Data Mining

Cheat Sheet

Count Matrix for a single attribute A_1

	$A_1 = 1$	$A_1 = 0$
C_0	#	#
C_1	#	#

如何计算 $GINI_{SPLIT}$ 见下文

Confusion Matrix

		Predicted Class	
		Class = Yes	Class = No
Actual Class	Class = Yes	TP	FN
	Class = No	FP	TN

TP = true positive
FN = false negative
FP = false positive
TN = true negative

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Limitation of Accuracy
- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Basic Statistical Description of Data

Mean $\frac{1}{n}\sum_{i=1}^{n} x_i$

Median 中位数

Mode 在数据中出现/次数最多的值

Empirical relation on unimodal data: mean - mode $\approx$ 3(mean - median)

Symmetric vs. Skewed Data

Symmetric : mode = medium

positively skewed : mode < medium

negatively skewed : mode > medium

Measuring the Dispersion of Data

Percentiles given a list of N ordered values (least to greatest), and number p between 0 and 100

从小到大，在随也会用到

the p-th percentile $X_{p\%}$ is the smallest value in the ordered values such that at least p% of the values are less than or equal to $X_{p\%}$

Example 30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110 (12 numbers)

- i-th percentile: the $\lceil \frac{N}{100} \cdot i \rceil$-th value
- 10-th percentile:
  - $\lceil \frac{12}{100} \cdot 10 \rceil = 2$: 36 ← 第2个 value
- 20-th percentile:
  - $\lceil \frac{12}{100} \cdot 20 \rceil = 3$: 47 ← 第3个 value

Quartiles, outliers and boxplots

Quartiles : Q_1 (25^{th} percentile) Q_2 (2 medium) Q_3 (75^{th} percentile)

Inter-quartile range : $IQR = Q_3 - Q_1$

Five number summary : min Q_1 Q_2 Q_3 max

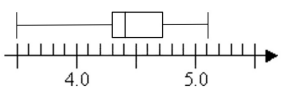
Outlier : usually, a value more than 1.5 x IQR above Q_3 or below Q_1

Example 3.5, 4.1, 4.2, 4.3, 4.3, 4.4, 4.4, 4.4, 4.4, 4.5, 4.5, 4.6, 4.7, 4.8, 4.9, 5.0, 5.1 [17 sorted values]

Q_1=4.3 Q_2 = 4.4 Q_3=4.7

IQR=4.7-4.3=0.4

Boxplot:



Variance : population variance vs. sample variance

Population variance considers the variance for a given population of size N

Example: Calculate the variance of the midterm score of FTEC4003. The population is the students enrolled into FTEC4003

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

Sample variance considers the variance of a extremely large (possibly infinite) population such that it is impossible to get the information of every record. Can only sample from the population.

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2 = \frac{1}{n-1}\left[\sum_{i=1}^{n}x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n}x_i\right)^2\right]$$

Example : Example data: 30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110

- Mean = 58
- $\sigma^2 = \frac{1}{12}(30^2 + 36^2 + 47^2 + \cdots + 110^2) - 58^2 \approx 379.17$
- $s^2 = \frac{1}{11}\left(30^2 + 36^2 + 47^2 + \cdots + 110^2 - \frac{1}{12}\cdot(58\cdot 12)^2\right) \approx 413.64$

Standard deviation σ or s : square root of the variance

Types of Data

Assign Weight : TF - IDF Scheme

Two heuristics

A word is more important if it occurs more frequently in a document
Term Frequency $tf(t,d)$ count of word t in doc d

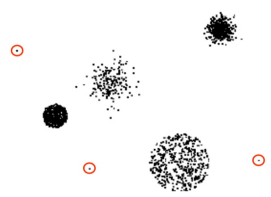
A word is more discriminative if it occurs only in fewer documents
IDF (Inverse Document Frequency)
- Assign higher weights to the rare terms
- $IDF(t) = \log_{10}\left(\frac{N+1}{df(t)}\right)$
  - N : Total number of docs in collection
  - $df(t)$: number of docs containing term t

TF - IDF Weighting scheme : $W(t,d) = TF(t,d) \times IDF(t)$

Data Quality

Outliers: data objects with characteristics that are considerably different from most of the other data objects in the dataset

- **Case 1:** Outliers are noise that interferes with data analysis
- **Case 2:** Outliers are the goal of our analysis
  - Credit card fraud
  - Intrusion detection



Binning Methods for Data Smoothing (to handle noisy data)

1st sort data and partition into (equal-frequency) bins

2nd can smooth by bin means / smooth by bin median / smooth by bin boundaries

Example:

- Sorted data for price (in dollars):
  4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
  - Partition into equal-frequency (equi-depth) bins:
    - Bin 1: 4, 8, 9, 15
    - Bin 2: 21, 21, 24, 25
    - Bin 3: 26, 28, 29, 34
  - Smoothing by bin means:
    - Bin 1: 9, 9, 9, 9
    - Bin 2: 23, 23, 23, 23
    - Bin 3: 29, 29, 29, 29
  - Smoothing by bin boundaries:
    - Bin 1: 4, 4, 4, 15
    - Bin 2: 21, 21, 25, 25
    - Bin 3: 26, 26, 26, 34

Similarity and Distance

Similarity / Dissimilarity between p and q . p and q are the attribute values for 2 data objects

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Similarity and dissimilarity

Dissimilarities between Data Objects

Euclidean Distance : $dist(\vec{x}, \vec{y}) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$

n: the number of dimensions (attributes)

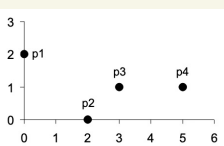
x_k & y_k: the k^{th} attributes (components) of data objects x and y

Minkowski Distance : $dist(\vec{x}, \vec{y}) = \sqrt[r]{\sum_{k=1}^{n} |x_k - y_k|^r}$ r : a parameter , we also call it Lr-norm

Special cases

- **Special cases:**
  - $r = 1$. Manhattan Distance (city block distance)
  - $r = 2$. Euclidean distance
  - $r = \infty$. Supremum
    - The *maximum difference* between any component of the two vectors: $\max(|x_1 - y_1|, |x_2 - y_2|, \cdots, |x_k - y_k|)$

Example



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Distance Matrix

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Issues with Minkowski Distance:

variables have different scales: income & age

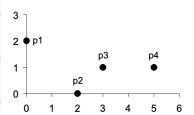
correlations between some attributes: income & credit card limit

Mahalanobis Distance: $\text{mahalanobis}(x, y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)}$

Σ is the covariance matrix of all the input data X $\Sigma_{j,k} = \frac{1}{n-1} \sum_{i=1}^{n} (X_{ij} - \bar{X}_j)(X_{ik} - \bar{X}_k)$

Deficiency: does not scale when the number of records is too large

Example:

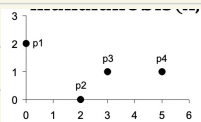


point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

$\Sigma_{1,1} = \frac{1}{3} \times ((0-2.5)\cdot(0-2.5) + (2-$

$\Sigma_{1,2} = \frac{1}{3} \times ((0-2.5)\cdot(2-1) + (2-$

$$\Sigma = \begin{bmatrix} \frac{13}{3} & -\frac{2}{3} \\ -\frac{2}{3} & \frac{2}{3} \end{bmatrix}$$



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

$$\Sigma = \begin{bmatrix} \frac{13}{3} & -\frac{2}{3} \\ -\frac{2}{3} & \frac{2}{3} \end{bmatrix} \Rightarrow \Sigma^{-1} = \begin{bmatrix} \frac{3}{11} & \frac{3}{11} \\ \frac{3}{11} & \frac{39}{22} \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

determinant

$$\mathbf{p_1} - \mathbf{p_2} = [-2,2]$$

$$\text{mahalanobis}(\mathbf{p_1}, \mathbf{p_2}) = \sqrt{[-2,2] \begin{bmatrix} \frac{3}{11} & \frac{3}{11} \\ \frac{3}{11} & \frac{39}{22} \end{bmatrix} \begin{bmatrix} -2 \\ 2 \end{bmatrix}} = \sqrt{6}$$

Similarity Between Binary Vectors

Compute similarities using the following quantities

f_{01} = the number of attributes where p was 0 and q was 1

f_{10} = p was 1 and q was 0

f_{00} = p was 0 and q was 0

f_{11} = p was 1 and q was 1

Simple Matching and Jaccard Coefficients:

SMC = number of matches / number of attributes

$= (f_{11} + f_{00}) / (f_{01} + f_{10} + f_{11} + f_{00})$

Jaccard = number of 11 matches / number of non-zero attributes

$= (f_{11}) / (f_{01} + f_{10} + f_{11})$

Example.

