{1}{5}x_0 = 0.4$$

$$\text{Gain for Temp} = 0.971 - 0.4 = 0.571$$

Step 2 for humidity

$$0.971$$

Step 3 for windy

$$0.020$$

Step 4 可以發現 humidity 是 the largest

Ex 2: 給出 $H(S)$, 計算各個 feature 的 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	$-(x/y) \cdot \log_2(x/y)$	x	y	$-(x/y) \cdot \log_2(x/y)$	x	y	$-(x/y) \cdot \log_2(x/y)$	x	y	$-(x/y) \cdot \log_2(x/y)$
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			

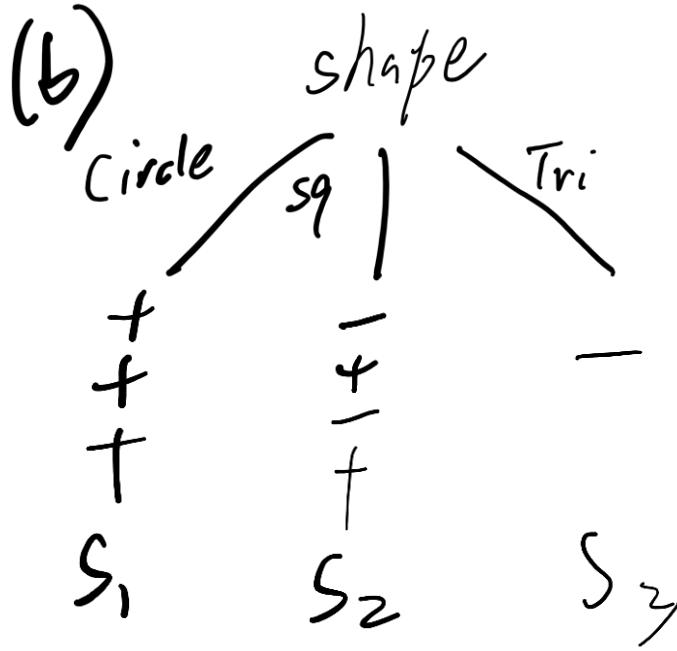
a) What is the **entropy** of this collection of training examples?

b) What is the **information gain** of the attribute **shape**?

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

$$\approx 0.42 + 0.53$$

$$= 0.95$$



$$T_1 = 0.95$$

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

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

因为是 pure set

$$H_2(s) = I\left(\frac{1}{4}, \frac{1}{4}\right) = -\sum_{i=1}^2 \log_2 \frac{1}{4} - \sum_{i=1}^2 \log_2 \frac{1}{2} \\ = 1$$

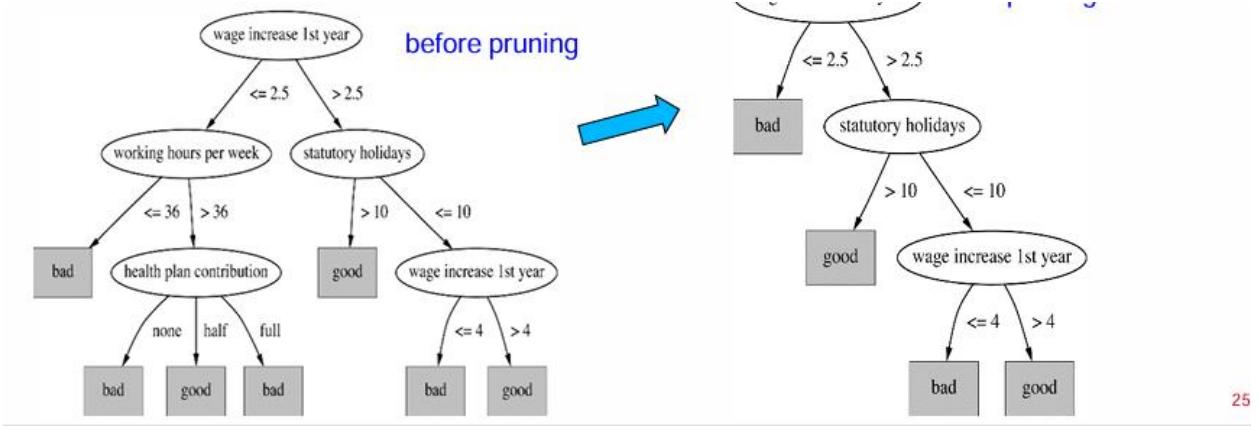
$$\bar{T}_2 = \frac{4}{8} = \frac{1}{2}$$

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

- 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 来决定

Method: sub-tree replacement

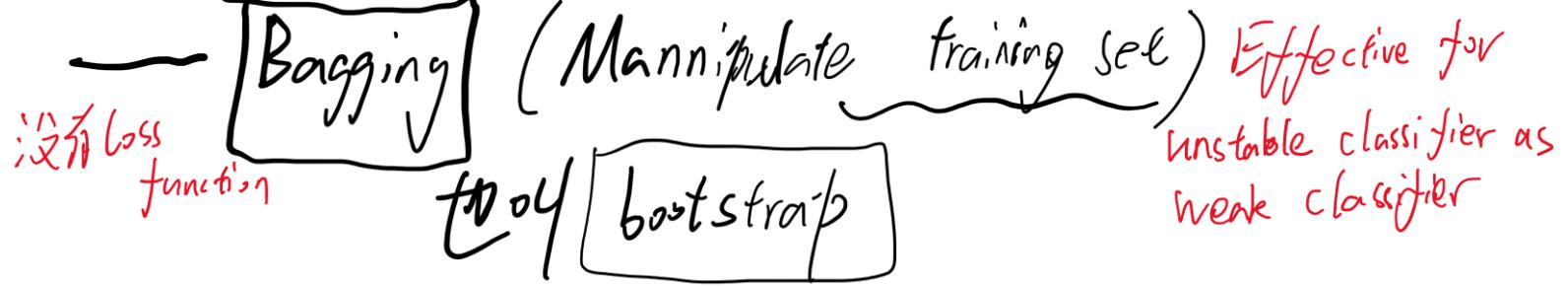
把分支简化成leaf node.
其中多的作为label



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. 不到 Ensemble 的 ε 等同于 $\text{avg}(\varepsilon_{\text{of base classifier}})$



我们知道从子集的 Dataset \rightarrow randomly choose data points at least 组成新的 Dataset, 新 Dataset 的 data points 数量和原先的一样, 但是 You'll see bootstrap sample 可能重复或不包含原先的一些 data point 在新的 data 中

假设我们要 Ensemble m 个 basic classifier, 我们就要分别创建 m 个 bootstrap sample 用这些 data 作为 training data 分别训练 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



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

were difficult for the previous classifier.)



Steps: \rightarrow 主要会针对二分类

$$\text{loss function: } L(h(x), y) = e^{-y h(x)}$$

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

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

- 可以看到 →
- 每次 training
set 权重不同
- Training a new weak classifier $h_t(x)$ by using $W_t(n)$
 - Calculate $\epsilon_t = \sum W_t(n) I[y_n \neq h_t(x_n)]$ # 错误率
 - Calculate weight β_t of current weak classifier \star
 - Update weight for data points
- $$\beta_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$
- $$W_{t+1}(n) \propto W_t(n) e^{-\beta_t y_n h_t(x_n)}$$
- 即若计算 $W_{t+1} = W_t(n) e^{-\beta_t y_n h_t(x_n)}$
- normalize it 以达到 $\sum W_{t+1}(n) = 1$

Lastly, output final classifier

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

注意的是：如果某些 data point 在之前的多轮分类中已被

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

2. Greedily minimize the exponential loss Function

It works because

逐层
模型
评估
性能

3. We must optimize $E_t = \sum w_t(n) I[Y_n = h_t(x_n)]$ to choose $h_t(x_n)$

4. and optimize by using $\beta_t^* = \frac{1}{2} \log \frac{1-E_t}{E_t}$ 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 m training data
- 从 m decision tree + random choose subset of features 作为 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 tree 的 Accuracy 越高

- 虽然单个 decision tree, Random forest 有缺点:
 - 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	+

$$P(H|E) = \frac{P(E_1|M_1) \cdots P(E_n|M_n)}{P(E)}$$

H: Yes

$$P(\text{home owner} = \text{no} | \text{Yes}) = \frac{3}{5} \quad P(\text{annual income} = \text{vh} | \text{Yes}) = \frac{1}{5}$$

$$P(\text{marital status} = \text{married} | \text{Yes}) = \frac{1}{5} \quad P(H) = \frac{5}{10}$$

