
BAYESIAN METHOD

Basic Idea_

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
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

Unknown record

$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$

Using Bayes Theorem for Classification

هر ویژگی و برچسب کلاس را به عنوان
متغیرهای تصادفی در نظر بگیرید

- Consider each attribute and class label as random variables
- Given a record with attributes

$$(X_1, X_2, \dots, X_d),$$

the goal : predict class Y

در اینجا مشاهداتمون یک فضای احتمالاتی داره یعنی میتوانیم بگیم احتمال اینکه وضعیت تا هل مجرد باشه و کلاس یا لبیل اخیریمون yes باشه چقدر؟
هدف: به کمک قوانین بیز شرایط داده ها را دریابیم و تصمیم گیری کنیم برای داده های جدید

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120K)$$

میخاییم احتمال پس یا no بودن رکورد اخیر را اندازه
بگیریم

اگریویوت هایی که داریم را به عنوان متغیر تصادفی در نظر میگیریم
داده های جدیدی که میخاییم پیش‌بینی کنیم مثل یه آزمایش است که براساس متغیرهای تصادفی که قبل مقدار گرفتن، نتیجه های این آزمایش چی میشه؟
هدف: پیش‌بینی کلاس و یا اخیرین اگریویوت

Tid	Refund	Marital Status	Taxable Income	Evade
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

Bayes Classifier

- A probabilistic framework for solving classification problems

- Conditional Probability:

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

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

- Bayes theorem:

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

Using Bayes Theorem for Classification

- Consider each attribute and class label as random variables
- Given a record with attributes

$$(X_1, X_2, \dots, X_d),$$

the goal : predict class Y

- Specifically, we want to find the value of Y that maximizes $P(Y| X_1, X_2, \dots, X_d)$
- Can we estimate $P(Y| X_1, X_2, \dots, X_d)$ directly from data?

Tid	Refund	Marital Status	Taxable Income	Evade
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

Using Bayes Theorem for Classification

- Approach:
 - compute posterior probability $P(Y | X_1, X_2, \dots, X_d)$ using the Bayes theorem

$$P(Y | X_1 X_2 \dots X_n) = \frac{P(X_1 X_2 \dots X_d | Y) P(Y)}{P(X_1 X_2 \dots X_d)}$$

- *Maximum a-posteriori:* Choose Y that maximizes $P(Y | X_1, X_2, \dots, X_d)$
- Equivalent to choosing value of Y that maximizes $P(X_1, X_2, \dots, X_d | Y) P(Y)$
- How to estimate $P(X_1, X_2, \dots, X_d | Y)$?

- Let X be a data sample with unknown class label
- Let Y be a hypothesis that X belongs to class C
- Classification is to determine $P(Y|X)$ (posteriori probability)

احتمال اولیه

$P(Y)$ (prior probability): the initial probability

E.g., X will buy computer, regardless of age, income, etc.

- $P(X)$: probability that sample data is observed
- $P(X|Y)$ (likelihood): the probability of observing the sample X, given that the hypothesis holds

E.g., Given that X will buy computer, the prob. that X is 31..40, medium income, etc

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

The diagram illustrates the components of Bayes' formula. The formula is $P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)}$. A blue bracket under $P(Y | X)$ is labeled "posterior probability". A blue bracket under $P(X | Y)$ is labeled "likelihood". A blue bracket under $P(Y)$ is labeled "prior probability". A blue bracket under $P(X)$ is labeled "prior probability".

Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Tid	Refund	Marital Status	Taxable Income	Evade
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

- We need to estimate

$$P(\text{Evade} = \text{Yes} | X) \text{ and } P(\text{Evade} = \text{No} | X)$$

In the following we will replace
Evade = Yes by Yes, and
Evade = No by No

Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Tid	Refund	Marital Status	Taxable Income	Evade
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

Using Bayes Theorem:

□ $P(\text{Yes} | X) = \frac{P(X | \text{Yes})P(\text{Yes})}{P(X)}$

□ $P(\text{No} | X) = \frac{P(X | \text{No})P(\text{No})}{P(X)}$

□ How to estimate $P(X | \text{Yes})$ and $P(X | \text{No})$?

هدف اصلی محاسبه‌ی اینهاست
هر کدام مакс بشه => میشه لبیل کلاس ما

Conditional Independence

- X and Y are conditionally independent given Z if

$$P(X|YZ) = P(X|Z)$$

Equivalently

$$P(XY|Z) = P(X|Z)P(Y|Z)$$

مثال: طول بازو و مهارت خواندن
- کودک خردسال در مقایسه با بزرگسالان دارای طول
بازوی کوتاه تر و مهارت های خواندن محدودتر است
- اگر سن ثابت باشد، هیچ رابطه آشکاری بین طول بازو و
مهارت خواندن وجود ندارد
- طول بازو و مهارت های خواندن به صورت شرطی
مستقل از سن هستند

- Example: Arm length and reading skills
 - Young child has shorter arm length and limited reading skills, compared to adults
 - If age is fixed, no apparent relationship between arm length and reading skills
 - Arm length and reading skills are conditionally independent given age

Naïve Bayes Classifier

در این روش فرض میکنیم که X_i ها از هم مستقل هستند یعنی x_1, x_2, x_3, \dots به شرط y از هم مستقل میشوند.
 وقتی دو تا متغیر تصادفی نسبت به یک متغیر دیگر مستقل شوند = $\text{conditionally independent}$ که باشند این اتفاق میفته:

- Assume independence among attributes X_i when class is given:

$$P(X_1, X_2, \dots, X_d | Y_j) = P(X_1 | Y_j) P(X_2 | Y_j) \dots P(X_d | Y_j)$$

- Now we can estimate $P(X_i | Y_j)$ for all X_i and Y_j combinations from the training data

- New point is classified to Y_j if $P(Y_j) \prod P(X_i | Y_j)$ is maximal.

ضرب

الآن محاسباتمون خیلی ساده شد چون اینارو میشه جداجدا حساب کرد

Naïve Bayes on Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120K)$$

Tid	Refund	Marital Status	Taxable Income	Evade
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

$$P(X | \text{Yes}) =$$

$$P(\text{Refund} = \text{No} | \text{Yes}) \times$$

$$P(\text{Divorced} | \text{Yes}) \times$$

$$P(\text{Income} = 120K | \text{Yes})$$

این سه تا ویژگی مستقل از هم هستند پس میشه احتمال های شرطی شون را در هم ضرب کرد

$$P(X | \text{No}) =$$

$$P(\text{Refund} = \text{No} | \text{No}) \times$$

$$P(\text{Divorced} | \text{No}) \times$$

$$P(\text{Income} = 120K | \text{No})$$

این داده که پیوسته است چکارش کنیم؟
continues

Estimate Probabilities from Data

Tid	Refund	Marital Status	Taxable Income	Evade
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

- $P(y)$ = fraction of instances of class y

— e.g., $P(\text{No}) = 7/10$,
 $P(\text{Yes}) = 3/10$

اتribut های دسته
 ی مثل وضعیت تاہل
 که ۳ تا حالت داره:
 single,married
 divorced

- For categorical attributes:

$$P(X_i = c | y) = n_c / n$$

- where $|X_i = c|$ is number of instances having attribute value $X_i = c$ and belonging to class y

- Examples:

$$P(\text{Status}=\text{Married}|\text{No}) = 4/7$$

$$P(\text{Refund}=\text{Yes}|\text{Yes})=0$$

Estimate Probabilities from Data

- For continuous attributes:

- Discretization:** Partition the range into bins:

- Replace continuous value with bin value

- Attribute changed from continuous to ordinal

عملیات های شکستن و تقسیم بندی و بین کردن به بحث های پیش پردازش بر میگردد مثلاً یه سری چارچوب و قوانین هست که چطوری اینا را بشکنیم بهتر بشه مدلمن

میتوانیم فرض کنیم که ستون در ادمون از یک توزیعی پیروی میکنه چون توزیع بمون احتمال میده پس میشه از میانگین و واریانس و انحراف معیار که به صورت احتمالی هم کار میکنه استفاده کرد به جای پارتبیشن کردن مقادیر پیوسته مثل در امد

میتوانیم بازه‌ی پیوسته‌ای که داریم را به یه تعداد مشخصی از بازه‌های فیکس بشکنیم و تقسیم کنیم بعد دیگه به راحتی میشه شمرد: مثلاً بگیم اگه سطح در امد از مقدار ۱۰۰ تا ۱۲۰ هزار بود تو یه دسته قرار بگیره

Probability density estimation:

- Assume attribute follows a normal distribution
- Use data to estimate parameters of distribution (e.g., mean and standard deviation)
- Once probability distribution is known, use it to estimate the conditional probability $P(X_i|Y)$

اینطوری شمارش
داده‌ها برای احتمال
پیدا کردن خیلی
راحتer میشه

- تخمین چگالی احتمال:
فرض کنید اتریبیوت از توزیع نرمال پیروی می‌کند.
استفاده از داده‌ها برای تخمین پارامترهای توزیع (به عنوان مثال، میانگین و انحراف استاندارد).
هنگامی که توزیع احتمال مشخص شد، از آن برای تخمین احتمال شرطی $P(X_i|Y)$ استفاده کنید.

برای صفات پیوسته:
- گسته سازی: محدوده را به bin‌ها تقسیم کنید: مقدار پیوسته را با مقدار bin جایگزین کنید
- تغییر اتریبیوت از پیوسته به ترتیبی

Estimate Probabilities from Data

Tid	Refund	Marital Status	Taxable Income	Evade
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

- Normal distribution:

$$P(X_i | Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- One for each (X_i, Y_i) pair

- For (Income, Class=No):

- If Class=No

- sample mean = 110
- sample variance = 2975

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

$$2975^{1/2} = 54.54$$

احتمال اینکه برچسب
مونه‌ی من با درآمد ۱۲۰،
کلاسش no باشد عدد
۰.۰۰۷۲ است

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$$

$$P(X | \text{No}) = P(\text{Refund} = \text{No} | \text{No})$$

$$\times P(\text{Divorced} | \text{No})$$

$$\times P(\text{Income} = 120\text{K} | \text{No})$$

$$= 4/7 \times 1/7 \times 0.0072 = 0.0006$$

$$\bullet \quad P(X | \text{Yes}) = P(\text{Refund} = \text{No} | \text{Yes})$$

$$\times P(\text{Divorced} | \text{Yes})$$

$$\times P(\text{Income} = 120\text{K} | \text{Yes})$$

$$= 1 \times 1/3 \times 1.2 \times 10^{-9} = 4 \times 10^{-10}$$

For Taxable Income:

If class = No: sample mean = 110
sample variance = 2975

If class = Yes: sample mean = 90
sample variance = 25

Since $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$

Therefore $P(\text{No}|X) > P(\text{Yes}|X)$

\Rightarrow Class = No

p(no)
را از کجا اورد؟

$P(\text{Yes}) = 3/10$
 $P(\text{No}) = 7/10$

Naïve Bayes Classifier can make decisions with **partial information about attributes** in the test record

Even in absence of information about any attributes, we can use Apriori Probabilities of Class Variable:

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$$

For Taxable Income:

If class = No: sample mean = 110
sample variance = 2975

If class = Yes: sample mean = 90
sample variance = 25

$$P(\text{Yes}) = 3/10$$

$$P(\text{No}) = 7/10$$

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120K)$$

If we only know that **marital status is Divorced**, then:

$$P(\text{Yes} | \text{Divorced}) = 1/3 \times 3/10 / P(\text{Divorced})$$

$$P(\text{No} | \text{Divorced}) = 1/7 \times 7/10 / P(\text{Divorced})$$

If we also know that **Refund = No**, then

$$P(\text{Yes} | \text{Refund} = \text{No}, \text{Divorced}) = 1 \times 1/3 \times 3/10 / P(\text{Divorced}, \text{Refund} = \text{No})$$

$$P(\text{No} | \text{Refund} = \text{No}, \text{Divorced}) = 4/7 \times 1/7 \times 7/10 / P(\text{Divorced}, \text{Refund} = \text{No})$$