$p =$ [1 0 0 0 0 0 0 0 0 0]
$q =$ [0 0 0 0 0 0 1 0 0 1]

$f_{01} = 2$ (the number of attributes where p was 0 and q was 1)
$f_{10} = 1$ (the number of attributes where p was 1 and q was 0)
$f_{00} = 7$ (the number of attributes where p was 0 and q was 0)
$f_{11} = 0$ (the number of attributes where p was 1 and q was 1)

$SMC = (f_{11} + f_{00}) / (f_{01} + f_{10} + f_{11} + f_{00}) = (0+7) / (2+1+0+7) = 0.7$

$Jaccard = (f_{11}) / (f_{01} + f_{10} + f_{11}) = 0 / (2 + 1 + 0) = 0$

Cosine Similarity: $\cos(d_1, d_2) = (d_1 \cdot d_2) / \|d_1\| \cdot \|d_2\|$ here "·" indicates dot product

Measure of the cosine of the angle between the two vectors

Example:

Compute the cosine similarity between d_1 and d_2

This is a data mining course.

d_1=(0.602, 0.301, 0, 0.125, 0, 0, 0)

We are studying text mining. Text mining is a subfield of data mining.

d_2=(0, 0.301, 0, 0.375, 0.602, 0.602, 0.602)

- $d_1 \cdot d_2 = 0.602 \cdot 0 + 0.301 \cdot 0.301 + 0 \cdot 0 + 0.125 \cdot 0.375 + 0 \cdot 0.602 + 0 \cdot 0.602 + 0 \cdot 0.602 \approx 0.137$
- $||d_1|| = \sqrt{0.602^2 + 0.301^2 + 0.125^2} = 0.685$
- $||d_2|| = \sqrt{0.301^2 + 0.375^2 + 0.375^2 + 3 \cdot 0.602^2} = 1.148$
- $\cos(d_1, d_2) = \frac{d_1 \cdot d_2}{||d_1|| \cdot ||d_2||} = \frac{0.137}{0.685 \cdot 1.148} \approx 0.174$

Extended Jaccard Coefficient $EJ(x, y) = \frac{x \cdot y}{\|x\|^2 + \|y\|^2 - x \cdot y}$

Variation of Jaccard for continuous or count attributes

Example

d_1=(0.602, 0.301, 0, 0.125, 0, 0, 0)
d_2=(0, 0.301, 0, 0.375, 0.602, 0.602, 0.602)

- From previous slide, we know $d_1 \cdot d_2 = 0.137$, $||d_1|| = 0.685$, $||d_2|| = 1.148$
- $EJ(d_1, d_2) = \frac{0.137}{0.685^2 + 1.148^2 - 0.137} = 0.083$

Correlation Analysis (Categorical Data): χ^2 (chi-square test) $\chi^2 = \sum \frac{(\text{Observed} - \text{Expected})^2}{\text{Expected}}$

χ^2 value ↑, more likely the variables are related

Correlation does not imply causality

Example:

	Play chess	Not play chess	Sum (row)
Like science fiction	250 (90)	200 (360)	450
Not like science fiction	50 (210)	1000 (840)	1050
Sum (col.)	300	1200	1500

(括号中的 expected value 是用 Sum(row 和 col.) 中的比例进行计算得到的)

χ^2 calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} = 507.93$$

- If χ^2> (the number of different atributes in rows-1)*(the number of different attributes in columns -1)
  - The attributes are not independent.

Pearson's Correlation: $corr(p, q) = \frac{cov(p, q)}{std(p) * std(q)} = \frac{S_{pq}}{S_p S_q}$

$$cov(p, q) = S_{pq} = \frac{1}{n-1} \sum_{k=1}^{n} (p_k - \bar{p})(q_k - \bar{q})$$

$$std(p) = S_p = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (p_k - \bar{p})^2} \qquad \bar{p} = \frac{1}{n} \sum_{k=1}^{n} p_k$$

Example:

Perfect Correlation

- Correlation is always in the range -1 and 1. A correlation of value 1 (-1) means that x and y have a perfect positive (negative) linear relationship, i.e., $x = a \cdot y + b$, where a and b are constants.
  - The follow two sets of x and y indicate two cases of correlation -1 and +1, respectively

$x = (-3, 6, 0, 3, -6)$
$y = (1, -2, 0, -1, 2)$
$corr(x, y) = -1$

$x = (3, 6, 0, 3, 6)$
$y = (1, 2, 0, 1, 2)$
$corr(x, y) = 1$

Drawback: only capture linear relationships

Mutual Information: nonlinear relation between attributes

measures the info that X & Y share: how much knowing one reduces uncertainty about the other

Defined based on entropy

Entropy: $H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$

where $H(X)$ is the entropy of X

X is a variable (event), with n possible values (outcomes), $x_1, x_2 \dots x_n$ each outcome having probability $p_1, p_2 \dots p_n$

$H(X)$ is between 0 and $\log_2 n$ and is measured in bits

entropy is a measure of how many bits it takes to represent an observation of X on average

Example:

For a coin with probability p of heads and probability $q = 1 - p$ of tails

$$H = -p \log_2 p - q \log_2 q$$

- For $p = 0.5, q = 0.5$ (fair coin) $H = 1$
- For $p = 1$ or $q = 1$, $H = 0$

Mutual Information: $I(X, Y) = H(X) + H(Y) - H(X, Y)$

Here $H(X, Y) = -\Sigma_i \Sigma_j p_{ij} \log_2 p_{ij}$, where p_{ij} is the probability that the i^{th} value of X and the j^{th} value of Y occur together

Example:

Student Status	Count	p	$-p\log_2 p$
Undergrad	45	0.45	0.5184
Grad	55	0.55	0.4744
Total	100	1.00	0.9928

Grade	Count	p	$-p\log_2 p$
A	35	0.35	0.5301
B	50	0.50	0.5000
C	15	0.15	0.4105
Total	100	1.00	1.4406

Student Status	Grade	Count	p	$-p\log_2 p$
Undergrad	A	5	0.05	0.2161
Undergrad	B	30	0.30	0.5211
Undergrad	C	10	0.10	0.3322
Grad	A	30	0.30	0.5211
Grad	B	20	0.20	0.4644
Grad	C	5	0.05	0.2161
Total		100	1.00	2.2710

$$I(student\ status, Grade) = 0.9928 + 1.4406 - 2.2710 = 0.1624$$

Combining Similarities for Different Attributes

Sometimes attributes are of many different types, but an overall similarity is needed.

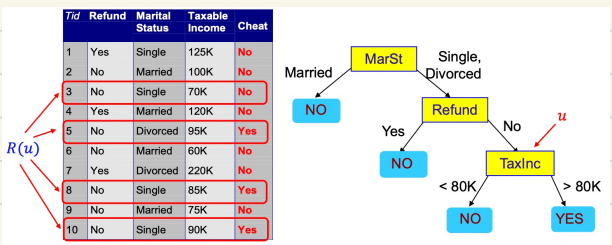
- For the k^{th} attribute, compute a similarity $s_k(\mathbf{x}, \mathbf{y})$ in range [0, 1].
- Define an indicator variable, δ_k, for the k^{th} attribute as follows:
  - δ_k = 0 if the k^{th} attribute is an asymmetric attribute and both objects have a value of 0, or if one of the objects has a missing value for the kth attribute
  - δ_k = 1 otherwise
- Compute $$\text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^{n} \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^{n} \delta_k}$$
- May not want to treat all attributes the same.
  - Use non-negative weights ω_k
  - $similarity(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^{n} \omega_k \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^{n} \omega_k \delta_k}$

Decision Tree

Definitions: Given a training set R, a node u in decision tree T. we define R(u) to be the set of training records such that:

1: If u is the root, R(u) is the set of all the training records

2: Otherwise, R(u) is the set of training records that satisfy the attribute condition (test) of all the edges from the root to u.