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

H: No

$$P(\text{home owner} = \text{no} | \text{No}) = \frac{3}{5} \quad P(\text{annual income} = \text{vh} | \text{No}) = \frac{2}{5}$$

$$P(\text{marital status} = \text{married} | \text{No}) = \frac{3}{5} \quad P(H) = \frac{5}{10}$$

$$P(\text{No} | E) = \frac{\frac{3}{5} \times \frac{3}{5} \times \frac{2}{5} \times \frac{5}{10}}{P(E)} = \frac{\frac{9}{250}}{P(E)}$$

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

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

$$\bar{E} = \frac{\sum E_i}{n}$$

$$G = \sqrt{\frac{\sum (E_i - \bar{E})^2}{n-1}}$$

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

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

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

$$G = \sqrt{\frac{970}{94}} = \frac{15.57}{10.38/6} = 10.38/6$$

$$G^2 = \frac{970}{44} = 107.78 \quad 15.57^2$$

$$pdf = \frac{1}{\sqrt{2\pi} \cdot 10.38/6} e^{-\frac{(E_i - 99)^2}{2 \cdot 107.78}} \quad 15.57^2$$

H: Yes

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

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

$$P(h_0 = no | Yes) = \frac{3}{5}$$

$$P(aI = 120 | Yes) = 2.8025 \times 10^{-3}$$

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

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

$$1-H: No: P(H) = \frac{5}{10}$$

$$6.19 \times 10^{-4}$$

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

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

$$M = \frac{70 + 120 + 60 + 220 + 75}{5} = 109$$

$$\sigma = \sqrt{\frac{1752^2}{94}} = \frac{44.121}{66.181}$$

$$\sigma^2 = 66.181^2$$

$$P(aI \leq 120 | N_0) = \frac{1}{\sqrt{2\pi \frac{94.121}{66.181}}} e^{-\frac{(120 - 109)^2}{2 \cdot 19.667}}$$

$$= 4.945 \times 10^{-3} = 4.94529 \times 10^{-3}$$

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

$$P(E)$$

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

$$P(1N_0 | E) > P(\text{Yes} | E) \quad \text{so } N_0$$

Week 5 TUI

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	+

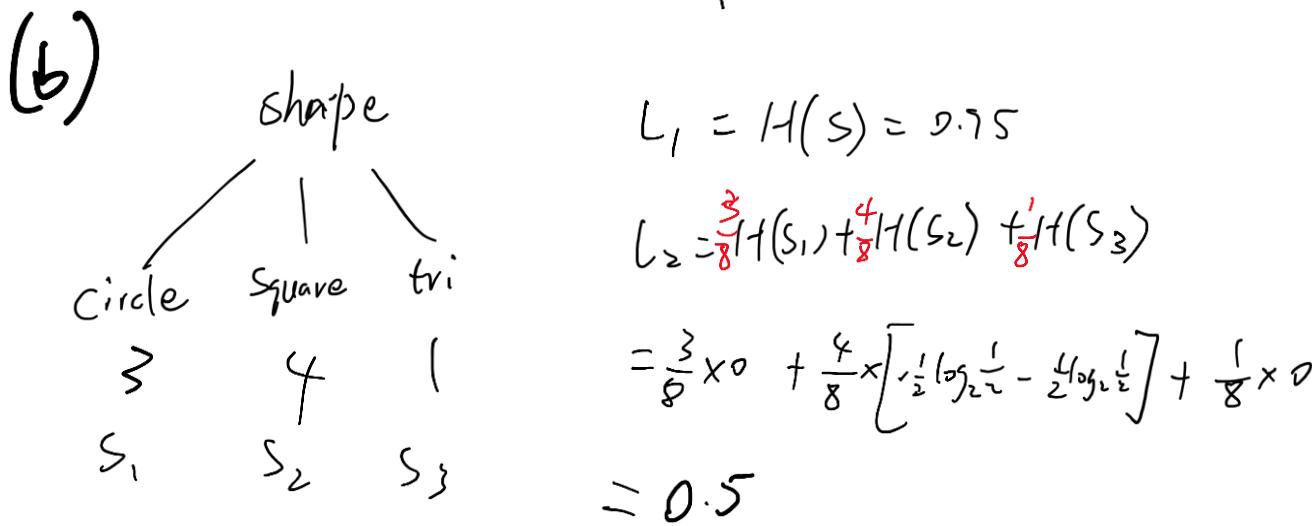
Adapted from M. Kubat, Introduction to Machine Learning, Springer, 2021

- What is the entropy of this collection of training examples with respect to the class?
- What is the information gain of the attribute *shape*?
- Which attribute will be selected as root of the tree based on information gain?
- 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 + } - Entropy = $\frac{5}{8} \log_2 \frac{5}{8} + \frac{3}{8} \log_2 \frac{3}{8}$
 $= 0.95$



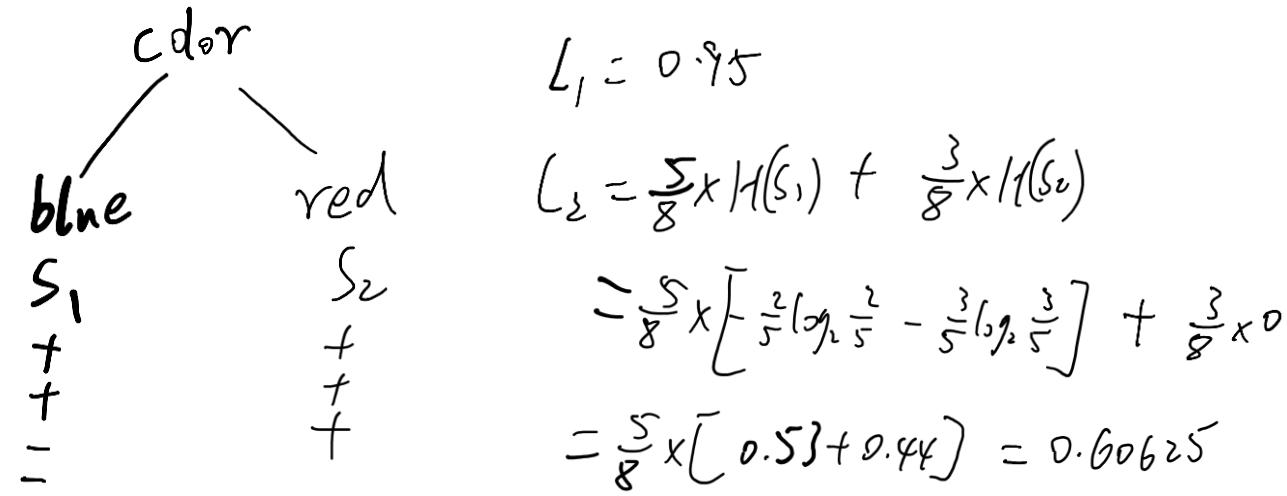
+

+

-

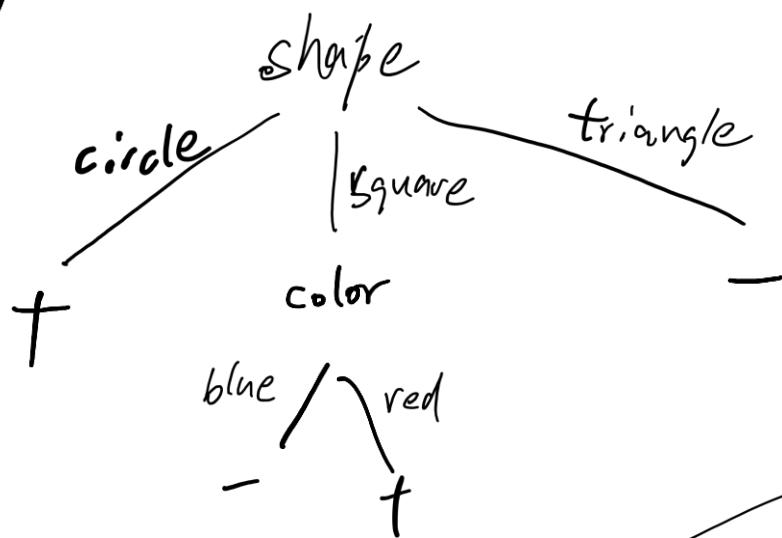
$$\text{Information Gain} = 0.95 \times 0.5 = 0.45$$

(c) Need calculate other Information Gain



Since $0.5759 > 0.45$ So choose ~~color~~ as root
 $0.34 < 0.45$ ~~shape~~

(d)



overfitting \rightarrow should reduce

underfitting \rightarrow should increase

还有一个 hyperparameter, **gamma**, 大的

表示小 width of margin,

小的表示大 width of margin

Week 6

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

① Support Vector Machines

SVM, ML, classification

- Support Vectors: data points from different classes which lie closest to the decision boundary; laying on hyperplane
↳ in the middle of margin

Goal

- Margin: the distance between support vectors

- 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?