If we also know that **Taxable Income = 120**, then

$$P(\text{Yes} | \text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120) = 1.2 \times 10^{-9} \times 1 \times 1/3 \times 3/10 / P(\text{Divorced}, \text{Refund} = \text{No}, \text{Income} = 120)$$

$$P(\text{No} | \text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120) = 0.0072 \times 4/7 \times 1/7 \times 7/10 / P(\text{Divorced}, \text{Refund} = \text{No}, \text{Income} = 120)$$

مشکلات این کلسیفایر؟

اگه داده هامون ناقص باشند چی میشه؟ مثلا یه رکوردی داشته باشیم که سطح درآمدش رو ندانیم ولی بقیه ای اتریبیوت هارو بدانیم مثلا بدانیم که متاهل است و ... چه مقادیری دارند missing features داشته باشیم آیا کلسیفایر با روش بیزین جواب درست میده بمون؟ چون اون اتریبیوت را نداریم احتمالش هم حساب نمیشه پس برآمون مهم نیست و براساس اونایی که هستند تصمیم میگیریم => مدل بیزین برای مقادیر گم شده هم درست کار میکنه

Issues with Naïve Bayes Classifier

Given a Test Record:

$X = (\text{Married})$

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$$

$$P(\text{Yes}) = 3/10$$

$$P(\text{No}) = 7/10$$

$$P(\text{Yes} | \text{Married}) = 0 \times 3/10 / P(\text{Married})$$

$$P(\text{No} | \text{Married}) = 4/7 \times 7/10 / P(\text{Married})$$

$$= p(\text{married} | \text{no}) * p(\text{no}) / p(\text{married})$$

For Taxable Income:

If class = No: sample mean = 110

sample variance = 2975

If class = Yes: sample mean = 90

sample variance = 25

چالش دیگه:
اگه احتمال شرطی صفر بشه کل جواب صفر
میشه
یعنی نمیتوانه تصمیم گیری کنه اگه احتمال صفر
بشه
مثل اینه که یه شرایطی پیش بیاد و ما براش!
رکورد نداشته باشیم یا تعداد داده ها کم باشه!
تل پاک کردن رکوردها در مثال بعدی که باعث
صرفشدن احتمال میشه

Issues with Naïve Bayes Classifier

$$P(X_1, X_2, \dots, X_d | Y_j) = P(X_1 | Y_j) P(X_2 | Y_j) \dots P(X_d | Y_j)$$

If one of the conditional probabilities is zero, then the entire expression becomes zero

اگر یکی از احتمالات شرطی صفر باشد، کل عبارت صفر می شود

Issues with Naïve Bayes Classifier

Consider the table with **Tid = 7 deleted**

Tid	Refund	Marital Status	Taxable Income	Evade
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
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 2/6$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/6$$

→ $P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/6$$

→ $P(\text{Marital Status} = \text{Divorced} | \text{No}) = 0$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/6$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$$

For Taxable Income:

If **class = No**: sample mean = 91
sample variance = 685

If **class = Yes**: sample mean = 90
sample variance = 25

Given **X** = (Refund = Yes, Divorced, 120K)

$$P(X | \text{No}) = 2/6 \times 0 \times 0.0083 = 0$$

$$P(X | \text{Yes}) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0$$

$p(\text{Refund} = \text{yes} | \text{Yes}) = 0$

$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 0$

Naïve Bayes will not be able to classify X as Yes or No!

Issues with Naïve Bayes Classifier

راه حل مشکل صفرشدن احتمال ها:
نقریب زدن احتمال ها مثلا صورت کسر
را بایک عددکوچکی جمع کنیم تا هیچ
وقت صفر تولید نشه
تخمین احتمالاتی (: probability estimation)

- If one of the conditional probabilities is zero, then the entire expression becomes zero
- Need to use other estimates of conditional probabilities than simple fractions
- Probability estimation:

original $P(X_i = c|y) = \frac{n_c}{n}$

Laplace Estimate: $P(X_i = c|y) = \frac{n_c + 1}{n + v}$

m – estimate: $P(X_i = c|y) = \frac{n_c + mp}{n + m}$

n : number of training instances belonging to class y

n_c : number of instances with $X_i = c$ and $Y = y$

v : total number of attribute values that X_i can take

p : initial estimate of $(P(X_i = c|y))$ known apriori

m : hyper-parameter for our confidence in p

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Redundant and correlated attributes will violate class conditional assumption
 - Use other techniques such as Bayesian Belief Networks (BBN)

عيوب اين روش

۱. نسبت به نقاط نویز مقاوم است

۲. اگه میسینگ ولیوز داشته باشیم بازم این روش بمون يه جوابی میده

۳. اگه متغیر هامون مستقل از هم باشند کار میکنه و جواب میده

۴. عیوبی که داره برای متغیرها و اtribut های وابسته است
که کریشن دارند میتواند احتمال شرطی را خراب کنه
ما در این مدل فرض کردیم متغیرها که همان اtribut هامون هستند از هم مستقل اند

مقاوم به نقاط نویز جدا شده

مقادیر گمشده را با نادیده گرفتن نمونه در طول محاسبات برآورد احتمال مدیریت کنید

مقاوم به ویژگی های نامرتب

صفات زائد و مرتبه، فرض شرطی کلاس را نقض خواهد کرد
- استفاده از تکنیک های دیگر مانند شبکه های باور بیزی (BBN)

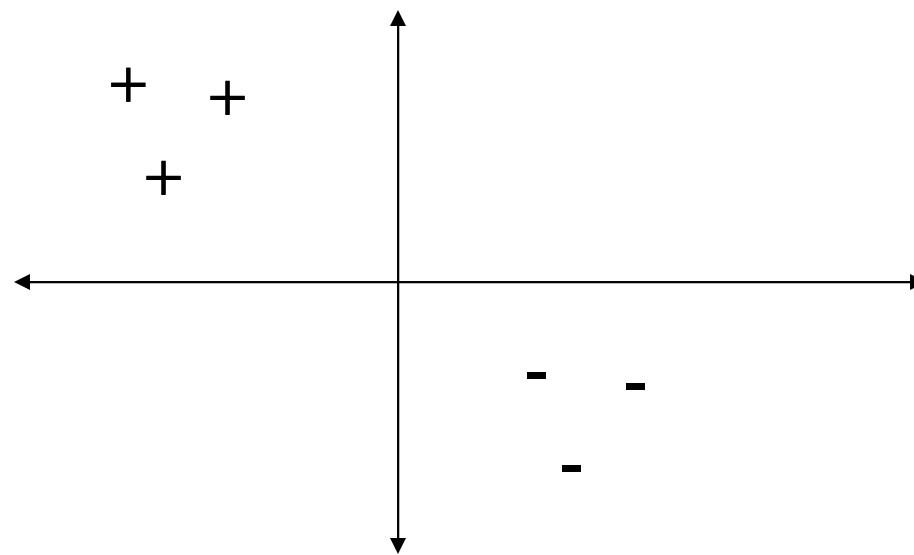
ARTIFICIAL NEURAL NETWORKS

Basic Idea_

چطوری با کمک ریاضی میشه داده ها را از هم

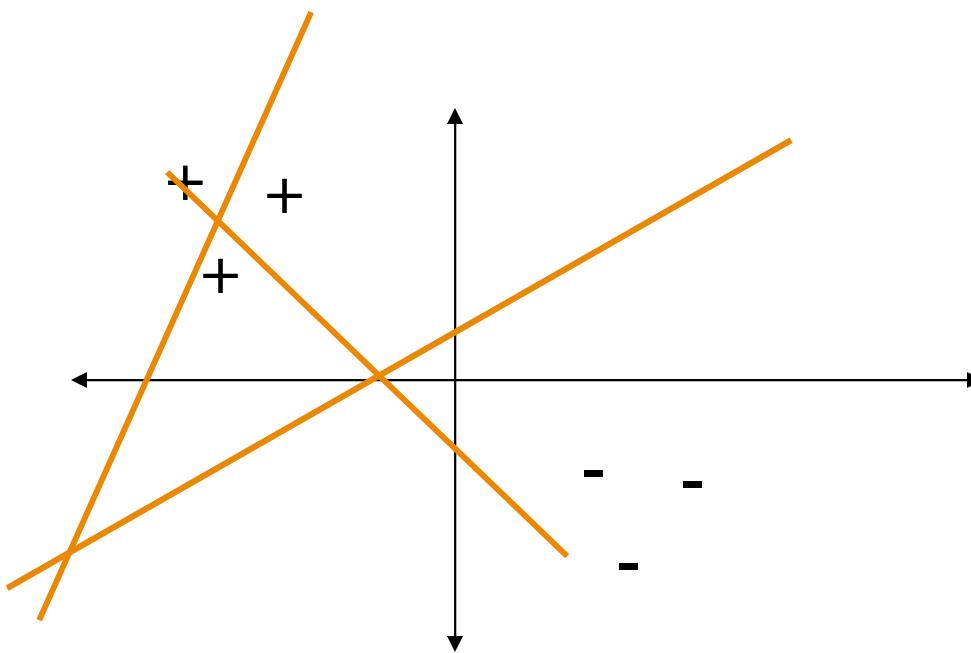
جدا کرد؟

باکشیدن خط؟



Decision boundary ($WX = 0$)

Basic Idea_

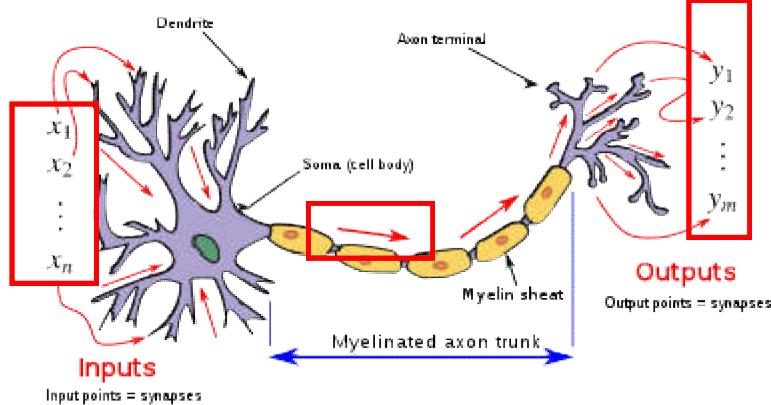


Decision boundary ($WX = 0$)

Artificial Neural Networks (ANN)

- Basic Idea: A complex non-linear function can be learned as a composition of simple processing units

ایده اصلی: یک تابع غیر خطی پیچیده را می توان به عنوان ترکیبی از واحدهای پردازش ساده یاد گرفت.