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Decision Tree Induction : Hunt's Algorithm

Algorithm Hunt(R)

Step 1: Base case, immediately return the decision tree:
- (i) If all training records in R belongs to the same class y_t, then return a leaf node with this class.
- (ii) If all records in R have the same attribute values, return a leaf node whose class label is the majority of records in R.

Step 2: Find the "best" attribute and test condition to split the data into smaller subset, create a node t for this attribute test. ← remains to explain

Step 3: For each subset $R_1, R_2, \cdots, R_i$, invoke $Hunt(R_1), Hunt(R_2) \cdots, Hunt(R_i)$ to create i subtrees, denote their root nodes as $t_1, t_2, \cdots t_i$. Connect t and $t_1, t_2, \cdots t_i$.

Step 4: Return the decision tree rooted at t

How to specify the attribute test condition?

How to determine the best split?

Methods for Expressing Test Conditions

Depends on attribute types:

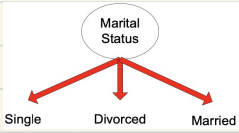
- Binary
- Nominal (标称)
- Ordinal (序数)
- Continuous

Depends on number of ways to split

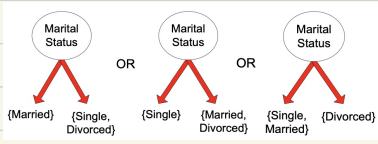
- 2-way split
- multi-way split

Test Condition for Nominal Attributes

Multi-way split: use as many partitions as distinct values



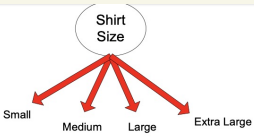
Binary split: divides values into 2 subsets



Test Condition for Ordinal Attributes

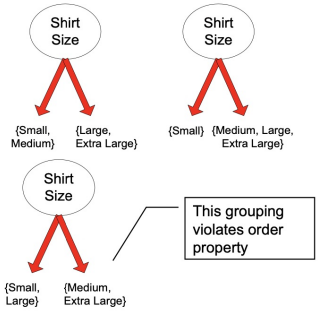
Multi-way split:

- Use as many partitions as distinct values

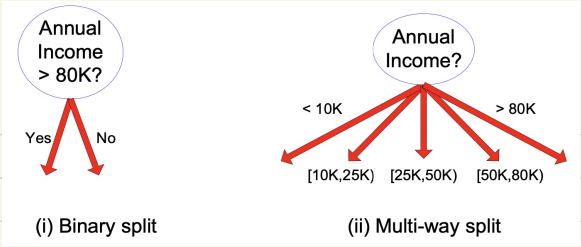


Binary split:

- Divides values into two subsets
- Preserve order property among attribute values



Test Condition for Continuous Attributes



Splitting Based on Continuous Attributes

Different ways of handling

Discretization: to form an ordinal categorical attribute. Ranges can be found by equal interval bucketing, equal frequency bucketing (percentile), or clustering

Static – discretize once at the beginning

Dynamic – repeat at each node

Binary Decision: $(A < v)$ or $(A \geq v)$

consider all possible splits and finds the best cut

compute intense

How to determine the Best Split?

Greedy approach: node with purer class distribution are preferred

Measures of Node Impurity *

Gini Index

Entropy

Misclassification error

*Remark: The three measures are adopted in different decision tree algorithms. For instance, CART adopts Entropy while ID3 adopts GINI index.

The choice of impurity measures usually has little affect to the performance of the same decision tree algorithm.

Finding the Best Split

1. Compute impurity measure (I_{before}) before splitting
2. Compute impurity measure (I_{after}) after splitting
    1. Compute impurity measure of each child node
    2. I_{after} is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain:
    $G = I_{before} - I_{after}$

    or equivalently, lowest impurity measure after splitting

Gini Index: for a given node t $GINI(t) = 1 - \sum_j [p(j|t)]^2$

Where $p(j|t)$ is the relative frequency of class j in $R(t)$

Maximum $(1 - 1/n_c$, n_c is the number of classes) when records are equally distributed among all classes

Minimum (0) when all records belong to one class

Example:

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)² – (5/6)² = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)² – (4/6)² = 0.444

for a single node

Splitting Based on GINI

When a node t is split into k partitions (children) $t_1, t_2 \dots t_k$, the quality of split is computed as

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(t_i)$$

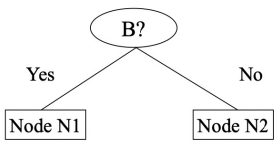
Where n_i is the number of records in partition t_i

n is the number of records in t

Example:

Binary Attributes : Computing Gini Index

- Splits into two partitions
- Effect of Weighting partitions:
  - Larger and Purer Partitions are sought for.



	Parent
C1	7
C2	5
Gini = 0.486	

Gini(N1)
= 1 – $(5/6)^2$ – $(1/6)^2$
= 0.278

Gini(N2)
= 1 – $(2/6)^2$ – $(4/6)^2$
= 0.444

Count matrix

	N1	N2
C1	5	2
C2	1	4
Gini=0.361		

Weighted Gini of N1 N2
= 6/12 * 0.278 +
6/12 * 0.444
= 0.361

Gain = 0.486 – 0.361 = 0.125

Categorical Attributes : Computing GINI Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes : Computing GINI Index

- Use binary decisions based on one value
  - Too expensive if we consider multi-split
- Several choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, $A < v$ and $A \geq v$
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally inefficient! Repetition of work.

ID	Home Owner	Marital Status	Annual Income	Defaulted
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Annual Income ?

	≤ 80	> 80
Defaulted Yes	0	3
Defaulted No	3	4

- For efficient computation: for each attribute,
  - Sort the attribute on values
    - Split positions: midpoint of consecutive distinct values
  - Linearly scan these values, each time updating the count matrix and computing Gini index
  - Choose the split position that has the least Gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220

Split Positions →	65		72		80		87		92		97		110		122		172	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0
No	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1
Gini	0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.400	

Entropy: at a given node t: $\text{Entropy}(t) = -\sum_j p(j|t) \log p(j|t)$

where $p(j|t)$ is the relative frequency of class j at node t

Maximum ($\log n_c$, n_c is the number of classes) when records are equally distributed

Minimum (0) when all records belong to the same class.

Example:

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	1
C2	5

P(C1) = 1/6 P(C2) = 5/6

Entropy = $-(1/6)\log_2(1/6) - (5/6)\log_2(1/6) = 0.65$

for a single node

C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6

Entropy = $-(2/6)\log_2(2/6) - (4/6)\log_2(4/6) = 0.92$

Information Gain:

$$GAIN_{split} = \text{entropy}(t) - \sum_{i=1}^{k} \frac{n_i}{n} \text{Entropy}(i)$$

Parent node t is split into k partitions, and n_i is the number of records in subset i

Choose the split that achieves the maximum gain

Problem: Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure

Gain Ratio:

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{split}}{SplitINFO}, SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

where Parent node t is split into k partitions, and n_i is the number of records in subset i.

- Adjusts Information Gain by the entropy of the partitioning.
  - Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5 algorithm
  - Designed to overcome the disadvantage of Information Gain

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7

SplitINFO = 1.52

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3

SplitINFO = 0.72

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10

SplitINFO = 0.97

Classification Error: at node t: $Error(t) = 1 - \max_j p(j|t)$

where $p(j|t)$ is the relative frequency of class j in t

Measures misclassification error made by a node.

- Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying least interesting information
- Minimum (0) when all records belong to one class, implying most interesting information

Gain with classification error can be defined similarly

Example:

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Modal Evaluation

confusion matrix : see cheat sheet

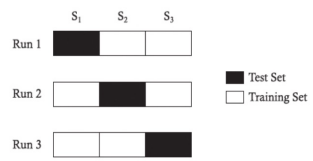
Accuracy, precision, recall, F-measure : see cheat sheet

Holdout (For test set with large number of samples)

- Reserve k% for training and remaining $(100 - k)$% for testing
  - E.g. reserve 2/3 for training and 1/3 for testing

Cross validation

- Partition data into k-disjoint subsets
- K-fold: train on k-1 partitions and test on the remaining
  - For test set with medium number of samples
- Leave-one-out: $k = n$, (n is the number of training records)
  - For test set with small number of samples



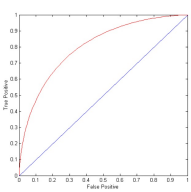
3-fold cross-validation

Bootstrap

- Sampling with replacement
- Sample n times from a set R of n training records.
  - We may sample a record multiple times.
  - Denote the sampled set as T. Then use $R \setminus T$ as the test set.
  - What is the probability that each record get sampled?
  - $1 - \left(1 - \frac{1}{n}\right)^n \approx 1 - \frac{1}{e} \approx 0.632$

(FP,TP):

- (0,0): declare everything to be negative class
  - t=1 in previous example
- (1,1): declare everything to be positive class
  - t=0 in previous example
- (0,1): ideal



- Diagonal line (blue line):
  - Random guessing
  - Below diagonal line: *prediction is opposite of the true class*

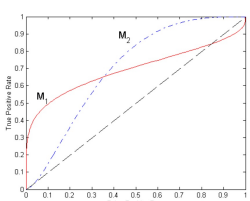
ROC Receiving Operating Characteristic

plots TP rate (on the y-axis)

FP rate (on the x-axis)

TP rate = True Positive / Positive

FP rate = False Positive / Negative



- No model may consistently outperform the other
  - M1 is better for small FPR
  - M2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

A model with larger area under ROC curve is usually preferred

KNN & Naïve Bayes

Instance Based Classfiers : K - Nearest Neighbors

Definition : K-nearest neighbors of a record x are data points that have the k smallest distance to x

Requirements : 1. a set of training records 2. distance metric 3. value of k

Classification 1. Compute distance to other training records
2. Identify k nearest neighbors
3. Use class labels of the nearest neighbors to determine the class label of the unknown record (e.g. by taking majority vote)

Weigh the vote according to distance d.
weight factor $w = \frac{1}{d^2}$

- 3-NN, two are positive with distance 0.5, one is negative with distance 0.2. Then the weighted vote of Yes is:

$$\frac{\left(\frac{1}{0.5}\right)^2 + \left(\frac{1}{0.5}\right)^2}{\left(\frac{1}{0.5}\right)^2 + \left(\frac{1}{0.5}\right)^2 + \left(\frac{1}{0.2}\right)^2} = \frac{8}{33}$$

Choosing k

k is too small, sensitive to noise points
too large, neighborhood may include points from other classes.

How to choose a good value of k?

Determined experimentally by varying k and draw the ROC curve.

Comparison : KNN vs. Decision Tree

Pros of kNN

- Easy to implement
- Incremental addition of training data trivial

Cons

- k-NN classifiers are **lazy learners**, which do not build models **explicitly**. This can be relatively more **expensive** than eager learners, e.g., decision tree, which builds a model and then can discard training records, when **classifying** a test/unknown record.
- Unlike decision tree that attempts to find a **global** model that fits the entire input space, nearest neighbor classifiers make the prediction based on **local** information, which can be more susceptible to noise.

Bayes Classifier

Bayes Theorem $P(X|Y) = \frac{P(X,Y)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$

Using Bayes Theorem for Classification

$$P(Y \mid A_1 = X_1, A_2 = X_2, \cdots A_d = X_d) = \frac{P(A_1 = X_1, A_2 = X_2, \cdots, A_d = X_d \mid Y) \cdot P(Y)}{P(A_1 = X_1, A_2 = X_2, \cdots, A_d = X_d)}$$

Example: Bayesian classfier estimates 2 terms:

$P(Y = Y_{ES})$

$P(A_1 = X_1, A_2 = X_2 \cdots A_d = X_d \mid Y = Y_{ES})$

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	92K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$P(Y = Y_{ES})$: the percentage of training records that the cheat attribute is Y_{ES}

$P(Y = Y_{ES}) = \frac{3}{10} = 0.3$

$P(Y = NO) = \frac{7}{10} = 0.7$

Assume INDEPENDENCE among attributes A_i when the class is given

$$P(A_1 = X_1, A_2 = X_2 \cdots A_d = X_d \mid Y = Y_{ES})$$
$$= P(A_1 = X_1 \mid Y = Y_{ES}) \cdot P(A_2 = X_2 \mid Y = Y_{ES}) \cdot \cdots \cdot P(A_d = X_d \mid Y = Y_{ES})$$

For discrete attributes:

$P(A_i = X_i \mid Y = Y_{ES})$: percentage of records that $A_i = X_i$ among all records whose class is Y_{ES}

e.g. $P(Refund = NO \mid Y = Y_{ES}) = 1$

For continuous attributes:

Discretization: Partition the range into bins

Replace continuous value with bin value

e.g. Partition into: [60K,70K), [70K,80K), [80K, 90K), [90K, 100K),[100K,110K), [110K, 120K), ...

Given a test record : $X = (\text{refund} = \text{No},\ \text{martial status} = \text{divorced},\ \text{taxable income} = 90k)$

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	92K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$\Pr[Y = Yes|X]$$
$$= \Pr[Refund = No|Y = Yes]$$
$$\cdot \Pr[martial\ status = Divorced|Y = Yes]$$
$$\cdot \Pr[taxable\ income = 90K|Y = Yes] \cdot \frac{\Pr[Y = Yes]}{\Pr[X]}$$
$$= 1 \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{3}{10} \cdot \frac{1}{\Pr[X]} = \frac{1}{15} \cdot \frac{1}{\Pr[X]}$$

$$\Pr[Y = No|X]$$
$$= \Pr[Refund = No|Y = No]$$
$$\cdot \Pr[martial\ status = Divorced|Y = No]$$
$$\cdot \Pr[taxable\ income = 90K|Y = No] \cdot \frac{\Pr[Y = No]}{\Pr[X]}$$
$$= \frac{4}{7} \cdot \frac{1}{7} \cdot \frac{1}{7} \cdot \frac{7}{10} \cdot \frac{1}{\Pr[X]} = \frac{2}{245} \cdot \frac{1}{\Pr[X]} < \Pr[Y = Yes|X]$$
$$= \frac{1}{15} \cdot \frac{1}{\Pr[X]}$$

Therefore, the Naïve Bayes classifier returns Yes.

Linear Classfiers

Perceptron learn a linear function $h(x) = w \cdot x - t$ s.t. it separates all training records by the binary class

Find a vector w and a constant t such that for any record x belonging to the set R of linearly separable records, it satisfies that:

$h(x) = w \cdot x - t > 0$ if the class label of x is Yes.

$h(x) = w \cdot x - t < 0$ if the class label of x is No.

Given an unknown test record y, how to make prediction?

If $h(y) > 0$, predict the class label to be Yes If $h(y) < 0$, predict the class label to be No

Example:

Given four three-dimensional training records $r_1 = (1,1,1), r_2 = (0,0,1), r_3 = (0,1,1), r_4 = (-1,-1,0)$, with their class labels to be Yes, No, Yes, and No, respectively,

- $h(x) = w \cdot x - t$ where $w = (0.3,0.3,0.3)^T$ and $t = 0.4$ linearly separates the four training records.
- To predict the class label of record $r = (1,0,1)$, the class is Yes since $h(r) = w \cdot r - t = 0.6 - 0.4 = 0.2 > 0$

How to find a valid weight vector w:

For ease of exposition, we first assume that $t = 0$.

- Recap $h(x) = w \cdot x - t$. We consider the case when $t \neq 0$ later

Algorithm:

- Initialization: make a random guess for w, say $(0,0,\cdots,0)$.
- Repeat:
- If there exists a point x from the training set R that violates the requirement according to the current value of w, update
  - if the label of x is Yes: $w = w + x$
  - Else (the label of x is No): $w = w - x$
- Until all points satisfy the requirement
  - Requirement: $h(x) > 0$ if x has label Yes. $h(x) < 0$ if x has label No.