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

↙ 使用 Lagrange multiplier

$$W = \sum_{k=1}^N \lambda_i y_i x_i$$

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

$\lambda_i = 0$ means data point is not

对应的第 k 个 feature

↓ N 是 support vector 数量

↳ 每个 data point

都要正确分类,

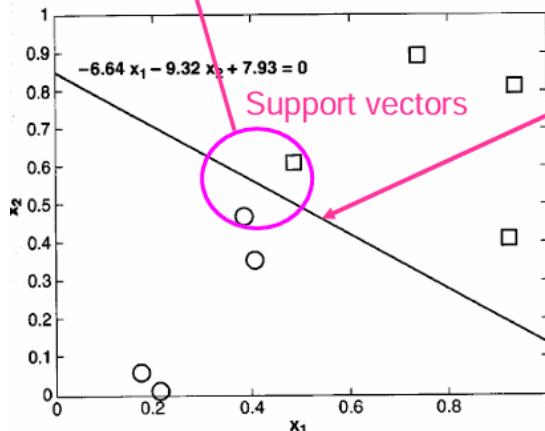
→ 0.1. (soft) margin 当不能被完全分类时

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

只用支持向量
vector of \mathbf{w}_i ,
最后结果是 \mathbf{w}

Linear Example

	features		class	Lagrange Multiplier
	x_1	x_2	y	
x_1	0.3858	0.4687	1	65.5261 λ_1
	0.4871	0.611	-1	65.5261 λ_2
x_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



- 8 2-dim. training examples; 2 classes: -1, 1
- After solving the problem with QP we find the λ s
- Only 2 λ s are non-zero (x_1 & x_2) and they correspond to the support vectors
- Using the λ s, the weights (defining the decision boundary are):

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

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

$$b = 7.93 \quad // \text{there is a formula for } b \text{ (not shown)}$$
- Classifying new examples:
 - above the decision boundary: class 1
 - below: class -1

$$\text{Classifier} = -6.64x_1 - 9.32x_2 + 7.93$$

19

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

- 有些时候容错率 ξ 越大，misclassify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 f(发现的) linear boundary 互吸
- ③ original space 无法有 non-linear boundary

$$f = w \cdot z + b = \sum_{i=1}^N \lambda_i y_i \underbrace{k(x_i, z)}_{\text{kernel trick}} + b$$

can help us do the calculation
in original space

Kernel function 需要满足 Mercer's
Theorem

b-bt⁰ poly $(x \cdot y + 1)^P$

Radial	$e^{...}$
tangent	$\tanh(\cdot)$

③ Dimensionally Reduction

PCA

compression • PCA is **unsupervised**

+ • removes redundant and highly correlated features and reduces noise
Feature Extraction

→ **Principal Components** z_i , new axes
 z_i orthogonal with each other

$$\text{Var}(z) > \text{Var}(z_1) > \dots > \text{Var}(z_m)$$

m is the # of features

Steps:

- ① find principal components z_1, \dots, z_m
- ② project original data to new axes z_1, \dots, z_k
 $k < m$

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

具体实现方法 SVD) Use this 2 methods

$$\begin{array}{ccc}
 U_{n \times m} & & U_{n \times k} \\
 \Lambda_{n \times m} & \Rightarrow & \Lambda_{k \times k} \\
 V_{m \times m} & & V_{k \times k}
 \end{array}$$

the transformed data
 (projection)
 diagonal matrix
 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:

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

- 压缩比

- 小数表示压缩比

Week 7: 下面再都是 Deep learning

a general-purpose framework for representation learning

① Perceptron: Simplest NIV

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

$$w^{new} = w^{old} + \epsilon x^T$$

其 Element 只会是 0 和 1

$$b^{new} = b^{old} + \epsilon x$$

$$\text{Error} = \frac{1}{n} \sum_{i=1}^n (t_i - a_i)^2$$

Steps:

- Initialize w & b to random small value
- while not reach stop condition:

- 这里对每个 data point 都要计算一次且 data point 的 w 从 w_i 变成 w_i^{new}
- calculate a
 - compute error = $t - a$
 - update w^{new} and b^{new}
 - check stop cond, such as no prediction error 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:

$$w = [-0.7 \ -0.8 \ 1.4]$$

$$b = -0.9$$

Check Continue

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

or not

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

$$a = \text{step}([-0.7 \ -0.8 \ 1.4][1 0 0] - 0.9) = \text{step}(-1.6) = 0, \text{ correct}$$

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

$$a = \text{step}([-0.7 \ -0.8 \ 1.4][1 0 1] - 0.9) = \text{step}(-0.2) = 0, \text{ incorrect} \Rightarrow \text{condition not satisfied}$$

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

=> continue training

Start epoch 2:

training: ...

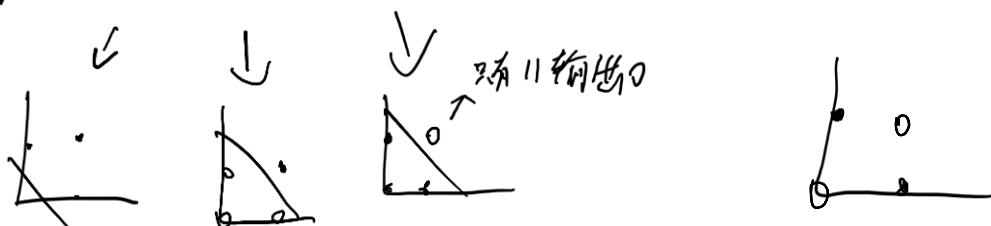
End epoch 2: check stopping criterion

...

- Limitation : *to converge*

- Only guarantee when data is linearly separable

Works for OR, AND, NAND Gate not works for XOR



要在 input layer 扁平化 data

- ③ MLP
- ↳ error function: $VE \& MSE$
↳ overfitting: \approx layers number
- 结构: 1 input layer + multiple hidden layer + 1 output layer
 - each neuron receive input only from the previous layer
 - Fully connected \rightarrow high training time
 - Transfer function need to be differentiable, since need to calculate gradient
- Back Propagation
- a. 需要先正向 forward propagation 得到 \rightarrow real output
- 再根据 real output 和 Expected output 算 Error δ for output neuron
 - 然后往前更新 weight, 直到达到预期的 output
 - Pq : neuron_p \rightarrow neuron_q
 - W_{pq} : weight we use between p and q
 - ΔW_{pq} : means the activation function for neuron q
- b. $W_{pq}^{new} = W_{pq}^{old} + \Delta W_{pq}$
- 这是每个 neuron 的 output
→ 在 output layer
neuron 会直接输出
出结果而已
要用 input f output
- Case 1: q is output layer neuron:
- 需要对 {real output t} and {expected output o}

可以看到 $w_{1,2}, w_{2,3}$
 w_{p_1, p_2} 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:

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

是旧的 w_{qi} 而不是上面
计算的新 w_{qi}

new δ_i

这是对 q neuron 后面下一层 neuron 的 δ_i 和

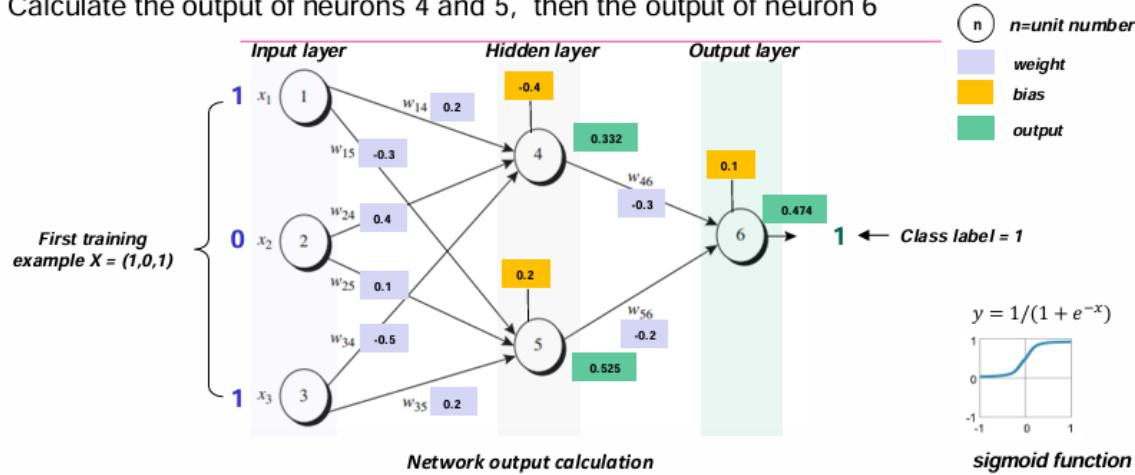
C. $bias_q^{\text{now}} = bias_q^{\text{old}} + \eta \times \delta_q$

input layer neuron 没有 b , 不用 update

. ⑤ δ_i bias 也要同步 update
neuron i

- Example

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



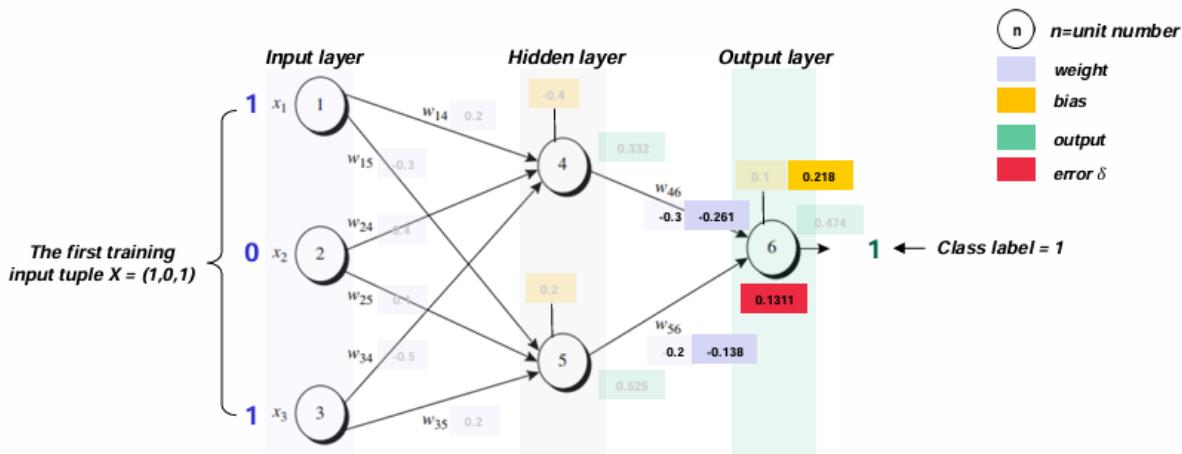
- Input to neuron 4: $z_4 = 1 * 0.2 + 0 * 0.4 + 1 * (-0.5) - 0.4 = -0.7$, output of neuron 4: $o_4 = 1/(1+e^{-0.7}) = 0.332$
- Input to neuron 5: $z_5 = 1 * (-0.3) + 0 * 0.1 + 1 * 0.2 + 0.2 = 0.1$, output of neuron 5: $o_5 = 1/(1+e^{-0.1}) = 0.525$ NN output
- Input to neuron 6: $z_6 = 0.332 * (-0.3)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

先根据 input layer neuron 的 output x 对应 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

→ 用 N_4, N_5 的 output 作 N_6 的 input

↓ ...

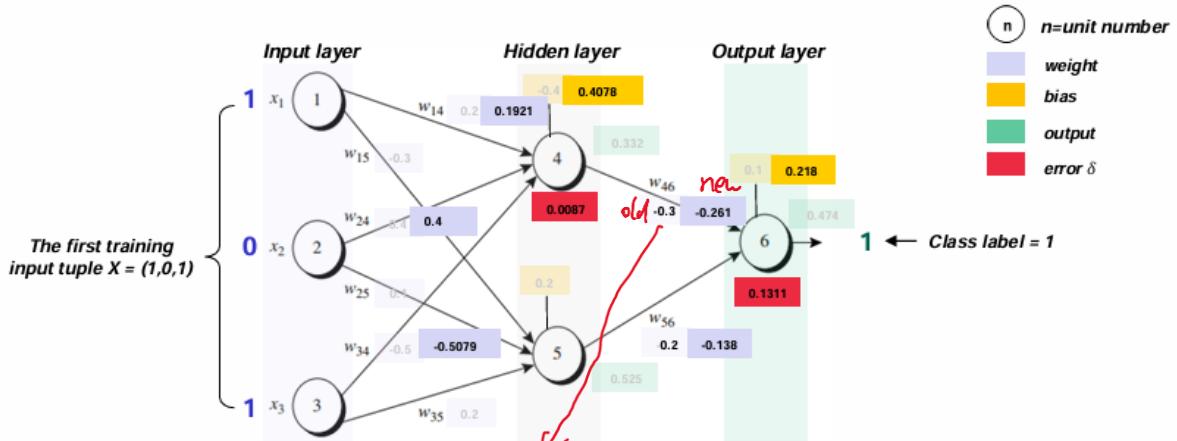


- $\delta_6 = (t_6 - o_6) * o_6 * (1 - o_6) = (1 - 0.474) * 0.474 * (1 - 0.474) = 0.1311$
- $\Delta w_{46} = \eta * \delta_6 * o_4 = 0.9 * 0.1311 * 0.332 = 0.039$, $w_{46\text{new}} = w_{46\text{old}} + \Delta w_{46} = -0.3 + 0.039 = -0.261$
- $\Delta w_{56} = \eta * \delta_6 * o_5 = 0.9 * 0.1311 * 0.525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

① 先算 δ_q

② 再 update W_{ij}

③ 然后 update $bias_q$



For hidden neuron 4:

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

所以看到，用的是 w_{46} old 而不是 w_{46} new



review: 之前在 linear regression 中谈到 GD 是看所有的 Data point, 一直减下降, 这里也一样,

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

SGD: update the weight one-by-one

minibatch: 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 \rightarrow overfitting
Too few \rightarrow underfitting

\hookrightarrow we can gradually increase the # of layers

Dropout rate: randomly drop out neurons

- less dependent on certain neurons
- ~~in backpropagation~~ drop

- Learning rate η

time based exponential

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

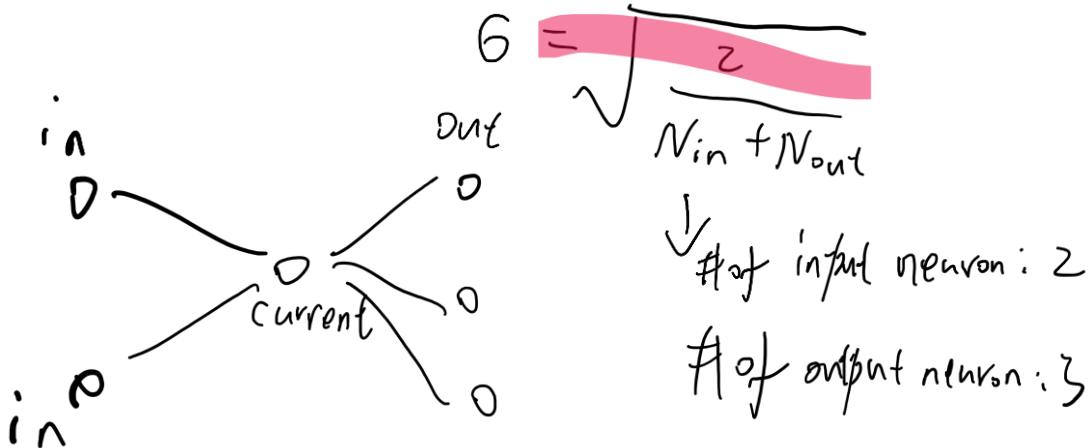
$$\eta_n = \eta_0 e^{-dn}$$

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

$$\eta_1 = 0.2 \quad \eta_2 = 0.1 \quad \dots$$

Weight initialization

Xavier \Rightarrow Sampled from normal distribution, st. centered at 0,



Vanishing Gradient

Opp sigmoid 神经元往

$$S = (t_g - o_g) o_g (1 - o_g) \phi \text{ 不难发现}$$

当 this part 很小时, 会导致 weight 变慢,
even does not update

解决方案: 用 ReLU or LReLU

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

formula:

$$\frac{e^{a_i}}{\sum e^{a_i}}$$

結果は常に 1

- 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 有 overfitting: 無 pooling layer
也會用 backpropagation 調整 weight