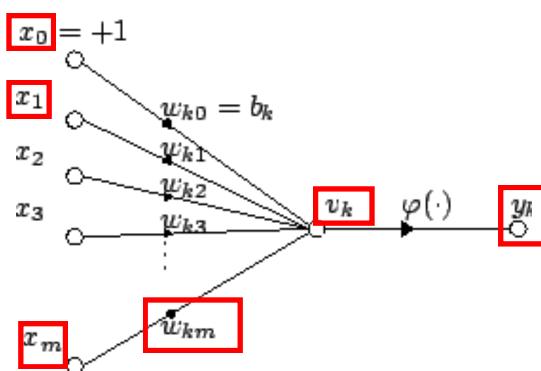


<https://en.wikipedia.org/>

ترکیب وزن دار X ها داره یه تابع میسازه

کوچکترین واحد پردازشی که معادل یک عصب است را میشه با یک معادله ریاضی نشان داد

$$\mathbf{Y} \sim \mathbf{WX}$$



Artificial Neural Networks (ANN)

- **Basic Idea:** A complex non-linear function can be learned as a composition of simple processing units
- ANN is a collection of **simple processing units** (**nodes**) that are connected by directed links (**edges**)
 - Every node receives signals from **incoming edges**, performs computations, and **transmits signals** to **outgoing edges**
 - Analogous to **human brain** where nodes are neurons and signals are electrical impulses
 - Weight of an edge determines the **strength of connection** between the nodes

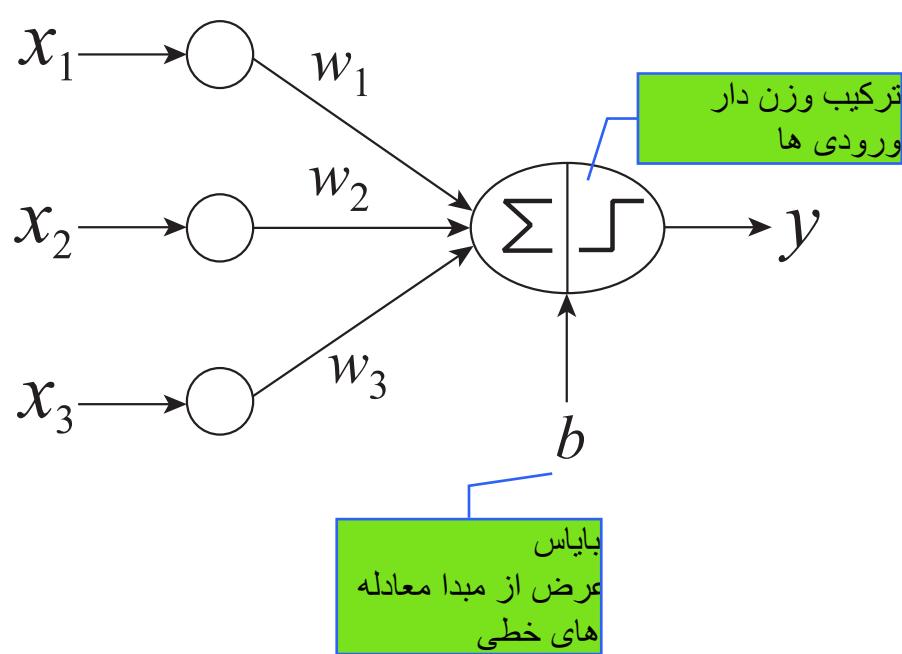
امکان ساخت توابع خطی

- **Simplest ANN: Perceptron** (**single neuron**)

- مشابه مغز انسان که در آن گره ها نورون هستند و سیگنال ها تکانه های الکتریکی هستند
- وزن یک لبه قدرت اتصال بین گره ها را تعیین می کند

ANN مجموعه ای از واحدهای پردازش ساده است (گره ها) که توسط پیوندهای هدایت شده (لبه ها) به هم متصل می شوند - هر گره سیگنال هایی را از لبه های ورودی دریافت می کند، محاسبات را انجام می دهد و سیگنال ها را به لبه های خروجی ارسال می کند

Basic Architecture of Perceptron



$$y = \begin{cases} 1, & \text{if } \mathbf{w}^T \mathbf{x} + b > 0. \\ -1, & \text{otherwise.} \end{cases}$$

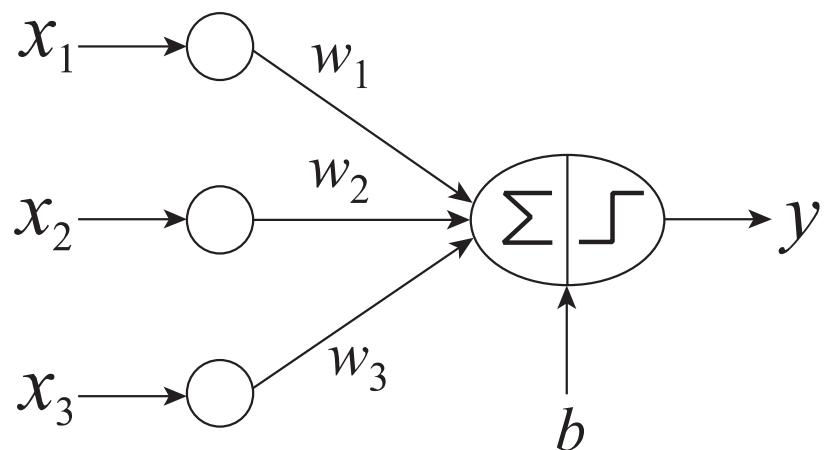
$$\tilde{\mathbf{w}} = (\mathbf{w}^T \ b)^T \quad \tilde{\mathbf{x}} = (\mathbf{x}^T \ 1)^T$$

$$\hat{y} = \text{sign}(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})$$

↑
Activation Function

- Learns linear decision boundaries
- Related to logistic regression (activation function is sign instead of sigmoid)

Basic Architecture of Perceptron



$$y = \begin{cases} 1, & \text{if } \mathbf{w}^T \mathbf{x} + b > 0. \\ -1, & \text{otherwise.} \end{cases}$$

$$\tilde{\mathbf{w}} = (\mathbf{w}^T \ b)^T \quad \tilde{\mathbf{x}} = (\mathbf{x}^T \ 1)^T$$

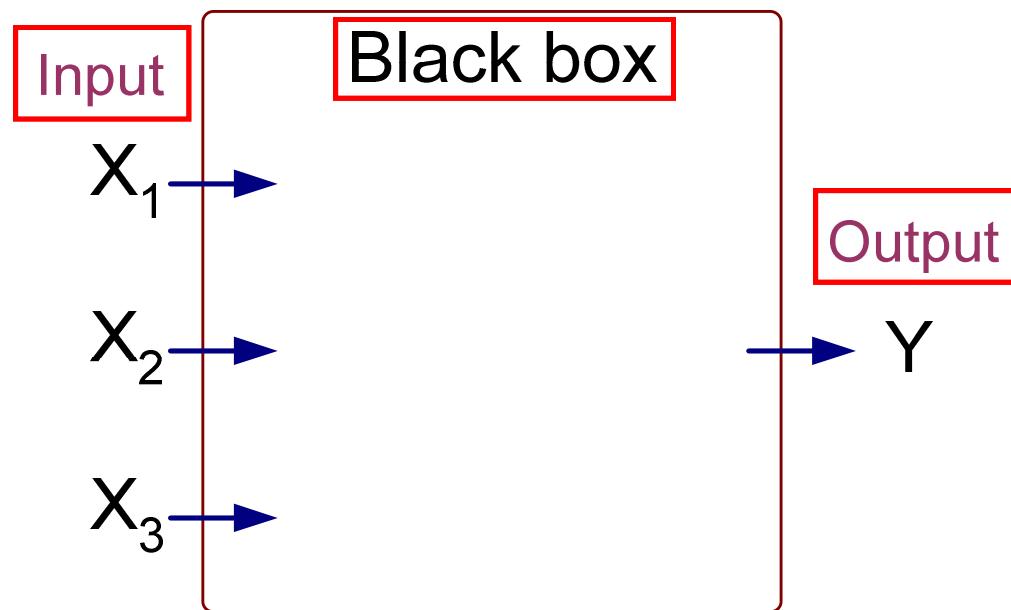
$$\hat{y} = sign(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})$$

Activation Function

- Learns linear decision boundaries
- Related to logistic regression (activation function is sign instead of sigmoid)

Perceptron Example

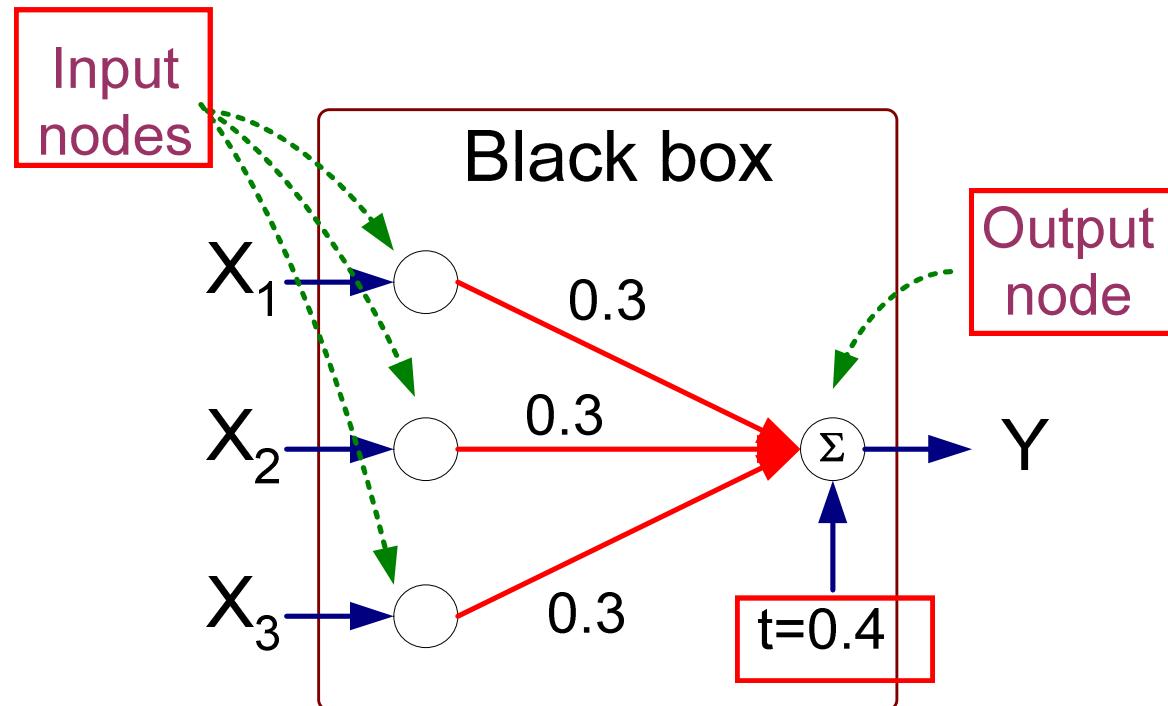
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output Y is 1 if at least two of the three inputs are equal to 1.

Perceptron Example

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



$$Y = \text{sign} (0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

where $\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$

Perceptron Learning Rule

- Initialize the weights (w_0, w_1, \dots, w_d)

- Repeat

- For each training example (x_i, y_i)

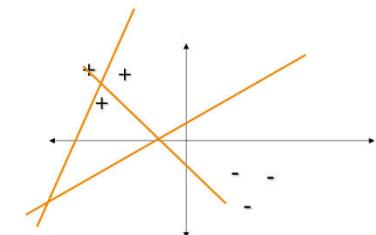
- ◆ Compute \hat{y}_i

- ◆ Update the weights:

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$

- Until stopping condition is met

- k : iteration number; λ : learning rate



به تعداد اتریبیوت هامون وزن داریم
برای X_1 یک w_1 داریم و برای X_2 یه
 w_2 داریم

Perceptron Learning Rule

- Weight update formula:

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$

- Intuition:

- Update weight based on error: $e = (y_i - \hat{y}_i)$

◆ If $y = \hat{y}$, $e=0$: no update needed

تابع خط:
مقداری که باید باشه
منهای مقداری که
پیشینی کرده

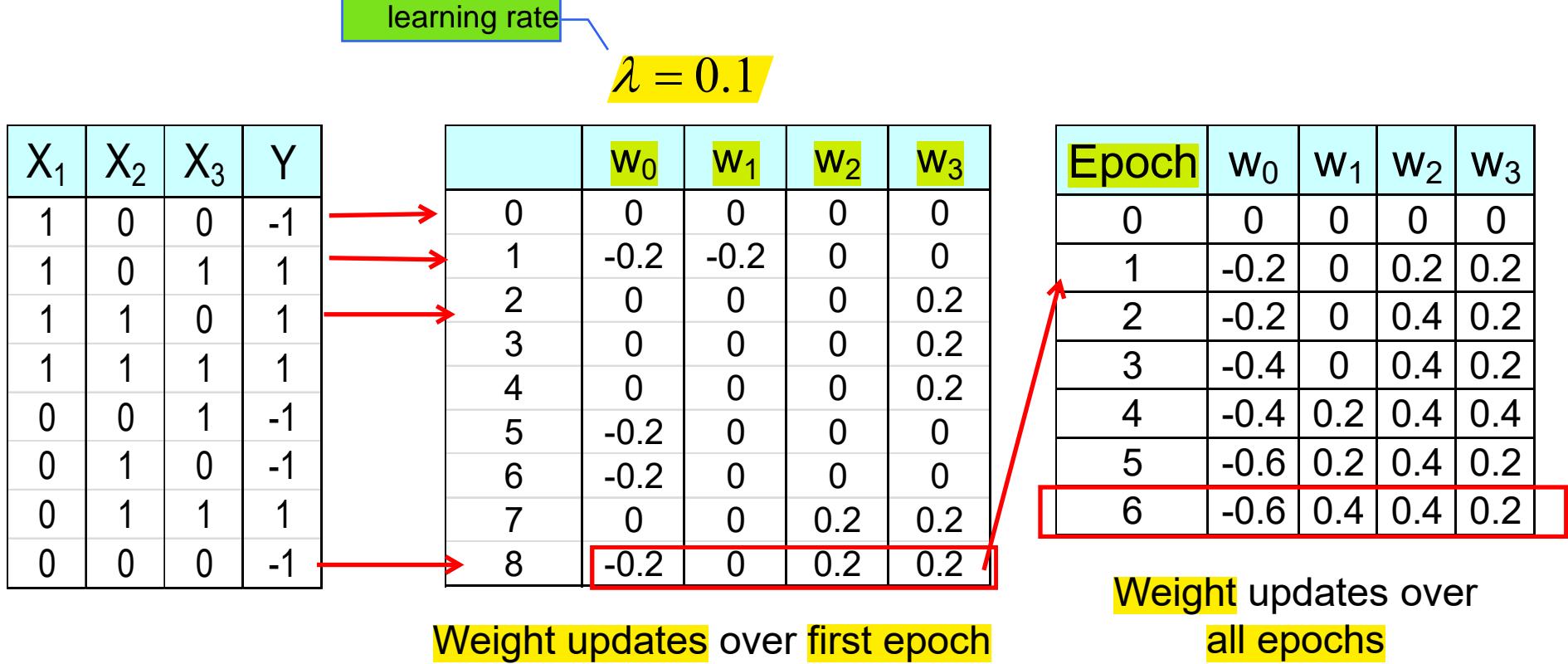
◆ If $y > \hat{y}$, $e=2$: weight must be increased (assuming x_{ij} is positive) so that \hat{y} will increase

◆ If $y < \hat{y}$, $e=-2$: weight must be decreased (assuming x_{ij} is positive) so that \hat{y} will decrease

وزن در جهتی که
خطا کمتر بشه
حرکت میکنه

چرا خطا میتوانه صفر یا -2 یا 2 باشه؟ چون خروجی y یا
یک است یا منفی یک

Example of Perceptron Learning

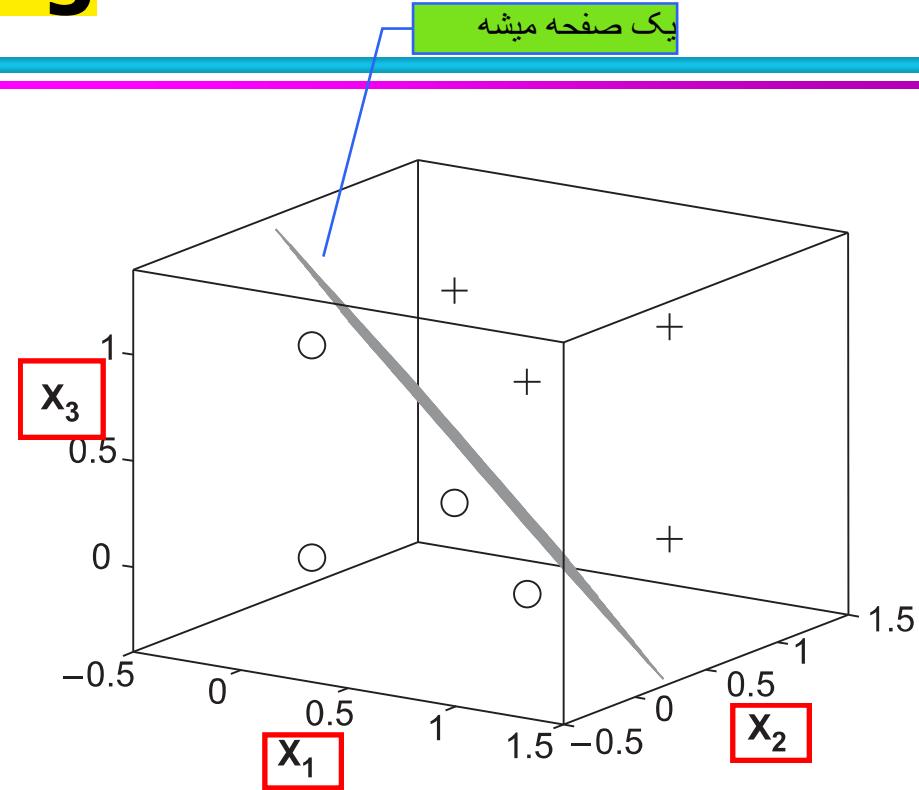


Perceptron Learning

یک صفحه میشه

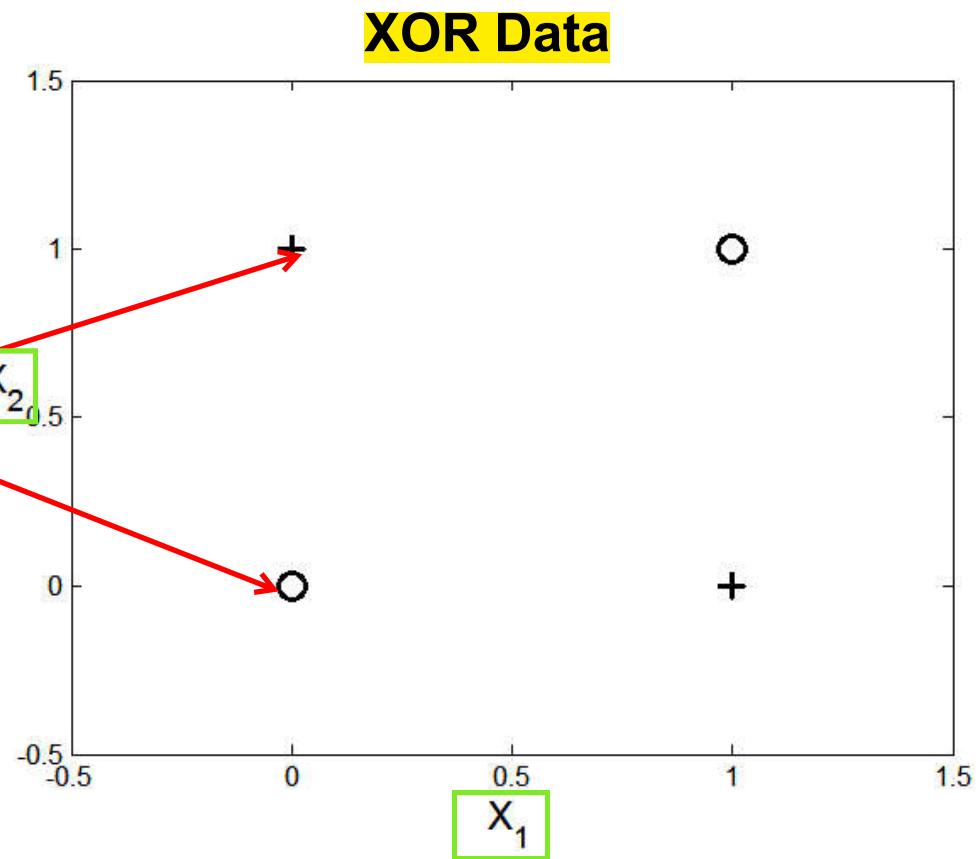
- Since y is a linear combination of input variables, decision boundary is linear

اگه داده هامون ۳تا اتریبیوت داشته باشه و در نتیجه فضامون ۳ بعدی باشه و ۲تا کلاس داشته باشیم : یه کلاس دایره و یه کلاس مثبت
مثبت در فضای ۲بعدی گفتیم یه خط داره میسازه در فضای ۳بعدی چی؟
یک صفحه میسازه



Nonlinearly Separable Data

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1



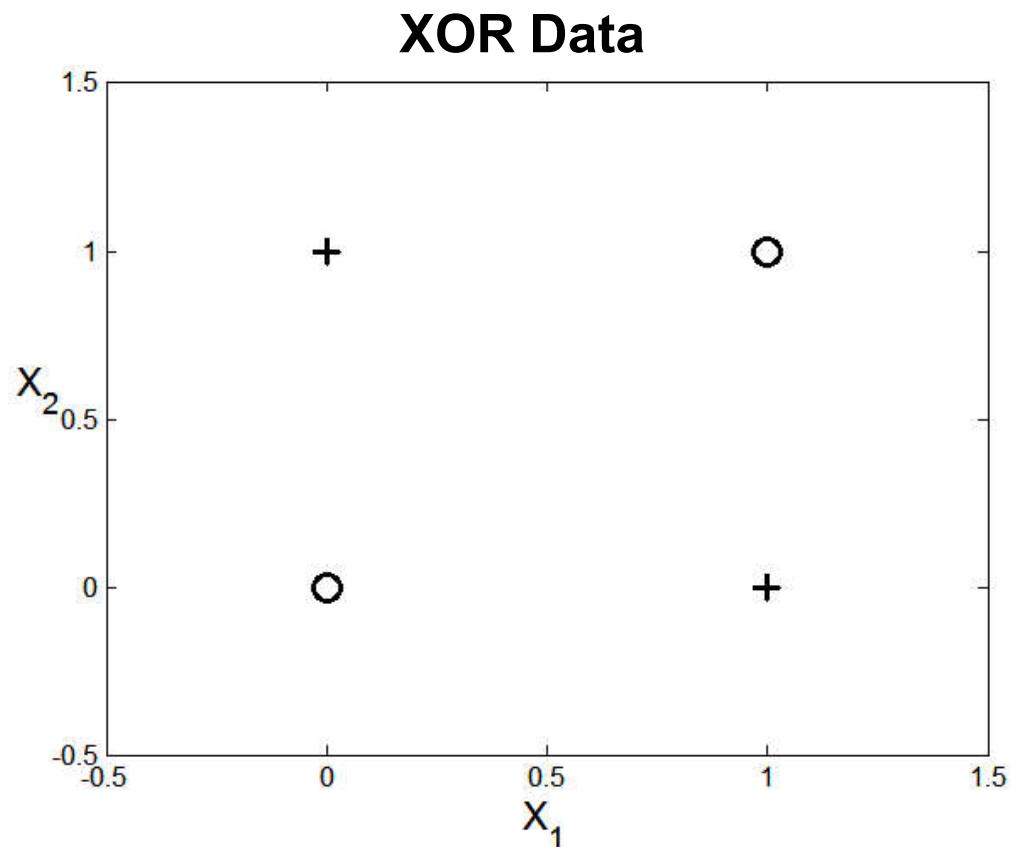
Nonlinearly Separable Data

For nonlinearly separable problems, **perceptron learning** algorithm
will fail because **no linear hyperplane** can separate the data perfectly

برای مسائل غیرخطی قابل تفکیک، الگوریتم یادگیری پرسپترون شکست خواهد خورد زیرا هیچ ابرصفحه خطی نمی تواند داده ها را به طور کامل جدا کند.