Example:

	d_1	d_2	class
x_1	1	0	Yes
x_2	0	-1	No
x_3	0	1	Yes
x_4	-1	0	No

- Initial: $w = (0,0)$
- Record 1 violates the condition since the class label is Yes while $w \cdot x_1 \leq 0$, update
  - $w = w + x_1 = (0,0) + (1,0) = (1,0)$
- Record 2 violates the condition, update
  - $w = w - x_2 = (1,0) - (0,-1) = (1,1)$
- No record violates the condition. Terminate and return $w = (1,1)$.

Generalization to t≠0

Trick: map each record $x = (x_1, x_2, \cdots, x_d) \in R$ form d-dimensional to $(d+1)$ −dimensional $x' = (x_1, x_2, \cdots, x_d, 1)$, denote the set as R'.

Theorem: R is linearly separable with $t \neq 0$ if and only if R' is linearly separable with $t = 0$

- How to solve the case when $t \neq 0$?
  - Extend each record by the above mentioned trick by extending a d-dimensional record to $(d+1)$-dimensional record
  - Then apply the Perceptron algorithm.

Example:

Given four points $x_1 = (-1,1), x_2 = (1,-1), x_3 = (0,1)$, and $x_4 = (1,0)$ with x_1, x_2 having No class and x_3, x_4 having Yes class, show how to apply the Perceptron algorithm to find the weight vector w and threshold t.

We map each point to $x'_1 = (-1,1,1)$, $x'_2 = (1,-1,1), x'_3 = (0,1,1)$ and $x'_4 = (1,0,1)$

Invoke the Perceptron algorithm with $w' = (0,0,0)$

Derive the weight vector $w' = (3,3,-1)$

- So, $w = (3,3)$, $t = 1$
- $h(x) = w \cdot x - 1$

Support Vector Machines / ANN / Ensemble Methods

see printed files

Centroid-based Clustering

K-Center

a set of records R, where each record mapped to a d-dimension point, denoted as $P = \{p_1, p_2 \dots p_n\}$

Let k be the number of clusters desired

Identify k points $C = \{c_1, c_2 \dots c_k\}$ (not necessarily in P) called the centroid.

Then, partition data points in P into k clusters $P_1, P_2, \dots, P_k$ such that:

P_x belongs to P_i if it is closest to centroid c_i compared to other centroids

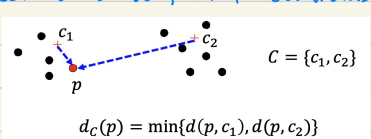
If P_x has equal-distances to different centroids, say c_i and c_j, assign to either P_i or P_j

Euclidean distance if no context is given

Define $d_c(p)$ as the minimum distance from p to the points in C

$$r(C) = \max_{p \in P} d_c(p)$$

k-center aims to find k centroids that minimizes the radius r, denoted as $r(C)$



But current solution is NP hard, thus there's an approximation.

Let C^* be the set of k centroids that achieves the minimum radius $r(C^*)$

Let C be the set of k centroids returned by an algorithm

If $r(C) \leq \rho \cdot r(C^*)$, we say the algorithm returns ρ-approximate solution

2- approximate solution

$\rho = 2$

Given a set $P = \{p_1, p_2, p_3, \cdots, p_n\}$ of n points

Let C be the centroid set, and initially to be empty.

- First: Randomly pick a point in P and add to C.
- Repeat:
  - Add a point $o \in P$ that maximizes $d_C(o)$ to C.
- Until C includes k points
- Return C as the centroid set

Simple and efficient: run in $O(k \cdot n)$ time

Example:

A set of 1-dimensional points $p_1 = 1, p_2 = 2, p_3 = 5, p_4 = 10, p_5 = 20, p_6 = 40, p_7 = 50, p_8 = 70$

$P = \{p_1, p_2, \cdots, p_8\}, k = 3$

Add a random point to C.

Assume that we add p_4 to C. Then $C = \{p_4 = 10\}$

- Next, $d_C(p_8)$ is maximum
  - Add p_8 to C. Then $C = \{p_4 = 10, p_8 = 70\}$.
- Next, $d_C(p_1) = 9, d_C(p_2) = 8, d_C(p_3) = 5, d_C(p_5) = 10, d_C(p_6) = 30, d_C(p_7) = 20$.
  - Add p_6 to C. Then $C = \{p_4 = 10, p_6 = 40, p_8 = 70\}$
- Return C. What is the radius of C, i.e., $r(C)$?

Given a set $P = \{p_1, p_2, p_3, \cdots, p_n\}$ of n points

Let C be the centroid set, and initially to be empty.

- First: Randomly pick a point in P and add to C.
- Repeat:
  - Add a point $o \in P$ that maximizes $d_C(o)$ to C.
- Until C includes k points
- Return C as the centroid set

Simple and efficient: run in $O(k \cdot n)$ time

K-Means

Define $d(p_1, p_2)$ as the distance of two points

$d_C(p)$ as the minimum distance from p to the points in C

The cost of C, denoted as $\phi(C)$ is:

$\phi(C) = \sum_{p \in P} (d_C(p))^2$

aka. sum of the square error (SSE)

Given a constant k, the goal of k-means clustering is to choose a size k centroid set C that minimizes $\phi(C)$

NP-hard

Algorithm:

1st. Select k records as the initial centroids set C

2nd Repeat:

Form k clusters by assigning points to the closest centroid in C

Denote C as C_{old}. Next generate new centroid set C.

Re-compute the centroid set C such that $\phi(C) < \phi(C_{old})$

3rd Until the centroid does not change and return C

Reducing the Cost of C: consider the case when the distance measure is Euclidean distance, i.e. L_2-norm

Then we update the centroid as follows

Let $P_i = \{x_1, x_2, \dots, x_l\}$ be the cluster corresponding to centroid c_i

Update c_i as the average of $x_1, x_2, \dots, x_l$:

$$c_i = \frac{1}{l}\sum_{j=1}^{l} x_j$$

Example:

Example, $x_1 = (2,3), x_2 = (5,4), x_3 = (3,4), x_4 = (4,3)$

- Then, c_i is updated as $\frac{1}{4}(x_1 + x_2 + x_3 + x_4)$

$$= \frac{1}{4}(2+5+3+4, 3+4+4+3) = (3.5, 3.5)$$

The distance can be other metrics, e.g., L_1-norm.

Converge very fast, usually terminates in a few iterations.

- Often the stopping condition is changed to 'Until relatively few points change clusters'

Complexity: $O(n \cdot k \cdot I \cdot d)$

- n = number of points, K = number of clusters, I = number of iterations, d = number of attributes

Limition & Issues see printed files

K-Means++: to solve the Initial Centroid Problem

If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.

- Chance is relatively small when K is large
- If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if K = 10, then probability is
  - $\frac{10!}{10^{10}} = 0.00036$
- Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't

k-means++ [1]:

- Carefully select the initial centroids
- Guarantees an approximation ratio $O(\log k)$ in expectation, where k is the number of centroids
- Proof omitted since it is rather involved

Differ with k-means in only the initial centroid selection.

K-means: random initialization

K-means++:

- Select one centroid from P uniformly at random add to C.
- Repeat k-1 steps:
  - Calculate the SSE ($\phi(C)$) of all the points with respect to C.