① CNN

- Introduce new type of layers: convolutional and pooling

- Can recognize spatial structure of Image 將 Image 轉為資訊

Structure:

Input

Conv

RELU \rightarrow non-linear activation function

Pool

Fully connected with softmax

Hyper para

optimizer: 梯度下降的方法, like adam

Stride: 滑動步長

kernel size: size of filter widow

Padding: 允許超出邊界的大小; 一般為 0

A

Receptive field

Feature map

(earned during training,
not pre-defined)

Filter

1	0	1
0	1	0
1	0	1



Image

1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Feature map

4		

Stride = 2 (filter = 3 x 3)

7 x 7 image

Stride = 2 (filter = 3 x 3)

7 x 7 image

3 x 3 feature map

Red	Green	
Blue		

An example of padding with size 1

Padding with size 1

0	0	0	0	0	0	0	0
0	60	113	56	139	85	0	
0	73	121	54	84	128	0	
0	131	99	70	129	127	0	
0	80	57	115	69	134	0	
0	104	126	123	95	130	0	
0	0	0	0	0	0	0	0

Ke
-1
0

- **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
 - 具体步骤为 使用 filter 从左往右，从上往下，按照 stride 生成 feature map

- **RELU**
 - no upper bound
 - 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

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 window and stride 2

6	8
3	4

b. ave

AP

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

ave

max pool with 2x2 window and stride 2

3.25	5.25
2	2

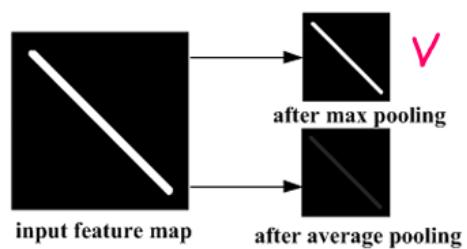
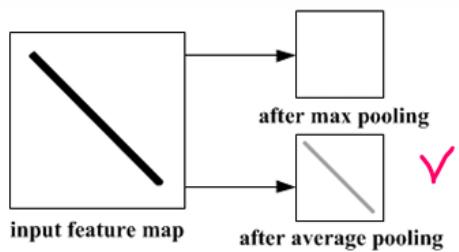
255 是白色

MP: useful for images with dark background



AP:

lighter background



- Fully connected layer : FLY MLP

- input : the first layer, need to "flatten" the output of pooling layer

- output layer : use soft max.

e.g. $[0.1, 0.6, 0.3]$

表示该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 : Hand writing

AlexNet : Image

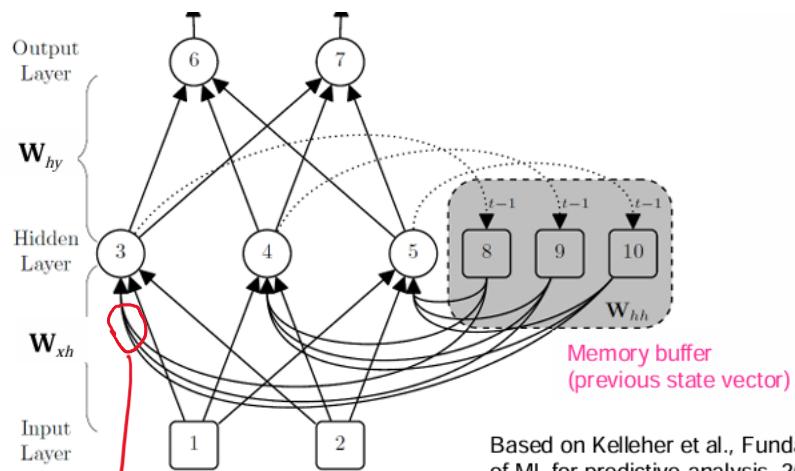
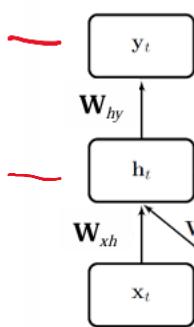


skip back propagation through time
(BPiT)

2.1 Simple RNN

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

b_y



Based on Kelleher et al., Funda
of MI for predictive analysis 20

It's ~~it's~~ need refer to formal info

- ~般用 \tanh as tranverse function:

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

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

- BPIT

• 在整个 sequence predict 完后进行

• 每个 t 都有对应的 Σ_t , 所以根据这个 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

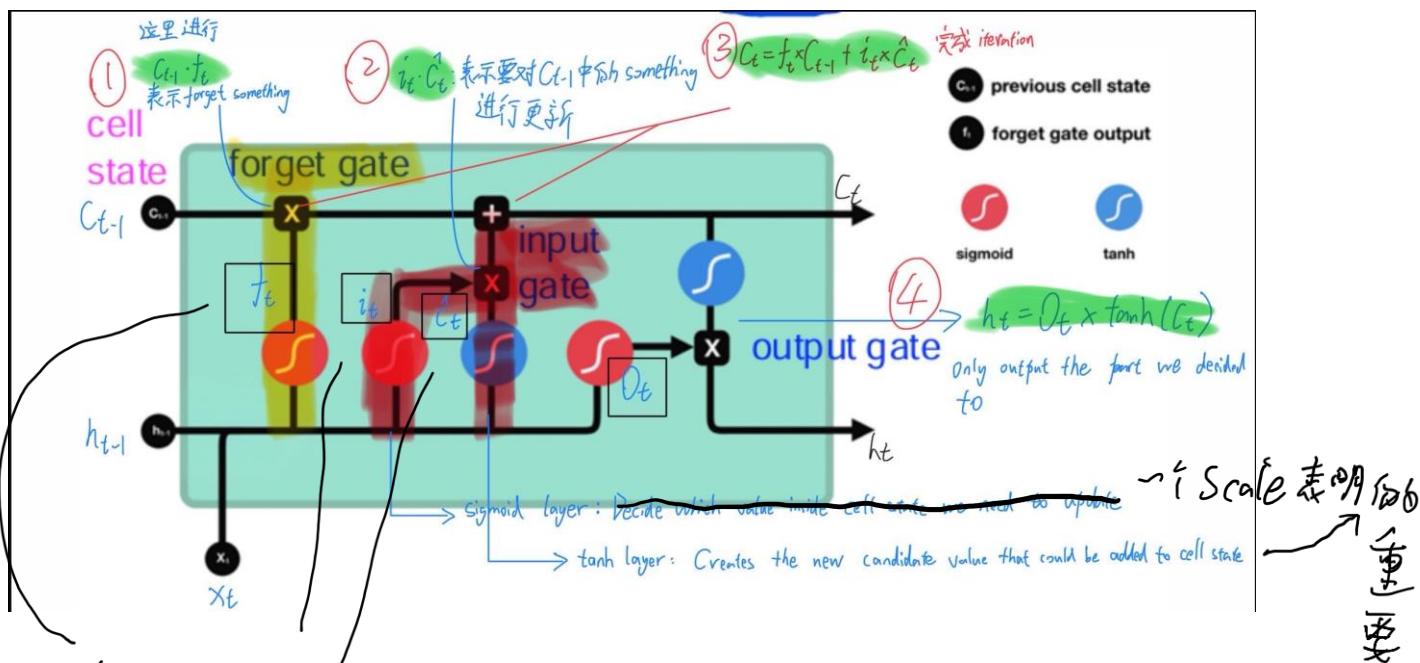
Limitation

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

b. long distance memory 的效果不好, → 使用 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} .



$$\left\{ \begin{array}{l} 6 \\ \text{or} \\ \tanh \end{array} \right. \left(W_i \cdot [h_{t-1}, x_t] + b_i \right)$$