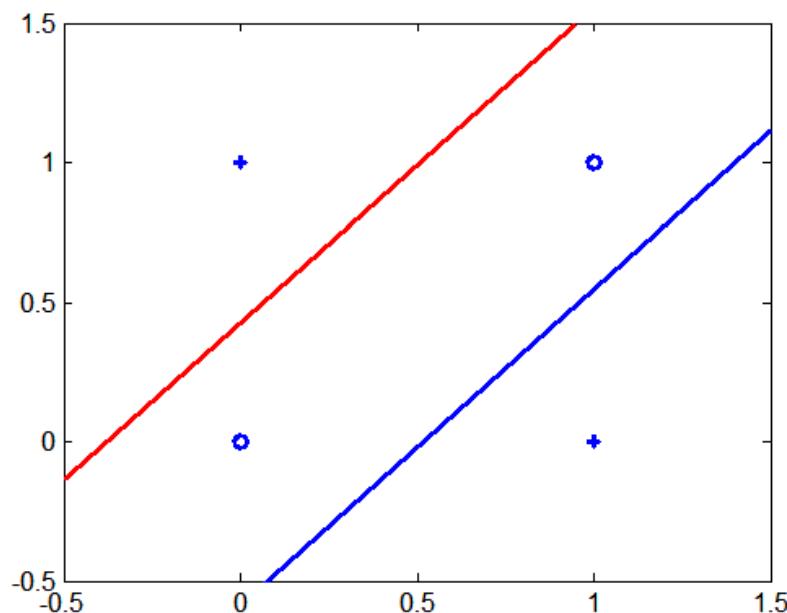
$$y = x_1 \oplus x_2$$

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1



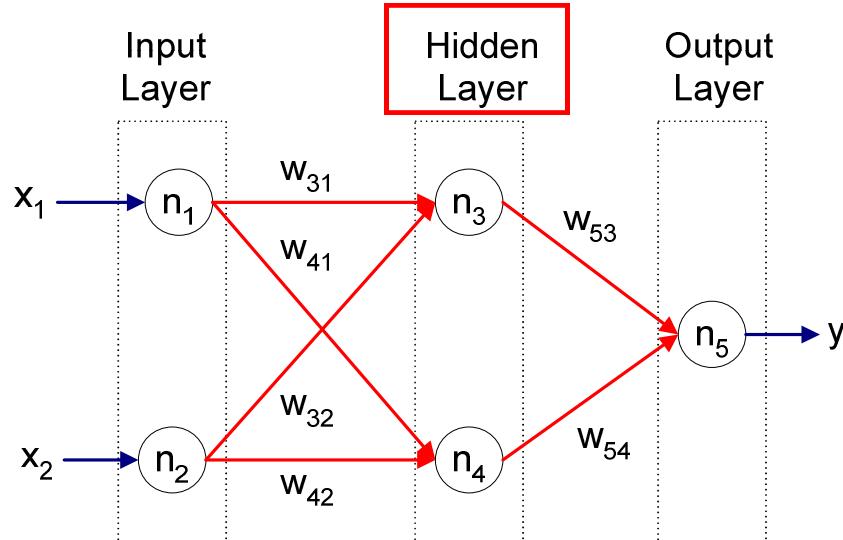
Nonlinearly Separable Data

XOR Data

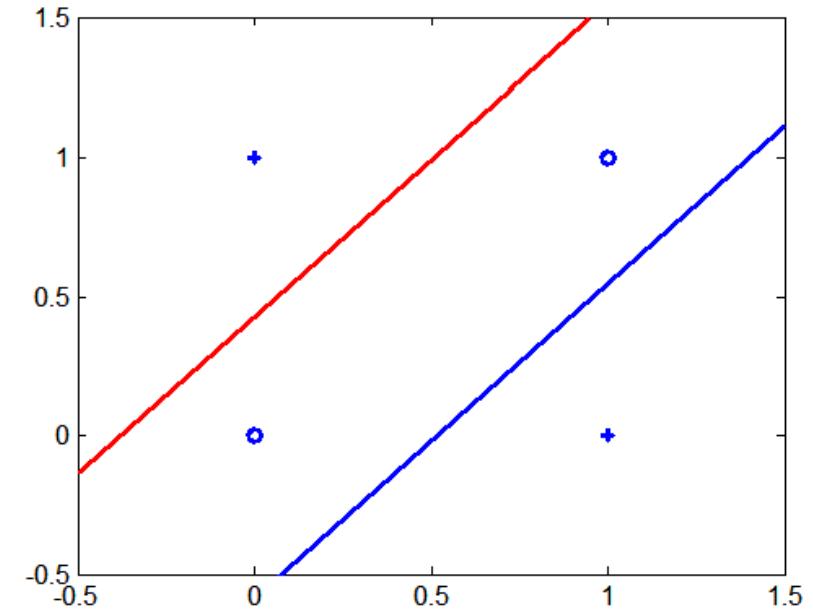


Multi-layer Neural Network

- Multi-layer neural networks with **at least one hidden layer** can solve any type of classification task involving **nonlinear decision surfaces**

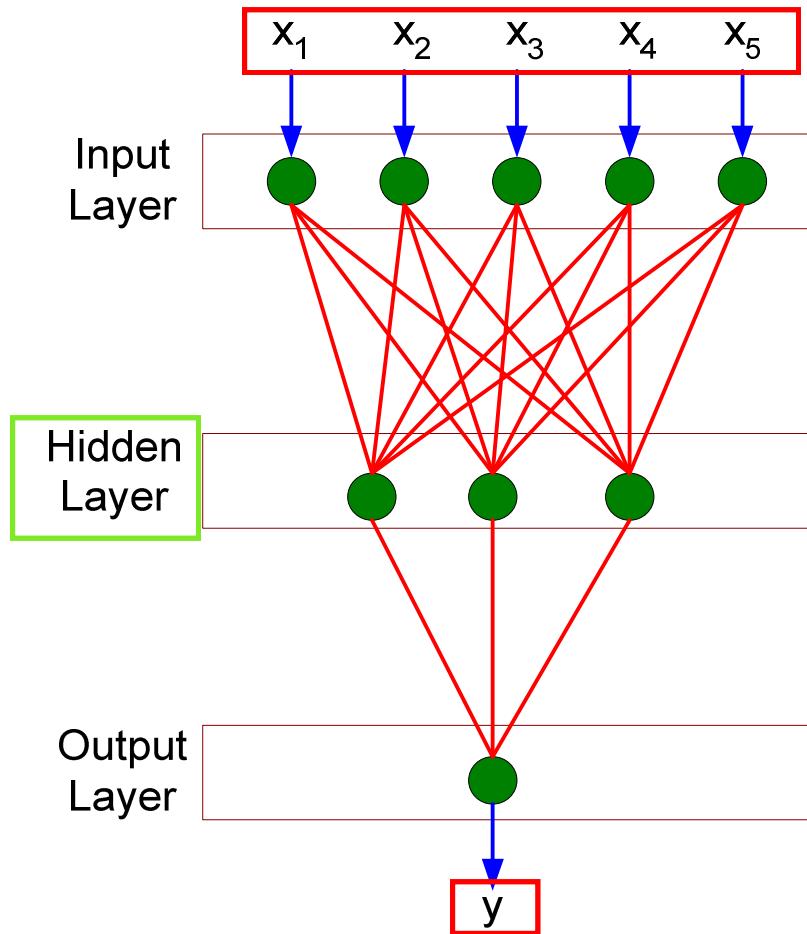


XOR Data



شبکه های عصبی چند لایه با حداقل یک لایه پنهان می توانند هر نوع کار طبقه بندی را که شامل سطوح تصمیم گیری غیرخطی باشد حل کنند.

Multi-layer Neural Network



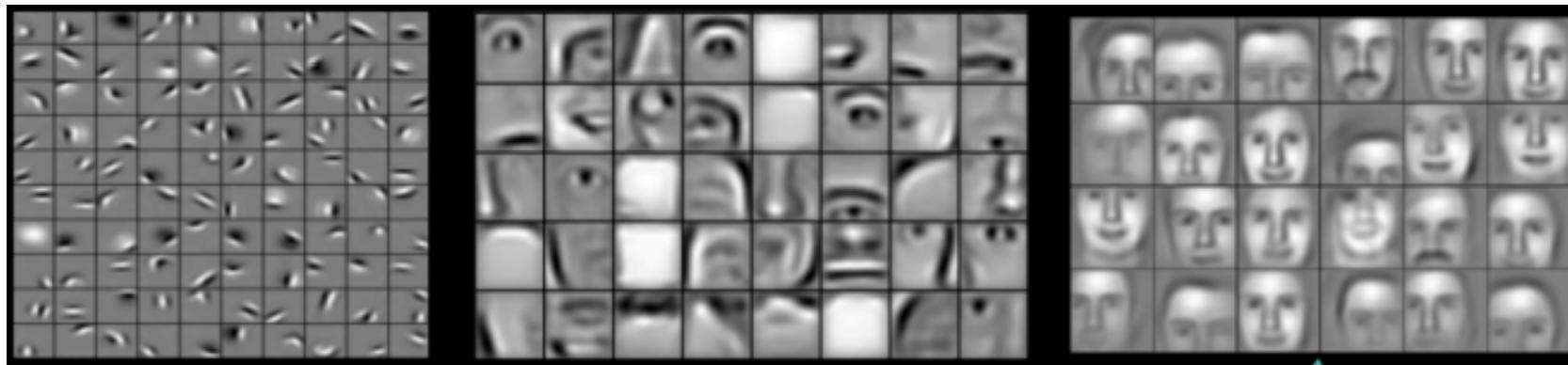
- More than one *hidden layer* of computing nodes
- Every node in a hidden layer operates on activations from preceding layer and transmits activations forward to nodes of next layer
- Also referred to as “**feedforward neural networks**”

بیش از یک لایه پنهان گره های محاسباتی هر گره در یک لایه مخفی بر روی فعال سازی از لایه قبلی عمل می کند و فعال سازی ها را به گره های لایه بعدی منتقل می کند.
همچنین به عنوان "شبکه های عصبی پیشخور" نیز شناخته می شود.

Why Multiple Hidden Layers?

- Activations at hidden layers can be viewed as features extracted as functions of inputs
- Every hidden layer represents a level of abstraction
 - Complex features are compositions of simpler features

هرچه در لایه های بالابریم ویژگی های سطح بالاتر میسازیم

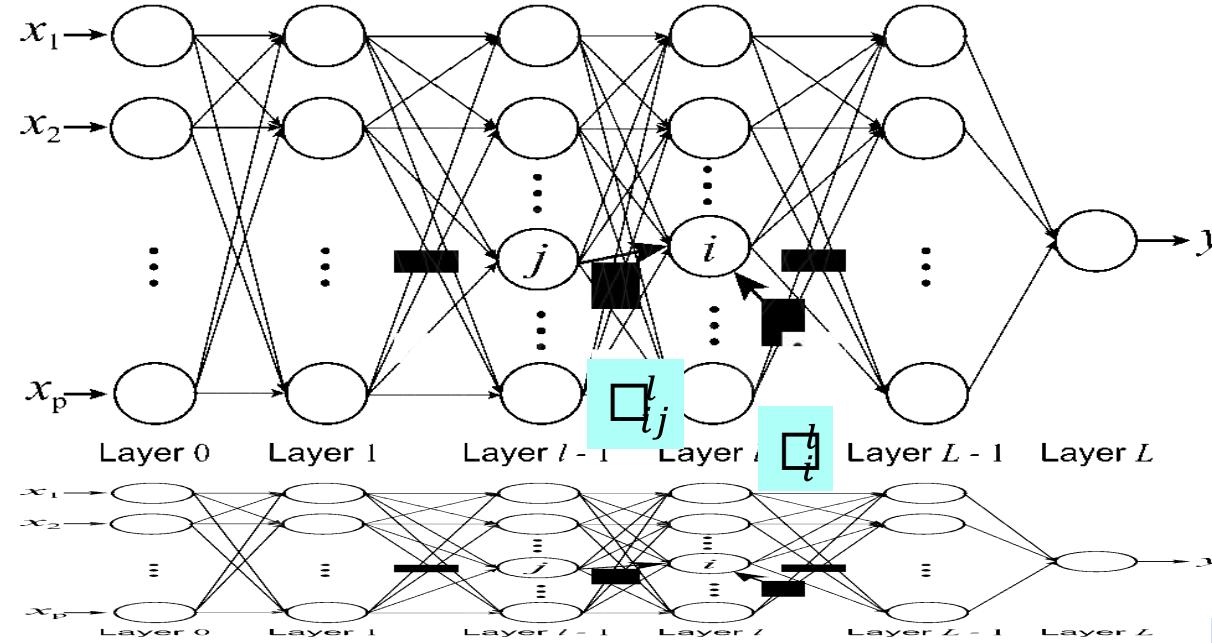


- Number of layers is known as depth of ANN
 - Deeper networks express complex hierarchy of features