    $\phi(C) = \sum_{q \in P}(d_C(q))^2$
  - For each record p, select it with probability $\frac{(d_C(p))^2}{\phi(C)}$.
  - Let c be a record selected to the above probability distribution.
  - Add c to C.
- Return C

Hierarchical Clustering

Common Proximities for Clusters

Given two clusters C_1 and C_2

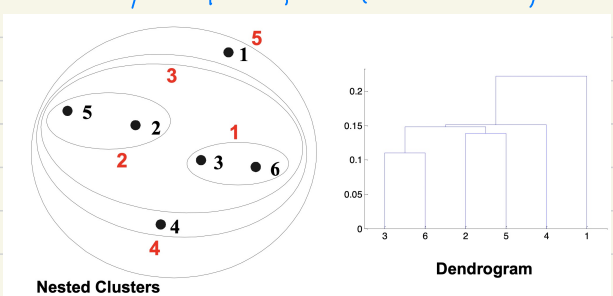
- $d_{min}(C_1, C_2) = \min_{p_1 \in C_1, p_2 \in C_2} d(p_1, p_2)$
- $d_{max}(C_1, C_2) = \max_{p_1 \in C_1, p_2 \in C_2} d(p_1, p_2)$
- $d_{mean}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p_1 \in C_1, p_2 \in C_2} d(p_1, p_2)$

Example: two clusters $C_1 = \{p_1 = (1,2), p_2 = (2,2)\}$ and $C_2 = \{p_4 = (-1,-1), p_5 = (-1,-2), p_6 = (-2,-2)\}$

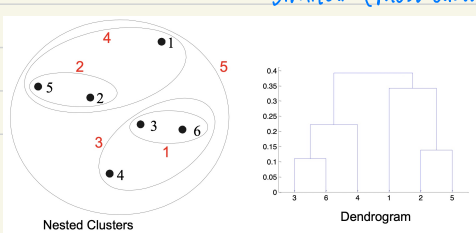
- $d(p_1, p_4) = \sqrt{2^2 + 3^2} = \sqrt{13}$, $d(p_2, p_4) = \sqrt{3^2 + 3^2} = \sqrt{18}$
- $d(p_1, p_5) = \sqrt{2^2 + 4^2} = \sqrt{20}$, $d(p_2, p_5) = \sqrt{3^2 + 4^2} = \sqrt{25}$
- $d(p_1, p_6) = \sqrt{3^2 + 4^2} = \sqrt{25}$, $d(p_2, p_6) = \sqrt{4^2 + 4^2} = \sqrt{32}$
- Then $d_{min}(C_1, C_2) = \sqrt{13}$,
  - $d_{max}(C_1, C_2) = \sqrt{32}$,
  - $d_{mean}(C_1, C_2) = \frac{1}{6}(\sqrt{13} + \sqrt{18} + \sqrt{20} + \sqrt{25} + \sqrt{25} + \sqrt{32})$

MIN: Single linkage clustering: Proximity of two clusters is based on the two closest points in the different clusters.

Determined by one pair of the points, i.e. by one link in the proximity graph



MAX: Complete Linkage Clustering: Similarity of two clusters is based on the two least similar (most distant) points in the different clusters



MEAN: Mean / Group average: Similarity of two clusters is the average of pairwise proximity between points in two clusters

between Single and Complete Link

Cost Analysis:

$O(N^2)$ space since it uses the proximity matrix.

- N is the number of records.

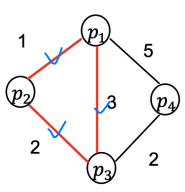
$O(N^3)$ time in many cases

- There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
- Complexity can be reduced to $O(N^2 \log(N))$ time for some cases
  - Connecting with minimum spanning tree

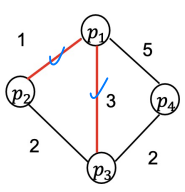
A minimum spanning tree based approach

Recap: Minimum Spanning Tree (MST)

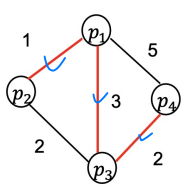
- Given an undirected graph $G = (V, E)$ with n nodes. Let T be a set of $n - 1$ edges of G with no cycles. If T includes no cycle, it is a spanning tree of G.
  - The cost of T is then the sum of the weights of all edges on T



Not a spanning tree, including a cycle

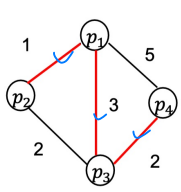


Not a spanning tree

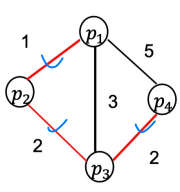


A spanning tree with cost 6

- A minimum spanning tree (MST) is a spanning tree such that the cost is minimum.



A spanning tree with cost 6



A minimum spanning tree: cost is 5

Kruskal's Algorithm: Given an undirected graph $G=(V, E)$ with n nodes and m edges

Goal: find the minimum spanning tree

Initially T is empty

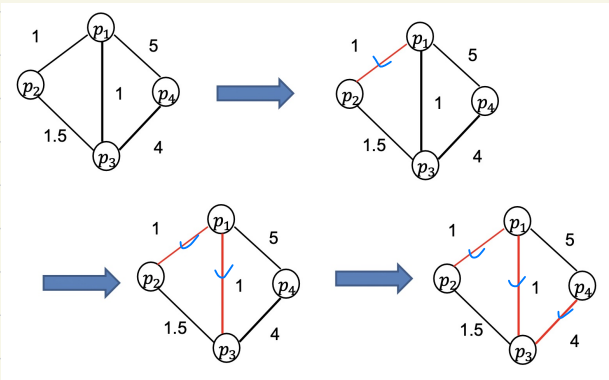
Repeat:

For all edges e that not in T and the addition of e does not create a cycle

Add the edge with the minimum cost to T

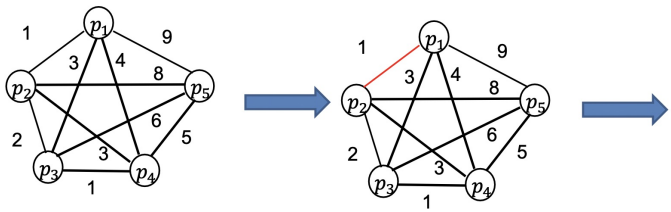
Till T contains n-1 edges

Example



Single-Linkage Clustering & MST

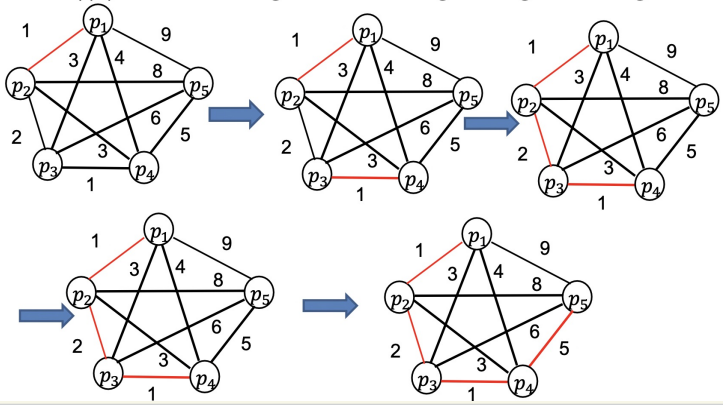
- Given 5 records $p_1 = 1, p_2 = 2, p_3 = 4, p_4 = 5, p_5 = 10$.
  - Show how to apply the Kruskal's algorithm to do single-linkage clustering



Create a graph where each node corresponds to a record, and each edge is a record pair with weight equal to their distance

Include a new edge with minimum cost but does not cause a cycle

- Given 5 records $p_1 = 1, p_2 = 2, p_3 = 4, p_4 = 5, p_5 = 10$.
  - Apply the Kruskal's algorithm to do single-linkage clustering



Density-Based Clustering & Cluster Evaluation

DBSCAN: Density-based Spatial Clustering of Applications with Noise

Two principles
- the area around a noise should be "sparse"
- if 2 points belong to the same cluster, it should be possible to go from one close point to another through only "dense" areas

Parameters in DBSCAN

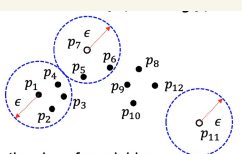
Input: a set P of d-dimensional points

Parameters
- ϵ: a distance threshold (to identify "close" points)
- MinPts: a constant integer (to define dense / sparse areas)

ϵ-neighbor of node p, denoted as $C(p, \epsilon)$
The set of nodes that are within distance ϵ to p (including p)

Example:

- $C(p_1, \epsilon) = \{p_1, p_2, p_3, p_4\}$
- $C(p_7, \epsilon) = \{p_5, p_6, p_7\}$
- $C(p_{11}, \epsilon) = \{p_{11}\}$



Black circled points: the size of ϵ-neighbor is no smaller than $MinPts = 4$

Different Types of Points in DBSCAN

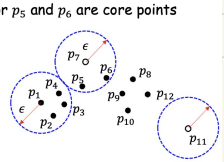
Core point: a point p such that $C(p, \epsilon)$ include at least MinPts points.