Week 9

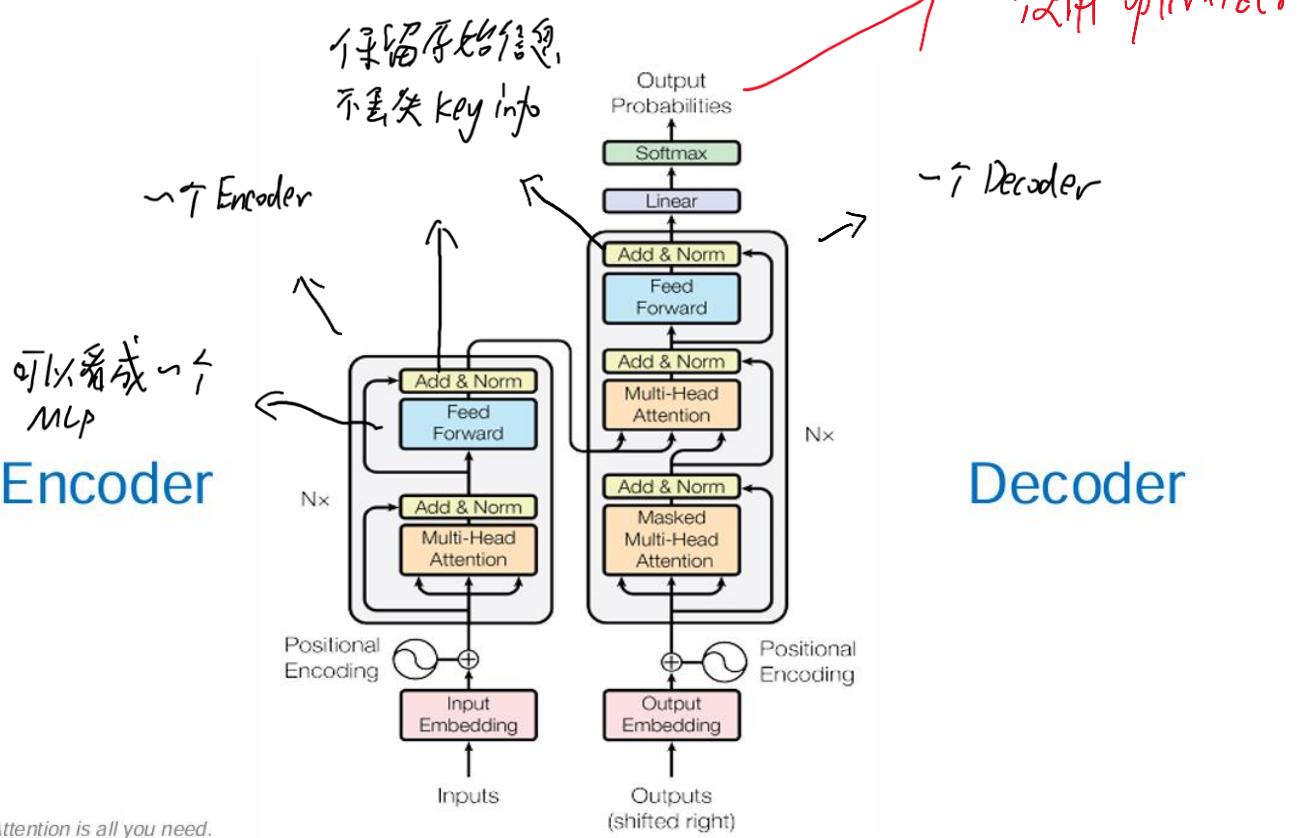
① Transformers

$O(n^2)$

Loss Func: Cross-Entropy

↓
从输出层 error

使用 optimizer



• Decoder 和 Encoder 数量一样

Encoder

- Embedding

• Vector size is the longest sentence in training set

• 与 one-hot 编码, 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

$Q = XW_Q$ $K = XW_K$ $V = XW_V$

W_Q, W_K, W_V 从 Xavier initialize
通过 backpropagation update

- output of self-attention layer is Z

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

drawn down irrelevant words

$\sqrt{d_k}$ the longest vector length in embedding layer

上面都是一头，多头只是

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

$$W \begin{bmatrix} \vdots \\ \vdots \end{bmatrix} \times \begin{bmatrix} Z_0 | Z_1 | \dots \end{bmatrix}$$

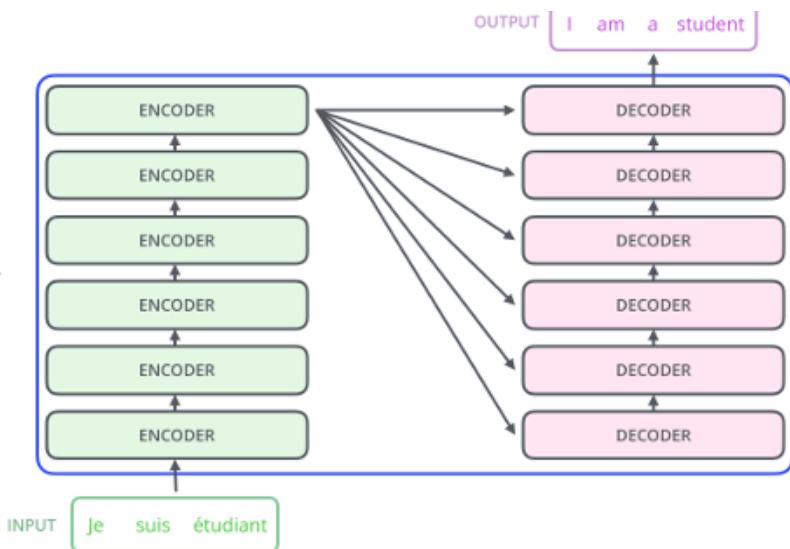
其中 W^o 是被 train 出来的,
一开始也是 Xavier initial
再使用 backpropagation

Decoder

Masked self-attention

- not allow to attend to earlier positions in the output sequence (不包括未来的位置)
- 从 Decoder embedding 获得 "x"

Encoder - Decoder Attention layer



不仅从之前 Decoder 获取信息，还从之前的 Encoder 获得信息。

- 1) Soft max at output

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

- 2) Iteration

Down: Computational cost is high

- Benefits:
- easier to parallelize
 - can be more deeper
- } let down 更好

Week 9 Tui

- 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

$$BSE = \sum_{i=1}^k \sum_{i=1}^K |k_i| \text{dist}(c, c_i)$$

all points
Centroid

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

- similar to other within the cluster (High cohesion)

- dissimilar to other in other clusters (High separation)

- Centroid: Typical not an actual data point

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

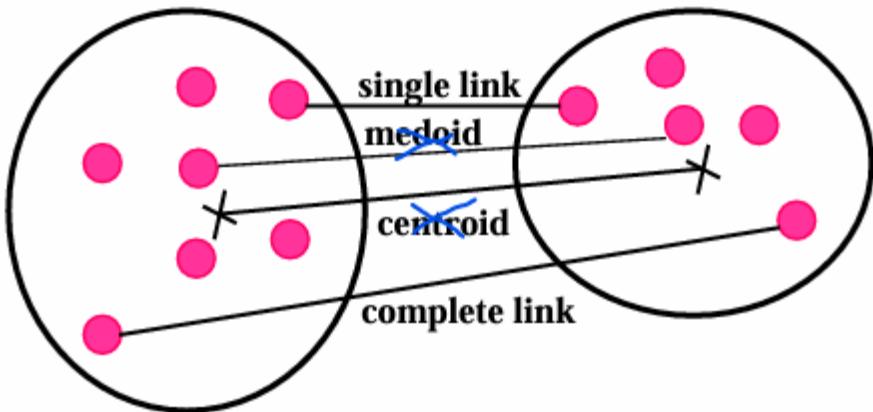
- Medoid: the centrally located data point in the cluster

- 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 \rightarrow # of iter

$O(n k i d)$

of clusters

\downarrow \rightarrow # of attr'

→ If the Data preprocessing has by
of remove outliers

2

K-means

- require k to be specified \rightarrow 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

- [it's over]

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

$$D(A, C) = 7 \times D(B, C) = 3 \checkmark$$

$$D(A, D) = 10 \times D(B, D) = 4 \checkmark$$

$$D(A, E) = 1 \checkmark \quad D(B, E) = 6 \times$$

so clusters should be

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

- [Properties]

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

- 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 ~~farthest~~ away from the current centroids
- \exists outliers

② GMM (Gaussian Mixture Model) high time complexity

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

- Steps

① Initialization, randomly set μ_i and σ_i , $\theta_i = (\mu_i, \sigma_i)$
共有 N clusters, θ_i 为 i 个 θ_i

② Repeat

②.1 Expectation Step:

Calculate prob that each object belongs

to each distribution by using

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

$$\sum w_i = 1$$

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)

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

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

2.2 Maximum Step:

- Recalculate M_i for each cluster
- θ_i does not change
- We use weighted avg points where weights are the P that the point assign to cluster i

