ARM 1:

Rule measure support, confidence

for $X \Rightarrow Z$ sup = P(X, Z) con = P(Z|X)

Apriori principle: 一频繁集的子集也应该是频繁集

Apriori 算法 C_k: 大小为k的候选集 L_k 为频繁集

L_1 = {frequent items}

for ($k=1$; L_k != 空; k++) {

C_{k+1} = candidates from L_k

计算 C_{k+1} 支持度 (scan-遍历数据)

L_{k+1} = C_{k+1} with min_support

end

return $\cup_k L_k$;

假设 L_{k-1} 是 sort 完的频繁集

step 1: self-joining L_{k-1}

insert into C_k

select p.item$_1$, ... p.item$_{k-1}$, q.item$_{k-1}$

from L_{k-1} p L_{k-1} q

where p.item$_1$=q.item$_1$, ... p.item$_{k-1}$ < q.item$_{k-1}$

step 2 pruning

for all itemsets c in C_k do

for all (k-1) subsets s of c do

if (s is not in L_{k-1}) then delete c from C_k

Rule Generation: 对每个 frequent itemset L

生成所有非空子集, 对每个集s, 生成规则 $s \Rightarrow L-s$

如果 conf($s \Rightarrow L-s$) = support(L)/support(s) ≥ min_con

则这是一个 strong association rule

The core of Apriori algorithm

Use frequent (k-1)-itemsets to generate C_k

Use database scanning and pattern match to collect counts for the candidate itemsets

Need (n+1) scans, where n is the length of the longest pattern. (最长的频繁集)

Interest / Lift: $\frac{P(A,B)}{P(A)P(B)}$ 小于1负相关

ARM 2: sequence consists of a list of itemset in temporal order and denoted <s1, s2, ..., sn>

sequential Association Rule Mining steps

1. sort phase: 转化为有序的表

2. frequent itemset phase: Find the set of all frequent/large itemset L

3. Transformation Phase: 将每个 sequence 转化成 frequent item 的形式

4. sequence phase: 用 Apriori 发现频繁 sequence

5. 最大化: 找到所有不是任何 sequence subsequence 的 sequence

ARM 2: 如果 Rule 的 support 与其子父规则预测值相近, 则此 Rule 是 redundant 的

在前 item 旁加上大类分类 (ancestor)

Algorithm Basic

- Form the extended transactions

Extended Transactions T'

Transaction	Items bought
100	Shirt, (Clothes)
200	Jacket, (Outerwear, Clothes,), Hiking Boots, (Footwear)
300	Ski Pants, (Outerwear, Clothes,) Hiking Boots, (Footwear)
400	Shoes, (Footwear)
500	Shoes, (Footwear)
600	Jacket, (Outerwear, Clothes)

- Find frequent itemsets (minsup=30%, minconf=60%)
- Find the rules

Frequent Itemsets

Itemset	Support
{Jacket}	2
{Outerwear}	3
{Clothes}	4
{Shoes}	2
{Hiking Boots}	2
{Footwear}	4
{Outwear, Hiking Boots}	2
{Clothes, Hiking Boots}	2
{Outerwear, Footwear}	2
{Clothes, Footwear}	2

Rules

Rule	Support	Conf.
{Outerwear} ⇒ {Hiking Boots}	33%	66%
{Outerwear} ⇒ {Footwear}	33%	66%
{Hiking Boots} ⇒ {Outerwear}	33%	100%
{Hiking Boots} ⇒ {Clothes}	33%	100%

设置有效的 min support

⇒ uniform / reduced 按每个类阈值一样 按比例降低

Classification: seen Data

Training Data Test Data (模型构建用是 unseen 的)

对决策树来说, 前个节点都是 test, 可以写作 if ... then

Information Gain (c=分类)

$$I(p,n) = -\frac{p}{n+p}\log_2\left(\frac{p}{n+p}\right) - \frac{n}{n+p}\log_2\frac{n}{n+p}$$

A 对 p.n 的信息熵 : $E(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I(p_i, n_i)$

A 对 p.n 的信息增益为 Gain(A) = I(p,n) − E(A)

Example: p=9 n=5 I(p,n)=0.94

age	p_i	n_i	$I(p_i,n_i)$
≤30	2	3	0.971
30-40	4	0	0
>40	3	2	0.971

$E[Age] = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$

过拟合: ① 太多分支 ② 对未知样本效果不好

① 预剪枝 ② 后剪枝

大数据库下的分类任务: 在 million 样本 hundreds 个特征下有足够运行速度

why DT:

① 速度快 ② 易转换为容易理解的分类规则

③ 可以用 SQL 连接数据库

回归树: Similarity: Used continuous 结果

分裂多次, 选最好优化的

不同:

① 预测值为所有叶节点下样本的均值

② 不能应用 sum square deviation

③ 表现用 rmse 来衡量

Classification: 回归树

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

Count = n = 14

Average = $\bar{x} = \frac{\sum x}{n} = 39.8$

Standard Deviation = $S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}} = 9.32$

Coefficient of Variation = $CV = \frac{S}{\bar{x}} * 100\% = 23\%$

- Standard deviation (S) is for tree building (branching)
- Coefficient of deviation (CV) is used to determine branching termination.
- Average (Avg) is the value in the leaf nodes

停止条件

① 该特征下所有样本一致

② 没有其他特征时多数为结果

③ 没有多的样本了

- Standard deviation for two attributes

$$S(T,X) = \sum_{c \in X} P(c)S(c)$$

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

S(Hours, Outlook) = P(Overcast)*S(Overcast) + P(Rainy)*S(Rainy) + P(Sunny)*S(Sunny)

= (4/14)*3.49 + (5/14)*7.78 + (5/14)*10.87

= 7.66

Standard Deviation Reduction (SDR)

- Calculate the standard deviation for each branch

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
	SDR=1.66	

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
	SDR=0.17	

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
	SDR=0.28	

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
	SDR=0.29	

$SDR(T,X) = S(T) - S(T,X)$

SDR(Hours, Outlook) = S(Hours) − S(Hours, Outlook)

= 9.32 − 7.66 = 1.66

选 SDR 最大的作根节点

Bayesian Classification: Why?

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

ification II

maximize $P(h|D) = \frac{P(h|D)P(h)}{P(D)}$

⇒ maximize P(h|D) P(h)

$P(C_j|X) \propto P(C_j)\prod_{i=1}^{n} P(x_i|C_j)$

若 i-th 离散, 用 C_j 中 i 的频率 = $P(x_i|C_j)$

如 i-th 连续 用高斯分布去算

eager learning: 提前构建假设, 无须新样本输入即可得出结论 (新样本不改变学习规律)

(决策树, 神经网络, 朴素贝叶斯)

Lazy learning: 预测时才建立假设, 预测时要遍历大量样本, 速度慢

Instance-Based Classifiers

1. store the training records/case

2. Use training records to predict of unseen case

找最相似的样本去预测 不可解释性

An Example

step 1

D'

CustomerID	Customer Sequence
1	<(30),(90)>
2	<(10 20),(30),(40 60 70)>
3	<(30 50 70)>
4	<(30),(40 70),(90)>
5	<(90)>

step 2 Minsup=25%

Freq. Itemsets†	Mapped to
(30)	1
(40)	2
(70)	3
(40 70)	4
(90)	5

† support is defined against customer (not transaction)!!

step 3

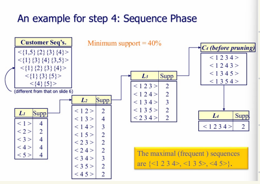
CID	Customer Sequence	Transformed Sequence	Mapping
1	<(30),(90)>	<{(30)} {(90)}>	<{1} {5}>
2	<(10 20),(30),(40 60 70)>	<{(30)} {(40),(70),(40 70)}>	<{1} {2,3,4}>
3	<(30 50 70)>	<{(30),(70)}>	<{1,3}>
4	<(30),(40 70),(90)>	<{(30)} {(40),(70),(40 70)} {(90)}>	<{1} {2,3,4} {5}>
5	<(90)>	<{(90)}>	<{5}>

AprioriAll Algorithm

step 4

```
AprioriAll()
{
  L1 = {frequent 1-itemsets};
  k = 2;           /* k represents the pass number. */
  while (Lk-1 != ∅)
  {
    F = F ∪ Lk;
    Ck = New candidates of size k generated from Lk-1 ;
    for each customer-sequence c ∈ D
      increment the count of all candidates in Ck that are contained in c ;
    Lk = All candidates in Ck with minimum support ;
    k++ ;
  }
  return ( F );
}
```

An example for step 4: Sequence Phase



How to generate candidate sequences?