فعالسازی در لایه های مخفی را میتوان به عنوان ویژگیهایی مشاهده کرد که به عنوان توابع ورودی استخراج میشوند.
هر لایه پنهان نشان دهنده سطحی از انتزاع است - ویژگی های پیچیده ترکیبی از ویژگی های ساده تر هستند

تعداد لایه ها به عنوان عمق ANN شناخته می شود - شبکه های عمیق تر سلسله مراتب پیچیده ای از ویژگی هارا بیان می کنند

Multi-Layer Network Architecture



$$a_i^l = f(z_i^l) = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\right)$$

Activation value at node i at layer l

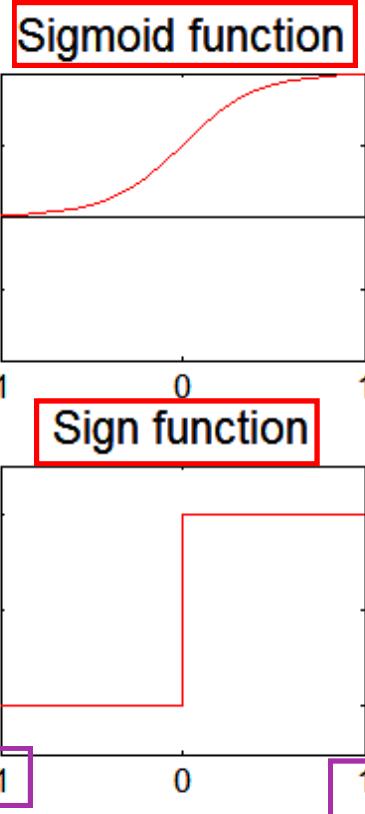
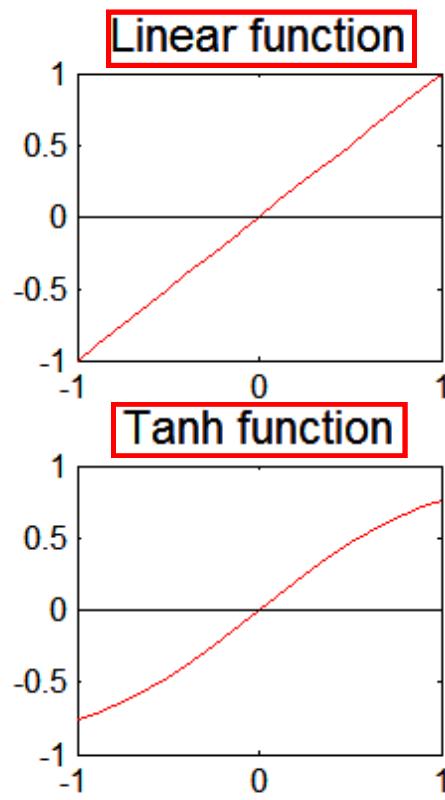
Activation Function

Linear Predictor

وزن نمونه‌ی آم در
اتribut زم در لایه
ی آم

Activation Functions

$$a_i^l = f(z_i^l) = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\right)$$



$$a_i^l = \sigma(z_i^l) = \frac{1}{1 + e^{-z_i^l}}$$

$$\frac{\partial a_i^l}{\partial z_i^l} = \frac{\partial \sigma(z_i^l)}{\partial z_i^l} = a_i^l(1 - a_i^l)$$

نگاشت مقادیر به یک بازه مشخص مثل منفی
یک تا یک

Learning Multi-layer Neural Network

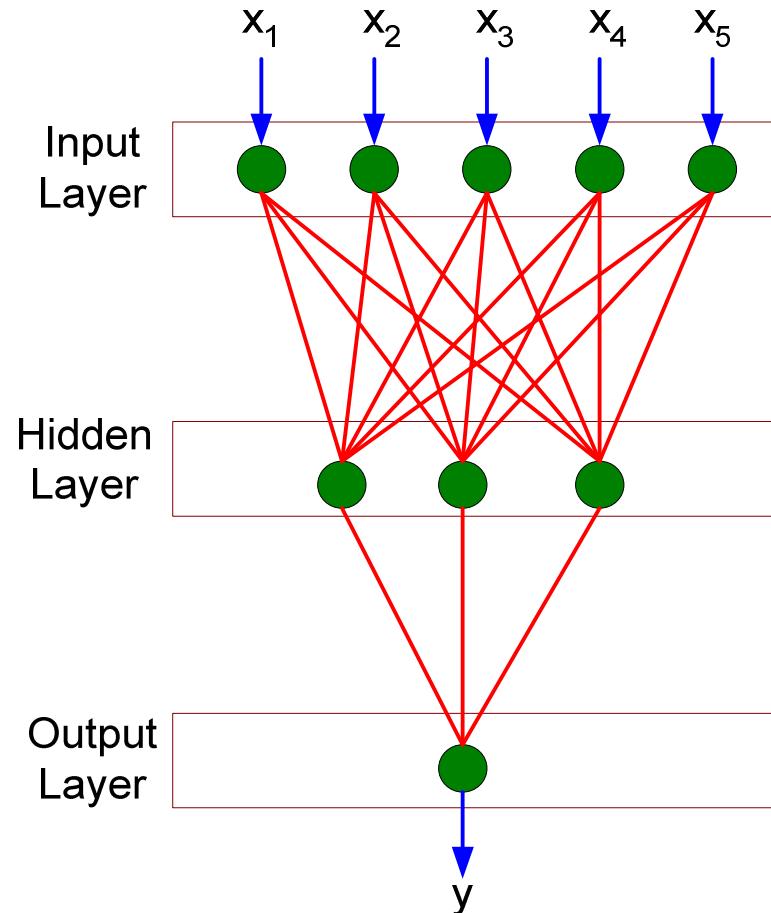
- Can we apply perceptron learning rule to each node, including hidden nodes?
 - Perceptron learning rule computes error term $e = y - \hat{y}$ and updates weights accordingly
 - ◆ Problem: how to determine the true value of y for hidden nodes?

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$

Learning Multi-layer Neural Network

- Can we apply perceptron learning rule to each node, including hidden nodes?
 - Perceptron learning rule computes error term $e = y - \hat{y}$ and updates weights accordingly
 - ◆ Problem: how to determine the true value of y for hidden nodes?
 - Approximate error in hidden nodes by error in the output nodes
 - ◆ Problem:
 - Not clear how adjustment in the hidden nodes affect overall error
 - No guarantee of convergence to optimal solution

Learning Multi-layer Neural Network



Gradient Descent

برای اندازه گیری
خطاهای در تمام نقاط
آموزشی

- Loss Function to measure errors across all training points

$$E(\mathbf{w}, \mathbf{b}) = \sum_{k=1}^n \text{Loss}(y_k, \hat{y}_k)$$

Squared Loss:

$$\text{Loss}(y_k, \hat{y}_k) = (y_k - \hat{y}_k)^2$$

- Gradient descent: Update parameters in the direction of “maximum descent” in the loss function across all points

$$w_{ij}^l \leftarrow w_{ij}^l - \lambda \frac{\partial E}{\partial w_{ij}^l},$$

مشتق تابع خطای نسبت
به وزن ها

λ : learning rate

$$b_i^l \leftarrow b_i^l - \lambda \frac{\partial E}{\partial b_i^l},$$

- Stochastic gradient descent (SGD): update the weight for every instance (minibatch SGD: update over min-batches of instances)

نزول گرادیان تصادفی (SGD): وزن را برای
هر نمونه بهروزرسانی کنید (مینی دسته)
بهروزرسانی در تعداد حداقل دستهای از نمونهها)

نزول گرادیان تصادفی (SGD): وزن را برای
هر نمونه بهروزرسانی کنید (مینی دسته)
بهروزرسانی در تعداد حداقل دستهای از نمونهها)

Design Issues in ANN

Number of nodes in input layer

- One input node per **binary/continuous** attribute
- k or $\log_2 k$ nodes for each **categorical** attribute with k values

Number of nodes in output layer

- One output for **binary class problem**
- k or $\log_2 k$ nodes for **k -class problem**

Number of hidden layers and nodes per layer

Initial weights and biases

Learning rate, max. number of epochs, mini-batch size for mini-batch SGD, ...

تعداد گره ها در لایه ورودی
- یک گره ورودی در هر ویژگی باینری/پیوسته
- گره های k یا $\log_2 k$ برای هر صفت طبقه بندی با مقادیر k

تعداد گره ها در لایه خروجی
- یک خروجی برای مسئله کلاس باینری
- $\log_2 k$ گره برای مسئله ای با k کلاس

تعداد لایه ها و گره های پنهان در هر لایه وزن ها و سوگیری های اولیه نرخ یادگیری، حداقل.
تعداد دوره ها، اندازه مینی دسته ای برای مینی دسته ای SGD، ...

- بزرگ باشد، ممکن است از برازش بیش از حد رنج ببرند

- به طور طبیعی سلسله مراتبی از ویژگی ها را در سطوح مختلف

انتزاعات نشان می دهد

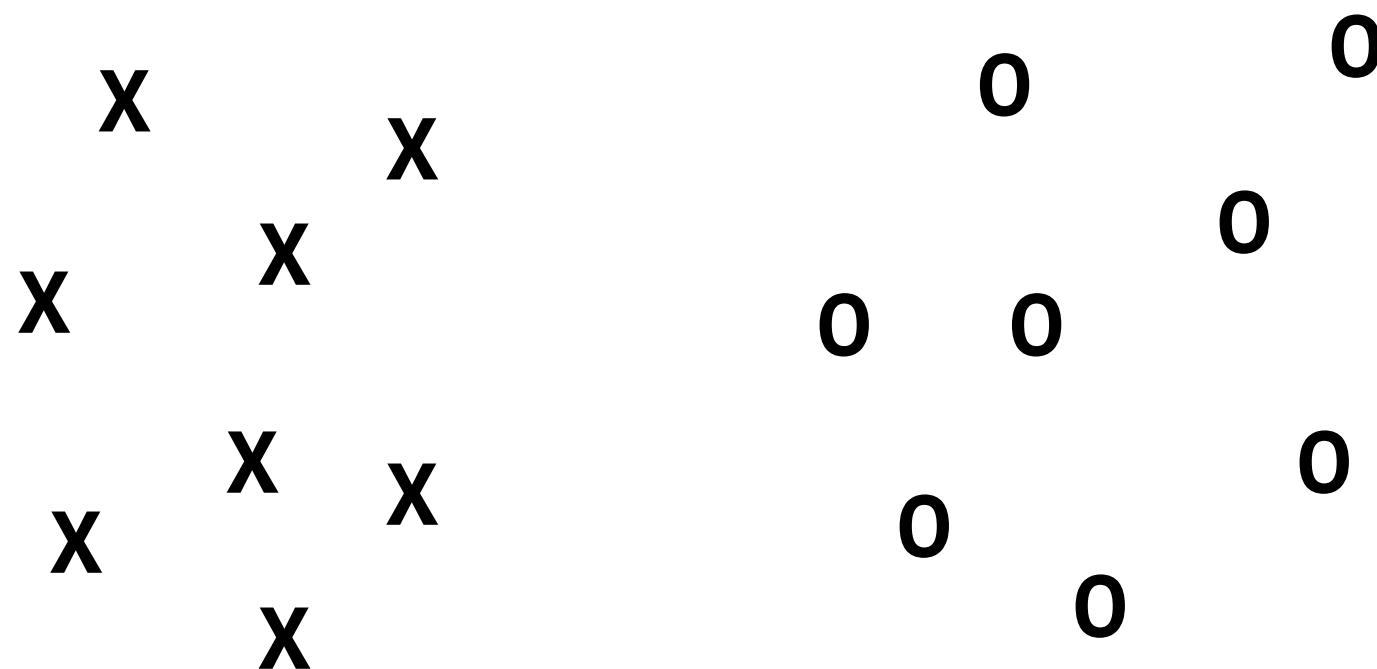
Characteristics of ANN

- Multilayer ANN are universal approximators but could suffer from overfitting if the network is too large
 - Naturally represents a hierarchy of features at multiple levels of abstractions
- Gradient descent may converge to local minimum
- Model building is compute intensive, but testing is fast
- Can handle redundant and irrelevant attributes because weights are automatically learnt for all attributes
- Sensitive to noise in training data
 - This issue can be addressed by incorporating model complexity in the loss function
- Difficult to handle missing attributes