3 Until M_i does not change

- Properties

- more flexible \Rightarrow allow elliptical clusters
vs k-means 的不同点

(3)

Hierarchical Clustering

Time: $O(n^3)$
Space: $O(n^2)$ } 会长点

- Easy to Visualize \rightarrow 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

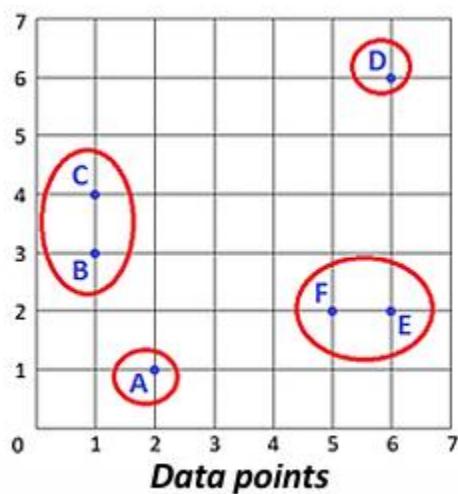
Update the proximity matrix

Until only one single cluster remain

使用
相似度
distance
function,
such as
Euclidean

of ρ_{uv} is used

MAX,
Min,
avg,
Ward



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

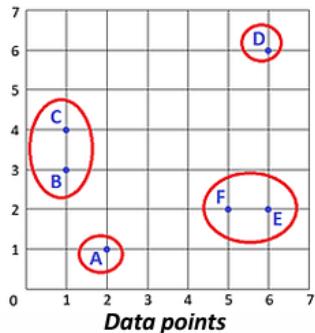
A
C
L
U

Distance matrix (not updated)

pick the 2 smallest one

这里是用 Min

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



	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

不擅长 high dimensional data

time: $O(n^3)$ & 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

根据 the density 分 cluster

- Terminology

- Neighborhood — Eps :
the area within a radius Eps
- Density threshold — MinPts :
the minimum number of points in the Eps
- 3 types of point: within Eps

Core points:

neighbour point $\geq \text{MinPts}$

border points:

not a core point and falls within
neighbour of a core point

Noise points:

既不是核心也不是边界

- 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
- Ties need to be solved when border point belongs to more than one core point

- How to select Eps and MinPts ?

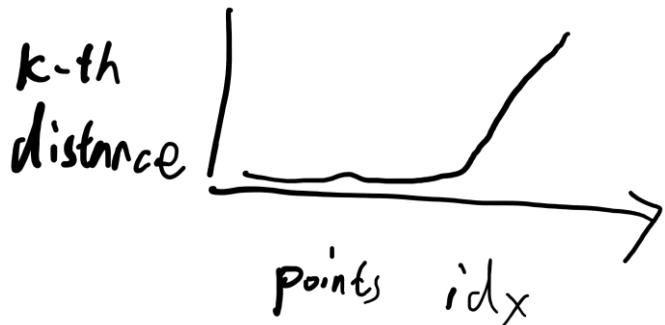
- k -dist
- Eps Examine the distance from a point to its k -th nearest neighbour

idea: if $k <$ the size of a cluster, then the distance should be very small

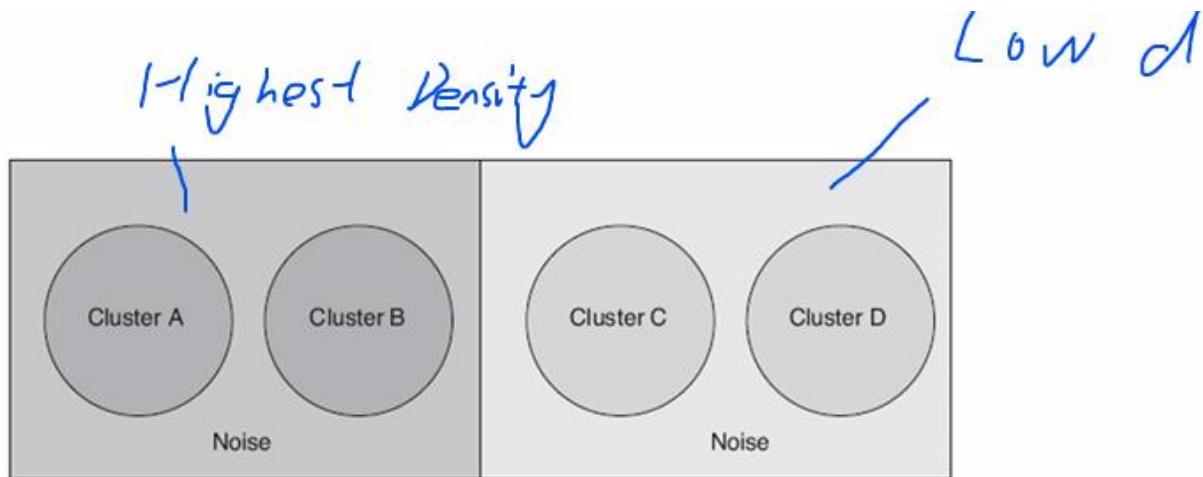
- We can plot the graph for all points

with the distance to its k -th nearest neighbour

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



不能处理 widely varying density data



- 如果 Eps 不够大, 就不行分成 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

主要用 internal — unsupervised measures

a. silhouette ~~☆~~

close to +1 or -1
表示好

b. corr \rightarrow M₁ 和 M₂ 之间的 corr

$$\text{Entry_sim} = \frac{1 - (d_i - d_{\min})}{d_{\max} - d_{\min}}$$

f Entry 的值

1 data point
 $\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix}$

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

$$\begin{bmatrix} 1 & 2 \\ 1 & 0 & 1 \\ 1 & 1 \end{bmatrix}$$

c. Visual

— 重点谈谈 Silhouette for a point

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

$\begin{bmatrix} -1 & 1 \end{bmatrix}$

\rightarrow 正确分类 cluster
 \rightarrow 在 cluster 分界
 \rightarrow 错误分类

b_i : distance from x to all points in other cluster

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

the higher, the better

lower is bad bcs cohesion > separation

- For a cluster:

$$\text{avg}(S_1, S_2, \dots) \leq k_n$$

- For a clustering

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

Trivio

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 中先标出符合
条件的点

⑤ 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, \dots, A_7 are noise

A_3 and A_5 are core and A_4, A_6

$\frac{1}{13}$

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

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

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

bigger is better

bcs $b_i > a_i$ is ok

$b_i < a_i$ means bad

Cluster labels:

point	cluster label
P1	1
P2	1
P3	2
P4	2

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

$$K_1: \underbrace{\frac{5}{6} + \frac{11}{13}}_{\Sigma} = \frac{13}{78} \quad \text{[This is wrong]} \\ = 0.8397$$

$$K_2: \underbrace{\frac{5}{9} + \frac{11}{23}}_{\Sigma} = \frac{214}{207} \quad \text{[This is wrong]} \\ = 0.5169$$

③ For clustering

$$\underbrace{\frac{13}{78} + \frac{214}{207}}_{\Sigma} \approx 1.366518 \\ 0.5169 + 0.8397 \\ 0.678$$

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 $s = 1 - (d - d_{min}) / (d_{max} - d_{min})$, where d_{min} and d_{max} are the minimum and maximum distances in the matrix: $d_{min}=0.1$ and $d_{max}=9$.