- The candidate generation steps are similar to those of non-sequential association rule mining. However, be mindful of the order!
- Step 1: self-joining L_{k-1}
  - insert into C_k
  - select p.itemset$_1$, p.itemset$_2$, ..., p.itemset$_{k-1}$, q.itemset$_{k-1}$
  - from L_{k-1} p, L_{k-1} q
  - where p.itemset$_1$=q.itemset$_1$, ..., p.itemset$_{k-2}$=q.itemset$_{k-2}$
- Step 2: pruning
  - forall sequence c in C_k do
    - forall (k-1)-subsequence s of c do
      - if (s is not in L_{k-1}) then delete c from C_k

A sequence <$a_1, a_2 \cdots a_n$> contained in <$b_1, b_2 \cdots b_m$> if exist $i_1 < i_2 < \cdots < i_n$

$a_1 \subseteq b_{i_1}, \cdots a_n \subseteq b_{i_n}$ ⇒ 一个 sequence 是 maximal 的如果 S 不包含于其他 S

sequence Association Rules: 对每个 frequent sequence into S_f S_1 (不能改顺序)

如果 conf($S_f \Rightarrow S_1$) = support(S)/support(S_f) ≥ min-conf, 则为 strong sequential association Rule

用 NN 分类 优势: 准确率高, 稳定 输出结果灵活 劣势: 训练时间长, 不易解释, 且不易融入先验知识

shallow: Perception, MLP, SVM Deep: CNN GNN RNN LSTM Transformer

classification II:

Measuring Error

	Predicted class	
True Class	Yes	No
Yes	TP: True Positive	FN: False Negative
No	FP: False Positive	TN: True Negative

- Error rate = # of errors / # of instances = (FN+FP) / N
- Recall = # of found positives / # of positives
  = TP / (TP+FN) = sensitivity = hit rate
- Precision = # of found positives / # of found
  = TP / (TP+FP)
- Specificity = TN / (TN+FP)
- False alarm rate = FP / (FP+TN) = 1 - Specificity

Estimate error rate

1. partition: 拆分 traning & test set (大型数据集)
2. k-折法：把数据拆成 k 个，用 k-1 训，重复 k 次 (中小)

Ensemble Methods

多个模型一起作出的预测评价小

- Basic idea:
  Build different "experts", and let them vote
- Advantages:
  Improve predictive performance
  Easy to implement
  No too much *parameter tuning*
- Disadvantages:
  The combined classifier is not so transparent (black box; low interpretability)
  Not a compact representation
- Two approaches: Bagging (Bootstrap Aggregating) and Boosting

Random Forests:

① Basic Bagging 对训练集N个树

② Feature Bagging 不同特征用N个不同树

Ada boost:

① 先训练模型 ② 提高错误样本权重 ③ 再训 ④ 投票

- Given: Training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ with $y_n \in \{-1, +1\}$, $\forall n$
- Initialize weight of each example $(\boldsymbol{x}_n, y_n)$: $D_1(n) = 1/N$, $\forall n$
- For round $t = 1 : T$
  - Learn a weak $h_t(\boldsymbol{x}) \to \{-1, +1\}$ using training data weighted as per D_t
  - Compute the weighted fraction of errors of h_t on this training data
    $$\epsilon_t = \sum_{n}^{N} D_t(n) \mathbb{I}[h_t(\boldsymbol{x}_n) \neq y_n]$$
  - Set "importance" of h_t: $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$(gets larger as ϵ_t gets smaller)
  - Update the weight of each example
    $$D_{t+1}(n) \propto \begin{cases} D_t(n) \times \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) = y_n \quad \text{(correct prediction: decrease weight)} \\ D_t(n) \times \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) \neq y_n \quad \text{(incorrect prediction: increase weight)} \end{cases}$$
    $$= D_t(n)\exp(-\alpha_t y_n h_t(\boldsymbol{x}_n))$$
  - Normalize D_{t+1} so that it sums to 1: $D_{t+1}(n) = \frac{D_{t+1}(n)}{\sum_{n=1}^{N} D_{t+1}(n)}$
- Output the "boosted" final hypothesis $H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$