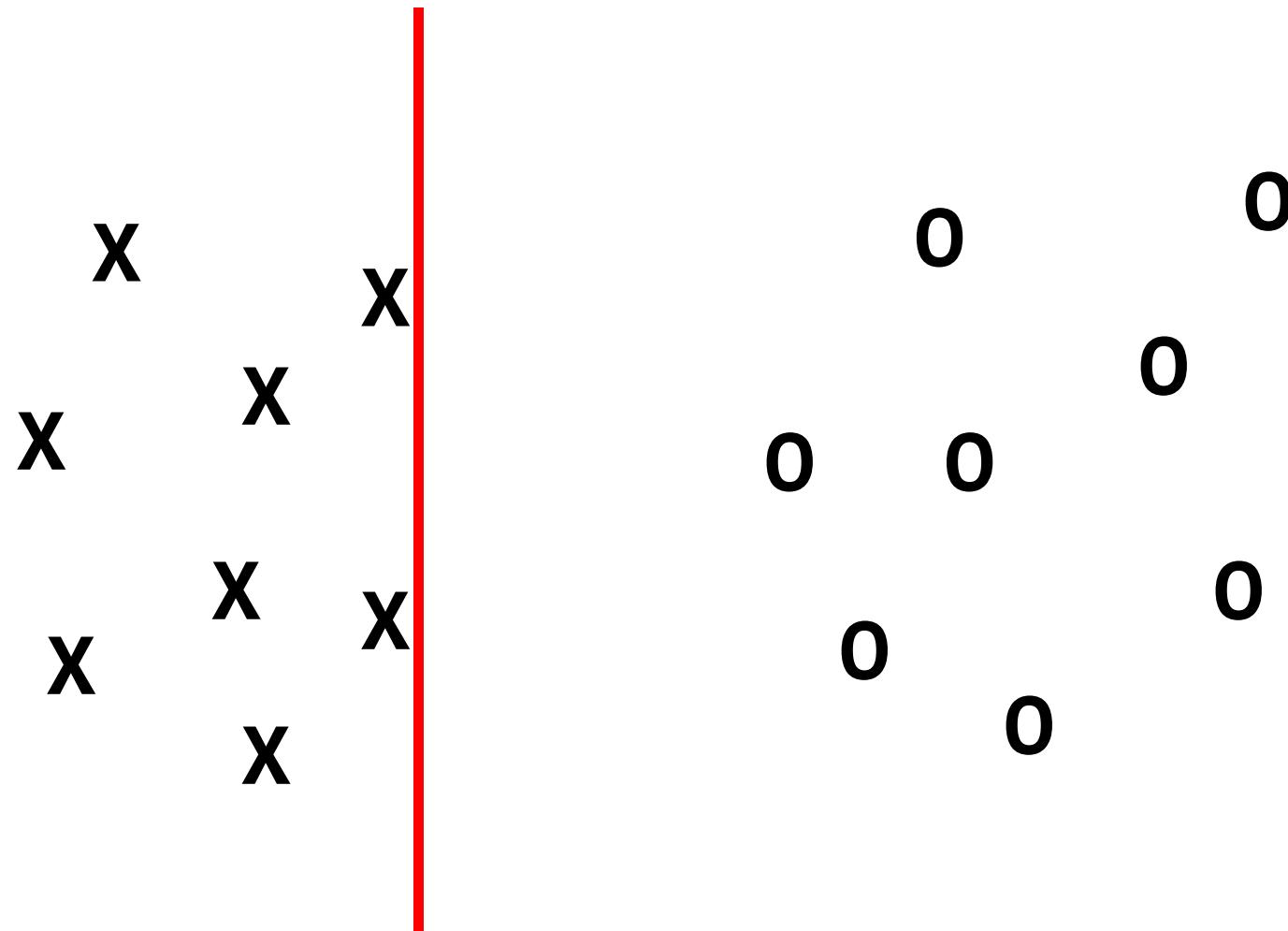
نزول گرادیان ممکن است به حداقل محلی همگرا شود
مدل سازی محاسبات فشرده است، اما آزمایش سریع است.
می تواند ویژگی های اضافی و نامربوط را مدیریت کند زیرا وزن ها به طور خودکار برای همه ویژگی ها یاد می گیرند.
حساس به نویز در داده های آموزشی
- این مسئله را می توان با گنجاندن پیچیدگی مدل درتابع ضرر برطرف کرد.
رسیدگی به ویژگی های از دست رفته دشوار است