Border Point: a point p that is not a core point but its ϵ-neighbor include at least a core point

Noise Point: a point p that is neither a core point nor a border point.

Example

- What are the types of p_1, p_7 and p_{11} if $MinPts = 4$? Core point, border point or noise point?
  - p_1: core point
  - p_7: border point, its ϵ-neighbor p_5 and p_6 are core points
  - p_{11}: noise point



Algorithm: 1st Identify point types and remove all noise points

2nd Cluster core points

Core-point graph: Create an undirected graph by making each core point a node, and add an edge between two core points if they are ϵ-neighbors

Cluster two points p and q_n to the same cluster if they are in the same connected component, or equivalently, there exists a path from p to q_n (using BFS or DFS)

At the end: each core point belongs to exactly one cluster

3rd Assign border points

For a border point, find an arbitrary core point p_c in its ϵ-neighbor, assign this border point to the cluster of p_c

Example:

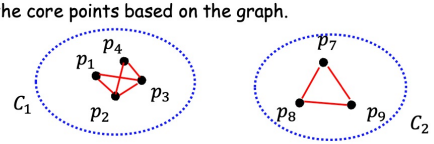
Given 10 records $p_1 = 1, p_2 = 2, p_3 = 3, p_4 = 4, p_5 = 7, p_6 = 9, p_7 = 12, p_8 = 13, p_9 = 14, p_{10} = 16$. Show how DBSCAN works with $\epsilon = 2$, $MinPts = 3$.

Step 1: point types:

- Core points: $p_1, p_2, p_3, p_4, p_7, p_8, p_9$
- Border points: p_{10}
- Noise points: p_5, p_6.
- Remove p_5 and p_6

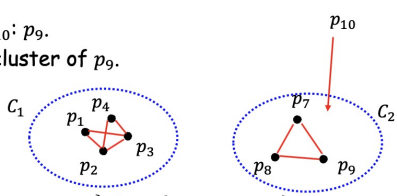
Step 2: construct an undirected graph for core points

- Construct the core-point graph
- Cluster the core points based on the graph.



Step 3: assign border points

- Border points: p_{10}
- The ϵ-neighbor of p_{10}: p_9.
  - Assign it to the cluster of p_9.



- Final cluster: $C_1 = \{p_1, p_2, p_3, p_4\}, C_2 = \{p_7, p_8, p_9, p_{10}\}$

Given n records, we require

- $O(n^2)$ space to store the ϵ-neighbor of each node and to construct the core-point graph

Cost Analysis:

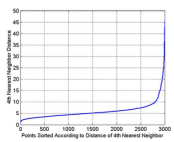
Time complexity: $O(n^2)$ running time.

- For each record, finding the ϵ-neighbor takes $O(n)$ time. We have n records, therefore $O(n^2)$ time
- Then, identifying core/border/noise points takes $O(n)$ time.
- Constructing the core-point graph takes $O(n^2)$
- Finding the clusters for each core-point takes $O(n^2)$
- Assigning the border points takes $O(n)$ cost
- In total $O(n^2)$ running time.

Determine ϵ and MinPts

Chossing ϵ: idea is that for points in a cluster, their k-th nearest neighbors are at roughly the same distance

- Noise points have the kth nearest neighbor at farther distance
- Plot sorted distance of every point to its kth nearest neighbor



Choosing $MinPts$:

- If $MinPts$ is too small, noise points may be included in the cluster
- If $MinPts$ is too large, small clusters will be labeled as noise.
- Original DBSCAN designer shows $MinPts = 4$ gets a good trade-off.

Cluster Evaluation

Cluster Validate 2 types

External (supervised) measure

Used to measure the extent to which cluster labels match externally supplied class labels.

Entropy

Internal (unsupervised) measure

Used to measure the goodness of a clustering structure without respect to external information.

Sum of Squared Error (SSE)

Cluster cohesion and cluster separation.

Entropy

After clustering, we might be able to further obtain additional class label information.

- The records in the same cluster should have similar class labels.

Entropy for clustering results

- Assume that there are L class labels.
- For each cluster P_i, calculate the entropy for each class.
  - $Entropy(P_i) = -\sum_{j=1}^{L} p_{ij} \cdot \log_2 p_{ij}$
  - p_{ij} is the fraction of records in cluster P_i that have class label j
- The total entropy is:
  - $Entropy = \sum_{i=1}^{k} \frac{n_i}{n} \cdot Entropy(P_i)$
  - k is the number of clusters, n_i is the number of records in P_i, and n is the number of input records P.

Example:

Given 6 records $p_1 = 1, p_2 = 2, p_3 = 3, p_4 = 12, p_5 = 13, p_6 = 14$ and a clustering result: $P_1 = \{p_1, p_2, p_3, p_4\}, P_2 = \{p_5, p_6\}$. The class labels are $Yes, No, Yes, Yes, No, No$. What is the entropy of the clustering result?

右和者是用这些 label 计算的

- $Entropy(P_1) = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 0.81$
- $Entropy(P_2) = 0$
- $Entropy = \frac{4}{6} \cdot 0.81 + \frac{2}{6} \cdot 0 = 0.54$

Cohesion and Separation

Cluster Cohesion: Measures how closely related are objects in a cluster

Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters

Example: Squared Error

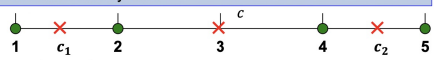
- Cohesion is measured by the within cluster sum of squares (SSE)
  - $SSE = \sum_{i=1}^{k} \sum_{x \in P_i} \left(d(c_i, x)\right)^2$
  - c_i is the centroid of cluster P_i by taking the mean of all points in P_i
- Separation is measured by the between cluster sum of squares (SSB)
  - $SSB = \sum_{i=1}^{k} |P_i| \left(d(c_i, c)\right)^2$
  - c_i is the centroid of cluster P_i and c is the mean of all points

Total sum of squares (TSS) is the sum of squares of the distance of each point to the mean of all points in the record set P.

- $TSS = \sum_{x \in P} \left(d(x, c)\right)^2 = \sum_{i=1}^{k} \sum_{x \in P_i} \left(d(x, c)\right)^2$
  - TSS only depends on the input record set P

Lemma 1: When Euclidean distance is used as the measure, $SSE + SSB$ is a constant no matter how many clusters are formed and how records are clustered. This constant is exactly TSS.

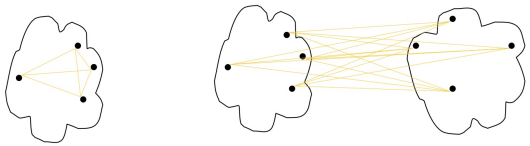
- Example



- When $k = 1$ $P_1 = \{1,2,4,5\}$. $c_1 = 3$, $c = 3$
  - $SSE = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$
  - $SSB = 4 \cdot (3-3)^2 = 0$
- How about when $k = 2$, $P_1 = \{1,2\}, c_1 = 1.5, P_2 = \{4,5\}, c_2 = 4.5, c = 3$?
  - $SSE = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$
  - $SSB = 2 \times (3-1.5)^2 + 2 \times (3-4.5)^2 = 9$

A proximity graph based approach can also be used for cohesion and separation.

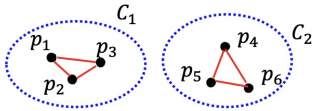
- Cluster cohesion is the sum of the weight of all links within a cluster.
- Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.



cohesion separation

Example:

Given 6 records $p_1 = 1, p_2 = 2, p_3 = 3, p_4 = 12, p_5 = 13, p_6 = 14$ and a clustering result: $C_1 = \{p_1, p_2, p_3\}, C_2 = \{p_4, p_5, p_6\}$. Calculate the cohesion and separation score using the proximity graph-based approach.