Parcel ID	Origin	Destination	Type	Weight
1	HK	HK	Parcel	Light
2	Kln	Kln	Letter	Light
3	NT	Kln	Letter	Light
4	HK	HK	Parcel	Heavy
5	Kln	Kln	Parcel	Light
6	NT	NT	Letter	Light
7	HK	HK	Letter	Light
8	Kln	Kln	Parcel	Heavy
9	Kln	Kln	Letter	Light
10	HK	HK	Letter	Light
11	HK	HK	Parcel	Heavy
12	Kln	Kln	Letter	Light
13	HK	HK	Letter	Light
14	Kln	Kln	Parcel	Light
15	HK	NT	Parcel	Heavy
16	NT	Kln	Letter	Light
17	HK	NT	Letter	Light
18	Kln	HK	Parcel	Light
19	HK	NT	Parcel	Heavy
20	HK	HK	Parcel	Light
21	Kln	Kln	Letter	Light
22	Kln	HK	Parcel	Heavy
23	Kln	Kln	Letter	Light
24	Kln	Kln	Letter	Light
25	HK	HK	Parcel	Light

a) By setting the support to 25% and confidence to 50%, use the Apriori algorithm to find all interesting association rules in the above Table.

b) Assume that a user sets the lift ratio to 1.75, which rules you discovered for a) above are still interesting?

Suggested Answer: (Please try it first!)

a) min_sup=25% (i.e., ≥ 7 records) and min_conf=50%.

The itemsets (in black) with count ≥ 7 records below are frequent itemsets.

1-itemset	Count	2-itemset	Count	3-itemset	Count
OHK	**11**	**OHK, DHK**	**8**	OHK, DHK, P	5
OKLN	**11**	OHK, DKLN	0	OHK, DHK, Li	6
ONT	3	**OHK, P**	**7**	~~OHK, P, Li~~	
DHK	**10**	OHK, L	4		
DKLN	**11**	**OHK, Li**	**7**	OKLN, DKLN, Li	8
DNT	4	OKLN, DHK	2	~~DHK, P, Li~~	
P	**12**	**OKLN, DKLN**	**9**	DKLN, L, Li	8
L	**13**	OKLN, P	5		
Li	**19**	OKLN, L	6		
H	6	**OKLN, Li**	**9**		
		DHK, P	**7**		
		DHK, L	3		
		DHK, Li	**7**		
		DKLN, P	3		
		DKLN, L	**8**		
		DKLN, Li	**10**		
		P, Li	6		
		L, Li	**13**		

The rules NOT highlighted in red are interesting ones w.r.t min_sup=25% and min_conf=50%. The same procedure can be repeated for frequent 2-itemsets but is omitted here.

b) Recall from lecture notes that

$$Interest(A \to B) = \frac{P(A \wedge B)}{P(A)P(B)} = \frac{P(A \wedge B)}{P(A)} \times \frac{1}{P(B)} = conf(A \to B) \times \frac{1}{P(B)}$$

{OKLN, DKLN, Li}		{DKLN, L, Li}	
OKLN, DKLN → Li	(8/9)/(19/25)	DKLN, L → Li	(8/8)/(19/25)
OKLN, Li → DKLN	(8/9)/(11/25)	DKLN, Li → L	(8/10)/(13/25)
DKLN, Li → OKLN	(8/10)/(11/25)	L, Li → DKLN	(8/13)/(11/25)
OKLN → DKLN, Li	(8/11)/(10/25)	DKLN → L, Li	(8/11)/(13/25)
DKLN → OKLN, Li	(8/11)/(9/25)	L → Li, DKLN	(8/13)/(10/25)
Li → OKLN, DKLN	(8/19)/(9/25)	Li → DKLN, L	(8/19)/(8/25)

The rules above with interest≥1.75 are considered interesting.

Customer	Transaction Time	Items
David	2 Feb 2002	30,50
David	10 Feb 2002	50
David	21 Feb 2002	70
John	5 Feb 2002	10,30
John	7 Feb 2002	50
John	16 Feb 2002	40,60,70
John	27 Feb 2002	50,90
Peter	16 Feb 2002	30,50,70
Aaron	5 Feb 2002	30
Aaron	7 Feb 2002	30, 50
Aaron	17 Feb 2002	70
Aaron	27 Feb 2002	50
Leon	13 Feb 2002	30

(a) *Answer*

Sort Phase:

Customer ID	Customer Sequence
David	<(30 50),(50),(70)>
John	<(10 30),(50),(40 60 70),(50 90)>
Peter	<(30 50 70)>
Aaron	<(30),(30 50),(70),(50)>
Leon	<(30)>

Litemset Phase: min_sup = 25% (i.e. >= 2 customers)

Freq. Itemsets	Mapped to
(30)	1
(50)	2
(70)	3
(30 50)	4

Sequence Phase:

1-sequence	Count	2-sequence	Count	3-sequence	Count	4-sequence	Count
<1>	5	~~<1 1>~~	~~1~~	**<1 2 3>**	**3**	**<1 2 3 2>**	**2**
<2>	4	**<1 2>**	**3**	**<1 3 2>**	**2**	~~<1 2 2 3>~~	0
<3>	4	~~<2 1>~~	0	**<1 2 2>**	**2**	~~<1 2 3 3>~~	0
<4>	3	**<1 3>**	**3**	~~<1 3 3>~~	0	~~<1 3 2 2>~~	0
		~~<3 1>~~	0	~~<2 2 2>~~	0	~~<1 2 2 2>~~	0
		~~<1 4>~~	~~1~~	~~<2 2 3>~~	~~1~~	~~<2 3 2 2>~~	0
		~~<4 1>~~	0	**<2 3 2>**	**2**		
		<2 2>	**3**	~~<2 3 3>~~	0		
		<2 3>	**3**	~~<3 2 2>~~	0		
		<3 2>	**2**	~~<4 2 2>~~	0		
		~~<2 4>~~	0	~~<4 2 3>~~	~~1~~		
		<4 2>	**2**	~~<4 3 2>~~	~~1~~		
		~~<3 3>~~	0	~~<4 3 3>~~	0		
		~~<3 4>~~	0				
		~~<4 3>~~	**2**				
		~~<4 4>~~	0				

Table 1. Social network user popularity data

User ID	A1: Number of followers	A2: Number of followings	A3: Privacy level	Popularity Score
A	Many	Few	Low	75
B	Few	Many	High	72
C	Few	Many	High	30
D	Many	Many	Low	90
E	Many	Few	Low	87
F	Few	Many	High	42

Suppose you are asked to construct a regression tree based on all the six records in Table I.

a) Given the following statistics about the popularity score, determine the root node of the regression tree to be constructed based on standard deviation reduction. Show the involved computations.

COUNT n		6
AVERAGE $\bar{x}$	$\frac{\sum x_i}{n}$	66.00
Std.Deviation S	$S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}}$	22.38
Coeff. Of Variation CV	$CV = \frac{S}{\bar{x}} \times 100\%$	34%

The following pre-computed statistics may help to speed up your calculation of the SDR values:
Average of (75, 90, 87)=84
Std. Deviation of (75, 90, 87)=6.48
Average of (72, 30, 42)=48
Std. Deviation of (72, 30, 42)=17.66
Average of (72, 30, 90, 42)=58.5
Std. Deviation of (72, 30, 90, 42)=23.76
Average of (75, 87)=81
Std. Deviation of (75, 87)=6

(18 marks)

Ans.
Standard deviation for two attributes (target and predictor).

Consider A1

		SD	COUNT
A1	Many	6.48	3
	Few	17.66	3

S(TP,A1)	12.07
SDR(TP,A1)	10.31

SDR: Standard Deviation Reduction

Consider A2

		SD	COUNT
A2	Many	23.76	4
	Few	6	2

S(TP,A2)	17.84
SDR(TP,A2)	4.54

Consider A3

		SD	COUNT
A3	High	17.66	3
	Low	6.48	3

S(TP,A3)	12.07
SDR(TP,A3)	10.31

So, either A1 or A3 should be selected as the root node.

(b) *Answer*

Rule	Support	Confidence	Strong or not?
1 → 2 3 2	2 (40%)	2/5 (40%)	No
1 2 → 3 2	2 (40%)	2/3 (~66.7%)	Yes
1 2 3 → 2	2 (40%)	2/3 (~66.7%)	Yes

b) Based on your regression tree with only the root node in part (a), predict the popularity score of users X and Y below.

User ID	A1: Number of followers	A2: Number of followings	A3: Privacy level	Popularity Score
X	Many	Many	High	?
Y	Few	Few	Low	?

(6 marks)

Ans.
Based on A1 as the root node, the popularity score of X is 84
Based on A1 as the root node, the popularity score of Y is 48

c) WITHOUT guessing/inducing the missing values (denoted as "---"), show how the popularity score of the following data records should be predicted by your regression in part (a).

User ID	A1: Number of followers	A2: Number of followings	A3: Privacy level	Popularity Score
V	Many	---	Low	?
W	---	Many	High	?

(6 marks)

Ans
Based on A1 as the root node, the popularity score of V is 84
Based on A1 as the root node, the popularity score of W is undetermined. If we consider also A3 as a complimentary solution, the popularity score of W is 48. Both solutions are acceptable.

- END -