1

COMP5318/COMP5318 Machine Learning and Data Mining, s1 2024

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				
	P1	P2	P3	P4
P1	1	1	0	0
P2		1	0	0
P3			1	1
P4				1

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

- $M_1 \rightarrow [1, 0.08, 0.25, 0, 0.17, 0.67] = x$
- $M_2 \rightarrow [1, 0, 0, 0, 0, 1] = y$

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

$$M_y = \frac{2}{6} = \frac{1}{3} = 0.33$$

$$\left(1 - \frac{217}{600}\right) + \left(0.08 - \frac{217}{600}\right) + \left(0.25 - \frac{217}{600}\right) + \left(0 - \frac{217}{600}\right) + \left(0.17 - \frac{217}{600}\right) + \left(0.67 - \frac{217}{600}\right)$$

$$\frac{71}{75} = \begin{matrix} x & x & x & x & x & x \\ \left(1 - \frac{1}{3}\right) & \left(0 - \frac{1}{3}\right) & \left(0 - \frac{1}{3}\right) & \left(0 - \frac{1}{3}\right) & \left(0 - \frac{1}{3}\right) & \left(1 - \frac{1}{3}\right) \end{matrix}$$

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

$$\frac{2\sqrt{3}}{3}$$

$$X \sqrt{2x\left(\frac{2}{3}\right)^2 + 4x\left(-\frac{1}{3}\right)^2}$$

$$= 0.939$$

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

$$P(\bar{x}_i | \pi_1, \dots, \bar{x}_{i-1}) = P(\bar{x}_i | \pi_{i-1})$$

- Predict sequence of data

$$P(\pi_1, \dots, \pi_n) = P(\pi_1) (P\bar{x}_2 | \pi_1) (P\pi_3 | \pi_2) \dots$$

↑
from A_0

② HMM

2.1 HMM P_1 , 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 $N^2 T$

• a_{jk} : 从 A 或 A_o 获取 表示从 $\pi_j \rightarrow \pi_k$ 的 P
 $\underbrace{a_{jk}}_{\text{PP}} = P(\pi_k | \pi_j)$

• $e_k(x_i)$: 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 state k 都要填

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

② Iteration: 通常为 $O(N^2)$, because double iterat

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

上一层的所有 f

③ After iter \rightarrow Termination

$$P(x) = \sum_k f_k(m)$$

②.2 Viterbi $\sim O(NT)$

- It is efficient because we did not do inner loop, we use $\max(\dots)$ instead of \sum
- Given observation sequence X , what's the most likely hidden state sequence p_{tr}

① Initialization

$$V_{ik}(1) = A_0(k) e_k(x_1)$$

② Iteration

$$V_k(i) = c_k(x_i) \max_j f_j(i-1) a_{jk}$$

$$P_{\text{fr}_k}(i) = \operatorname{argmax}_j V_j(i-1) a_{jk}$$

③ Termination

$$\pi_m = \operatorname{argmax}_{|K|} f_k(m)$$

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

$$\pi_1 \rightarrow \dots \rightarrow \pi_m$$

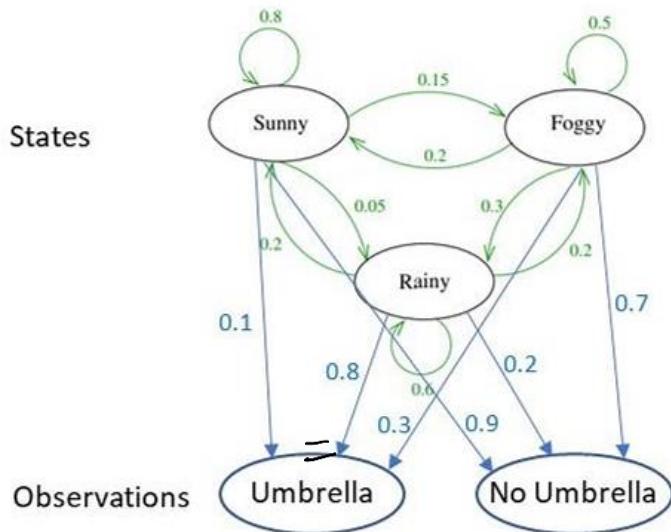
Week 11 Int

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: $A_0(\text{Sunny})=0.4$, $A_0(\text{Rainy})=0.3$ and $A_0(\text{Foggy})=0.3$.



Green is A

Blue is E

$x_1 = \cancel{U_m} \text{ No}$

$x_2 = \cancel{N_o} U_m$

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

$$f_{\text{Rainy}}(1) = A_0(\text{Rainy}) C_{\text{Rainy}}(x_1)$$

$$= 0.3 \times 0.8 = 0.24$$

(b) Sunny

$$f_{\text{Sunny}}(1) = A_0(\text{Sunny}) C_{\text{Sunny}}(x_1)$$

$$= 0.4 \times 0.1 = 0.04$$

(c) Foggy

$$f_{\text{Foggy}}(1) = \alpha_0(\text{Foggy}) e_{\text{Foggy}}(x_1)$$

$$= 0.3 \times 0.3 = 0.09$$

Rainy

Step 2: $i=2$

(a) Rainy

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

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

$$= 0.0288$$

$$P_{\text{tr}_{\text{Rainy}}}(2) = \text{Rainy}$$

(b) Sunny

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

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

$$\approx 0.0432$$

$$P_{tr_{sunny}(z)} = \text{Rainy}$$

(c) Foggy

$$f_{Foggy}(z) = C_{Foggy}(x_z) \cdot \max \left\{ f_{Foggy}^{(1)} a_{foggyfoggy}, f_{Rainy}^{(1)} a_{rainyfoggy}, f_{Sunny}^{(1)} a_{sunnyfoggy} \right\}$$

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

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

$$P_{tr_{foggy}(z)} = \text{Rainy}$$

~~$P_{tr}(z) = \text{Rainy}$~~ 应该每个 Hidden state 都有

Step 3

$$\bar{\pi}_2 = \arg \max \left[\underbrace{f_s(z)}, f_R(z), f_F(z) \right]$$

$$= \underset{\text{sunny}}{\text{sunny}}$$

$$\bar{\pi}_1 = P_{tr_{sunny}(z)} = \text{rainy}$$

上面因为 sequence 看错了，所以需要整个重做

Step 1 :

(a) Rainy: $\check{V}_R(1) = A_o(R) C_R(x_1)$
 $= 0.3 \times 0.2 = 0.06$

(b) Sunny

$$\check{V}_S(1) = A_o(S) C_S(x_1) = 0.4 \times 0.9 = 0.36$$

(c) Foggy

$$\check{V}_F(1) = A_o(F) C_F(x_1) = 0.3 \times 0.7 = 0.21$$

Step 2. Iteration, only loop once

$$i = 2$$

(a) Rainy

$$\begin{aligned} \check{V}_R(2) &= C_R(x_2) \cdot \max \left\{ \underbrace{\check{V}_R(1) \alpha_{RR}}_{0.06 \times 0.6}, \underbrace{\check{V}_S(1) \alpha_{SR}}_{0.36 \times 0.05}, \underbrace{\check{V}_F(1) \alpha_{F2}}_{0.21 \times 0.3} \right\} \\ &= 0.8 \times \max \{ 0.06 \times 0.6, 0.36 \times 0.05, 0.21 \times 0.3 \} \\ &= 0.8 \times 0.21 \times 0.3 = 0.0504 \end{aligned}$$

$$P_{tr_R}(2) = \hat{Foggy}$$

(b) Sunny

$$\begin{aligned} V_S(z) &= e_S(x_L) \max \left\{ V_S(1) a_{SS}, V_R(1) a_{RS}, V_F(1) a_{FS} \right\} \\ &= 0.1 \times \max \left\{ \underbrace{0.36 \times 0.8}_{\dots}, \dots \right\} \\ &= 0.1 \times 0.36 \times 0.8 = 0.0288 \end{aligned}$$

$P_{tr_S}(z) = \text{Sunny}$

(c) Foggy

$$\begin{aligned} V_F(z) &= e_F(x_L) \cdot \max \left\{ \dots \right\} \\ &\approx 0.3 \times 0.21 \times 0.5 = 0.0315 \end{aligned}$$

$P_{tr_F}(z) = \text{Foggy}$

Step 3:

$$\pi_2 = \operatorname{Argmax} \left\{ V_F(z), V_S(z), V_R(z) \right\}$$

= Rainy

$\pi_1 = P_{tr_K}(z) = F$

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 states

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. 只由上一步的
State 状态决定

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.

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

↑
fix initial action a_0

By definition:

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

- \hat{Q} the max Expected cumulative reward from a given pair

Bellmen equation

給る suggestion \Rightarrow \hat{Q} 算出

$$\hat{Q}(s, a) = \text{Expected reward} + \text{next state cumulative reward}$$

$$= \mathbb{E}_{\substack{s' \sim \mathcal{E} \\ a'}} [r + \gamma \max_{a'} \hat{Q}(s', a') \mid s, a]$$

$$r^o = |$$

next state Action-Value function



次回の状態で以降 $\hat{Q}(s', a')$

- Q-learning

① calculate

$\hat{Q}(s, a)$ before taking action a

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

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

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

new

→ we want this become,

Since it means we find the best action

- Deep Q-Learning

不需要传入 action, 只用传入 state

• Recall

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

• we have loss function

$$L = \left[\underbrace{r + \max_{a'} Q(s', a')}_{\text{Target}} - \underbrace{Q(s, a)}_{\text{current}} \right]^2$$

• 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