Cohesion:

$$d(p_1, p_2) + d(p_1, p_3) + d(p_2, p_3) + d(p_4, p_5) + d(p_4, p_6) + d(p_5, p_6)$$
$$= 1 + 2 + 1 + 1 + 2 + 1 = 8$$

Separation:

$$d(p_1, p_4) + d(p_1, p_5) + d(p_1, p_6) + d(p_2, p_4) + d(p_2, p_5) + d(p_2, p_6)$$
$$+ d(p_3, p_4) + d(p_3, p_5) + d(p_3, p_6)$$
$$= 11 + 12 + 13 + 10 + 11 + 12 + 9 + 10 + 11 = 99$$

Association Analysis : Apriori

Association Rule Mining

Concepts

Apriori algorithm

Concepts: Problem statement

Given a set T of transactions, where each transaction is a list of items

Find all rules that correlate the presence of one set of items with another set of items

Definition: Frequent Itemset

Input: a set $\mathcal{T}$ of transactions, where each transaction T is a set including several items. Let S be the set of all items appeared in $\mathcal{T}$.

- $S = \{Beer, Bread, Coke, Diaper, Eggs, Milk\}$

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

A set $\mathcal{T}$ of Transactions

Itemset

- A collection of one or more items
  - Example: {Milk, Bread, Diaper}

K-itemset

- An itemset that contains k items
- {Milk, Bread, Diaper} is a 3-itemset

A is a superset of B if $B \subseteq A$

Support 的 def ☆

Support of an itemset I: $support(I)$

- The number of transactions in $\mathcal{T}$ that is a superset of I
- $support(\{Milk, bread, diaper\}) = 2$ (即在集合 T 中有 2 个集合为 输入的 {Milk, bread, diaper} 的超集)

Definition: Association Rule

Association Rule

- An implication expression of the form $X \rightarrow Y$, where X and Y are non-empty itemsets, and $X \cap Y = \emptyset$ (交集为空)
  - Example: {Milk, Diaper} → {Beer}

Rule Evaluation metrics

- Support of an association rule $R = X \rightarrow Y$, denoted as $\sup(R)$, is the support of $X \cup Y$
  - Support of $\sup(\{Milk, Diaper\} \rightarrow \{Beer\}) = support(\{Milk, Diaper, Beer\}) = 2$
- Confidence of an association rule $R = X \rightarrow Y$
  - $conf(X \rightarrow Y) = \frac{support(X \cup Y)}{support(X)}$
  - $conf(\{Milk, Diaper\} \rightarrow \{Beer\})$
  - $= \frac{support(\{Milk, Diaper, Bear\})}{support(\{Milk, Diaper\})} = \frac{2}{3}$

Association 的 Support 和 Confidence 的 definition

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Association Rule Mining: Formal Definition

Association rule mining:

- Given a set $\mathcal{T}$ of transactions, and two constants minsup and minconf, we want to find all the association rules R such that

$$\sup(R) \geq minsup$$
$$conf(R) \geq minconf$$

Two-step Approach:

Two-step approach:

- Frequent Itemset Generation [困难]
  - Generate all itemsets whose support $\geq$ $minsup$
- Rule Generation
  - Generate high-confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation is still computationally expensive!

Apriori Algorithm:

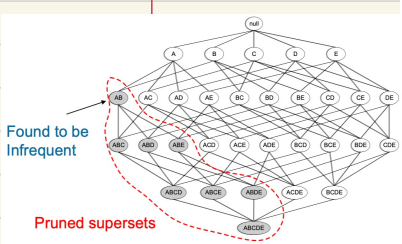
Apriori principle:

If an itemset I is frequent, then all of its subsets, i.e. any set $I' \subseteq I$, must also be frequent

- E.g., if $\{A, B, C\}$ is frequent, then $\{A\}, \{B\}, \{C\}, \{AB\}, \{BC\}, \{AC\}$ are frequent

If an itemset I is infrequent, then all of its supersets, i.e., any set $I \subseteq I'$, must also be infrequent

- E.g., if $\{A\}$ is infrequent, then $\{A, B\}, \{A, C\}$, etc. are infrequent.



Apriori principle holds due to the following property of the support measure:

$$X \subseteq Y, \sup(X) \geq \sup(Y)$$

- Support of an itemset never exceeds the support of its subsets
- This is known as the *anti-monotone* property of support

Denote $\mathcal{F}_k$ as the set of all frequent k-itemsets. Then, the set $\mathcal{F}$ of all frequent itemsets is equal to:

$$\mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \cdots \cup \mathcal{F}_{|S|}$$

- Where S is the set of all items appeared in $\mathcal{T}$.

Apriori Algorithm

Find $\mathcal{F}_1$ by scanning the set $\mathcal{T}$ of all transactions

For $k = 2$ to $|S|$

- Generate candidate set $\mathcal{C}_k$ based on F_{k-1}
- Maintain a counter $\mathcal{C}.counter$ for each candidate $\mathcal{C} \in \mathcal{C}_k$, initialize the counter to be 0 for all candidates.
- For each transaction $T \in \mathcal{T}$ // Scan all transactions
  - Generate all k-itemsets of T, denoted as $\mathcal{T}_k$.
  - Let $\mathcal{C} = \mathcal{T}_k \cap \mathcal{C}_k$
  - For each $\mathcal{C} \in \mathcal{C}$
    - Increment $\mathcal{C}.counter$ by 1.
- $\mathcal{F}_k = \{\mathrm{C} \in \mathcal{C}_k \mid \mathrm{C.counter} \geq minsup\}$

Let $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cdots \mathcal{F}_{|S|}$

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Generate candidate associate rules according to $\mathcal{F}$, verify the rules and output those with confidence $\geq minconf$

FP-Growth 算法

Drawback of conf.

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea → Coffee

Confidence ≅ P(Coffee|Tea) = 15/20 = 0.75

Confidence > 50%, meaning people who drink tea are more likely to drink coffee than not drink coffee

Rule seems reasonable.

But P(Coffee) = 0.9, which means knowing that a person drinks tea reduces the probability that the person drinks coffee!

⇒ Note that P(Coffee|$\overline{\text{Tea}}$) = 75/80 = 0.9375

Association Rule Evaluation
Pattern Evaluation
Computing Interestingness Measure

- Given X → Y or {X,Y}, information needed to compute interestingness can be obtained from a contingency table

Contingency table

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

$\bar{Y}$ indicates the case when a set S does not fully contain every element in Y, i.e., $Y \not\subseteq S$

Measures taking into account of statistical independency:

- Interest factor, or simply the interest, also known as the lift.
  - $Interest = \frac{P(X,Y)}{P(X)\cdot P(Y)}$
- Piatesky-Shapiro measure, or PS measure
  - $PS = P(X,Y) - P(X) \cdot P(Y)$

Example: #1

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75 $\frac{15}{20}$

but P(Coffee) = 0.9

⇒ Interest = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated) ↓ $\frac{P(\text{Coffee}|\text{Tea})}{P(\text{Coffee})}$

Example: #2

X	1	1	1	1	0	0	0	0	4
Y	1	1	0	0	0	0	0	0	2
Z	0	1	1	1	1	1	1	1	7

Support ratio: support/(# of transactions)

Rule	Support ratio	Confidence
$X \to Y$	25%	50%
$X \to Z$	37.5%	75%

Rule	Support ratio	Interest
$X \to Y$	25%	2
$X \to Z$	37.5%	0.9

Interest: $\frac{P(Z|X)}{P(Z)} = \frac{5/8}{7/8}$ $\frac{P(Y|X)}{P(Y)} = \frac{2/8}{2/8}$

$P(X,Y) = \text{Interest} \times P(X) \times P(Y)$

Ps measure for $X \to Y$ $X \to Z$

$Ps(X \to Y) = 2 \times \frac{1}{2} \times \frac{1}{4} \times \frac{1}{2} \times \frac{1}{4} = \frac{1}{16} \times \frac{1}{8} = \frac{1}{32}$

$Ps(X \to Z) = \frac{6}{7} \times \frac{8}{8} \times \frac{1}{2} \times \frac{7}{8} \times \frac{1}{2} = \frac{3}{4} \times \frac{1}{4} \times \frac{7}{8} = \frac{21}{16 \times 8} = \frac{21}{128}$