SUPPORT VECTOR MACHINES

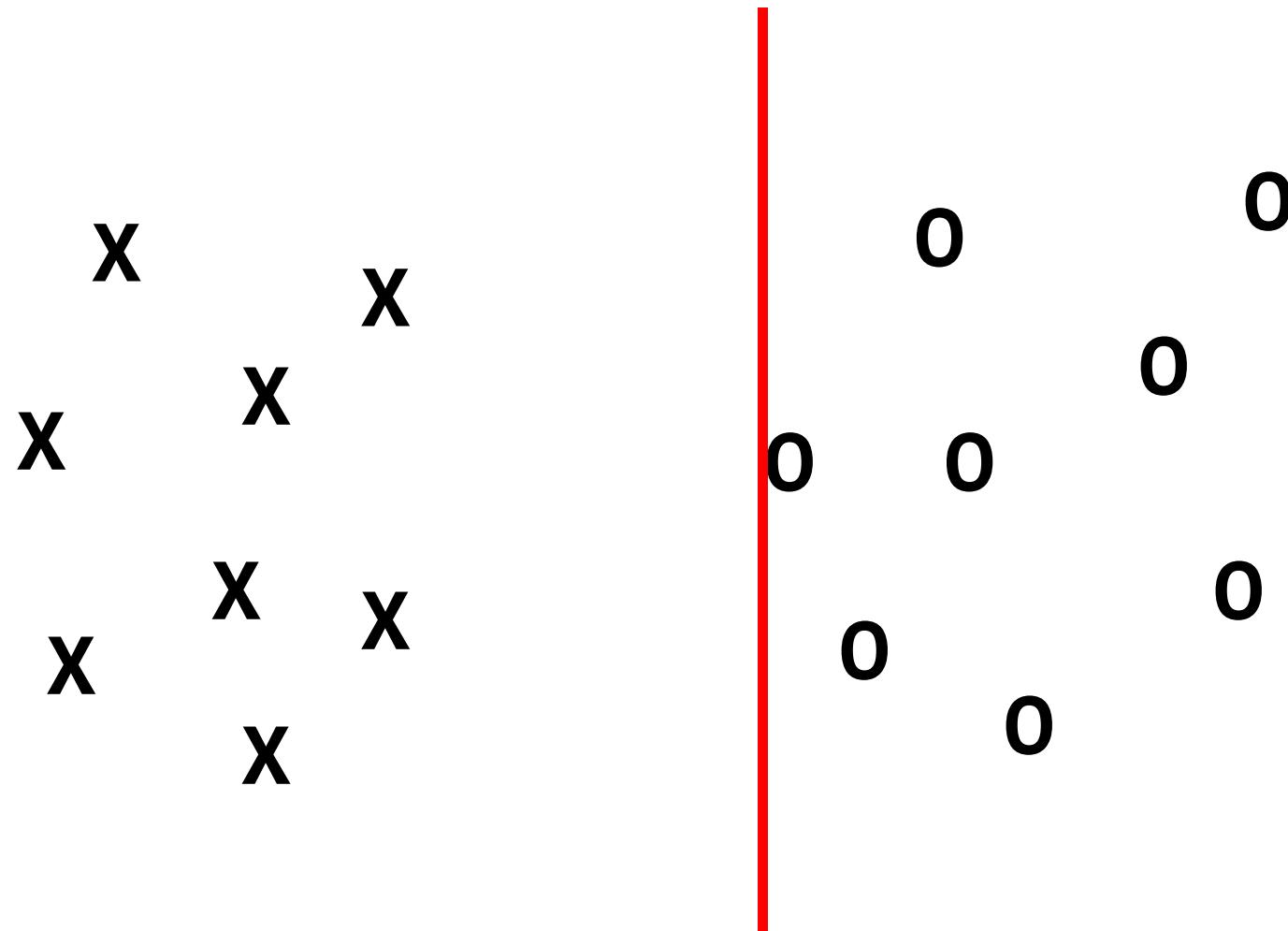
Intuitions



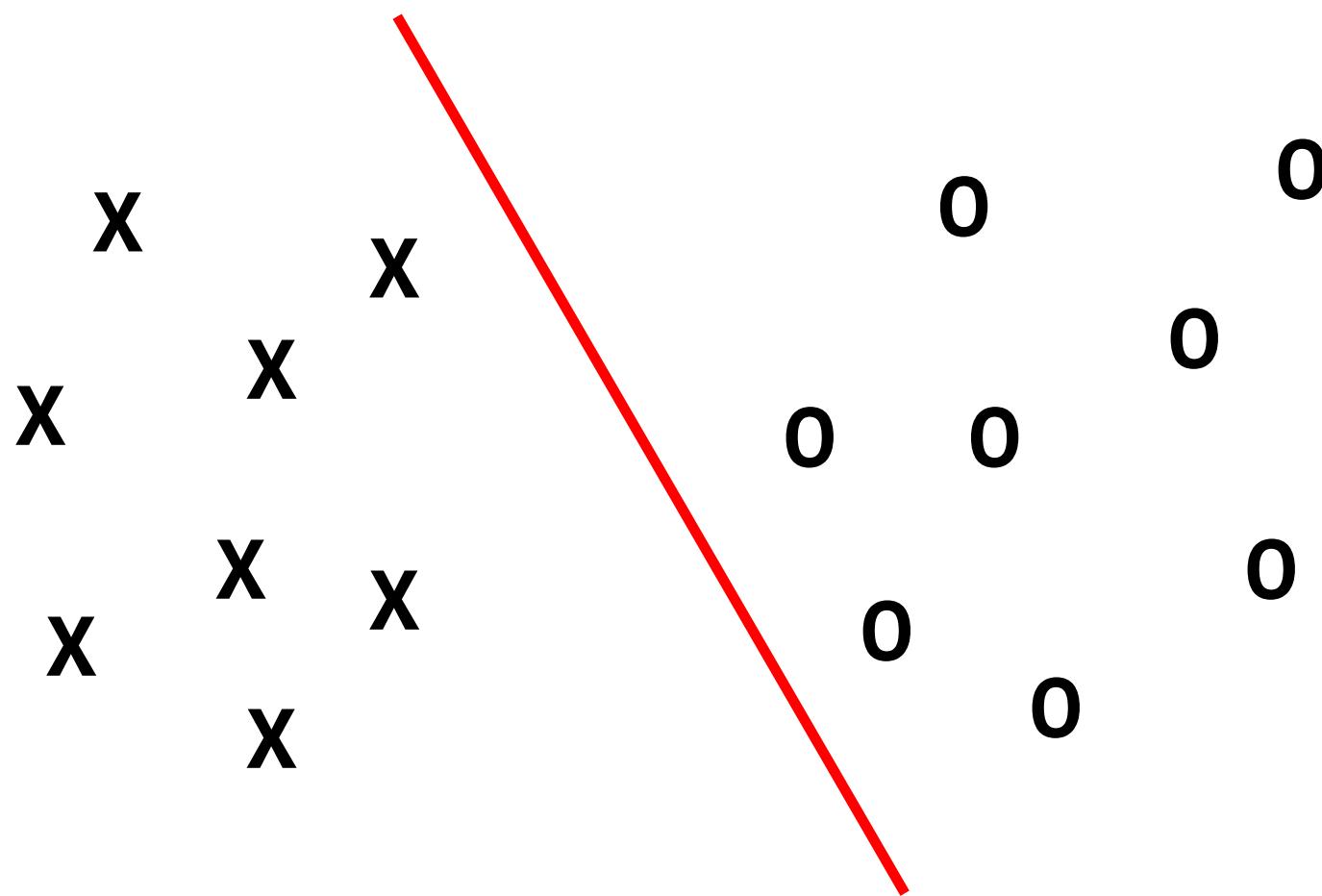
Intuitions



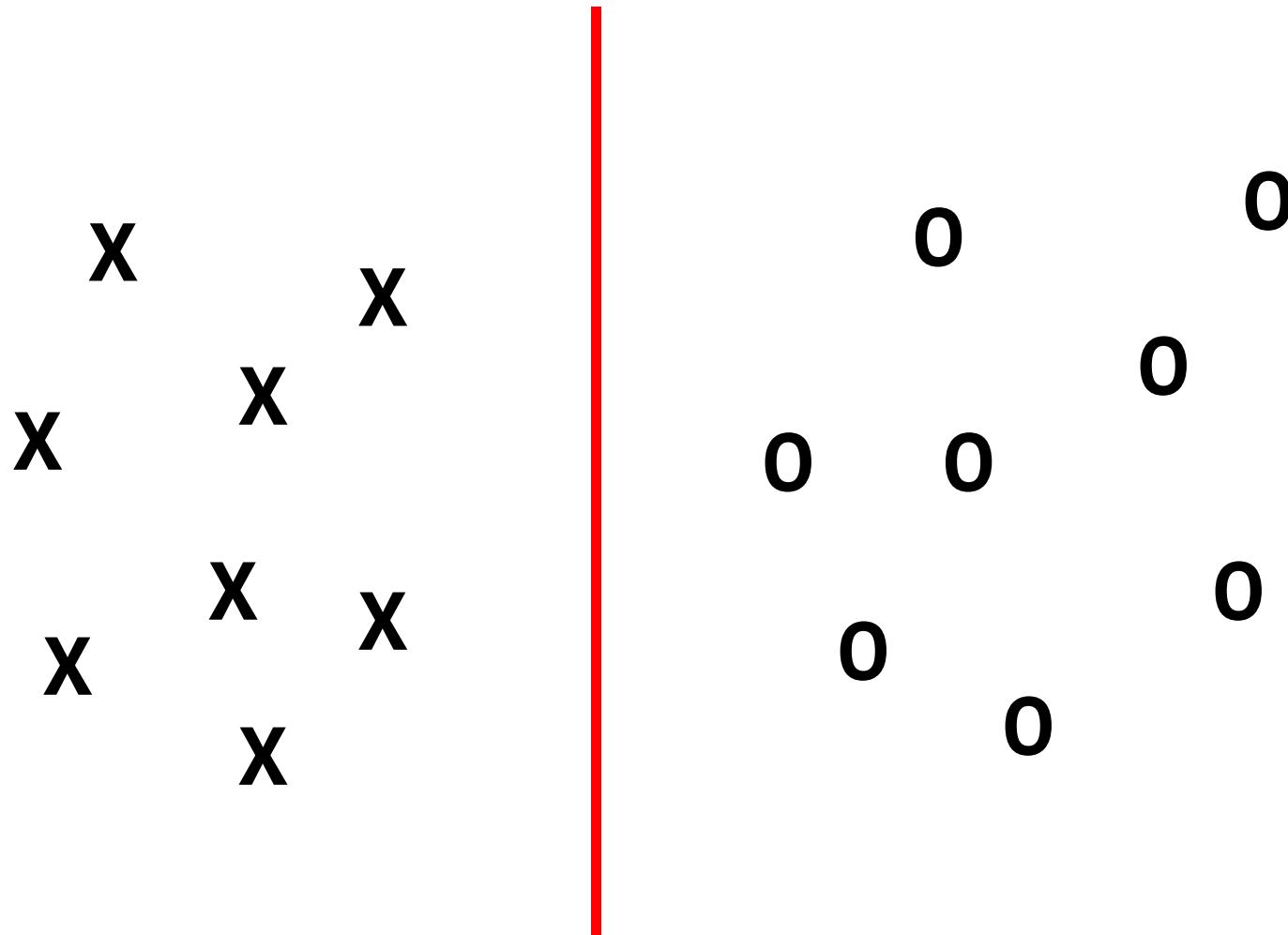
Intuitions



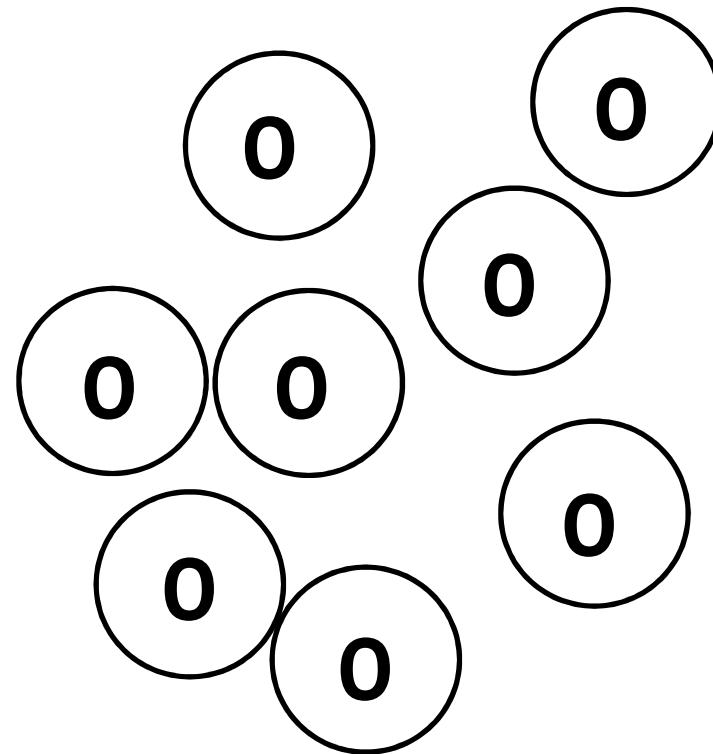
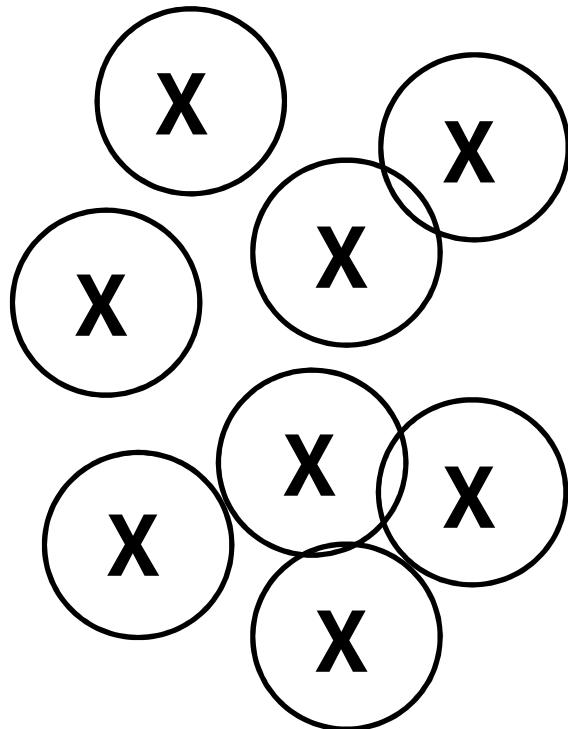
Intuitions



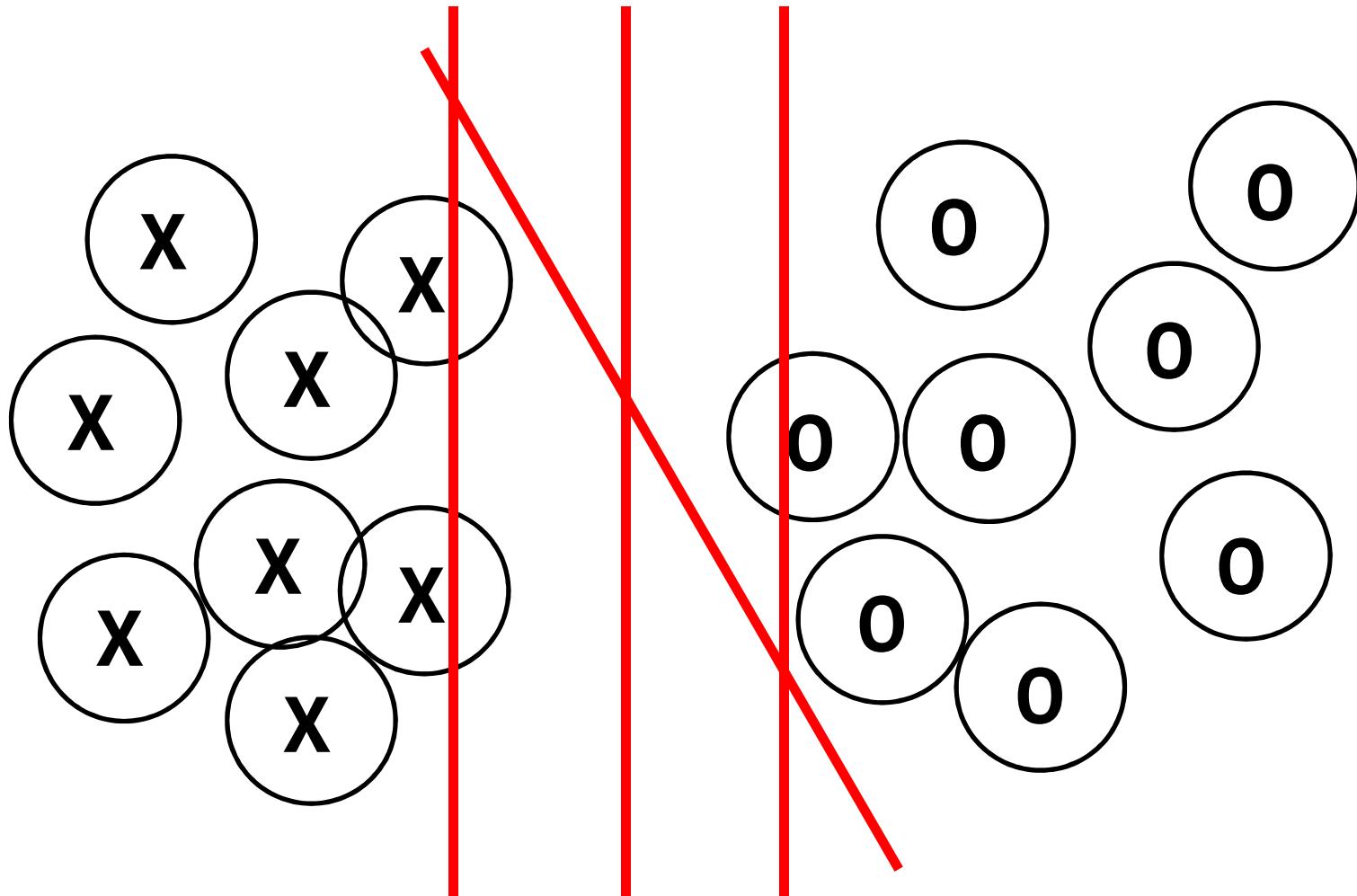
A “Good” Separator



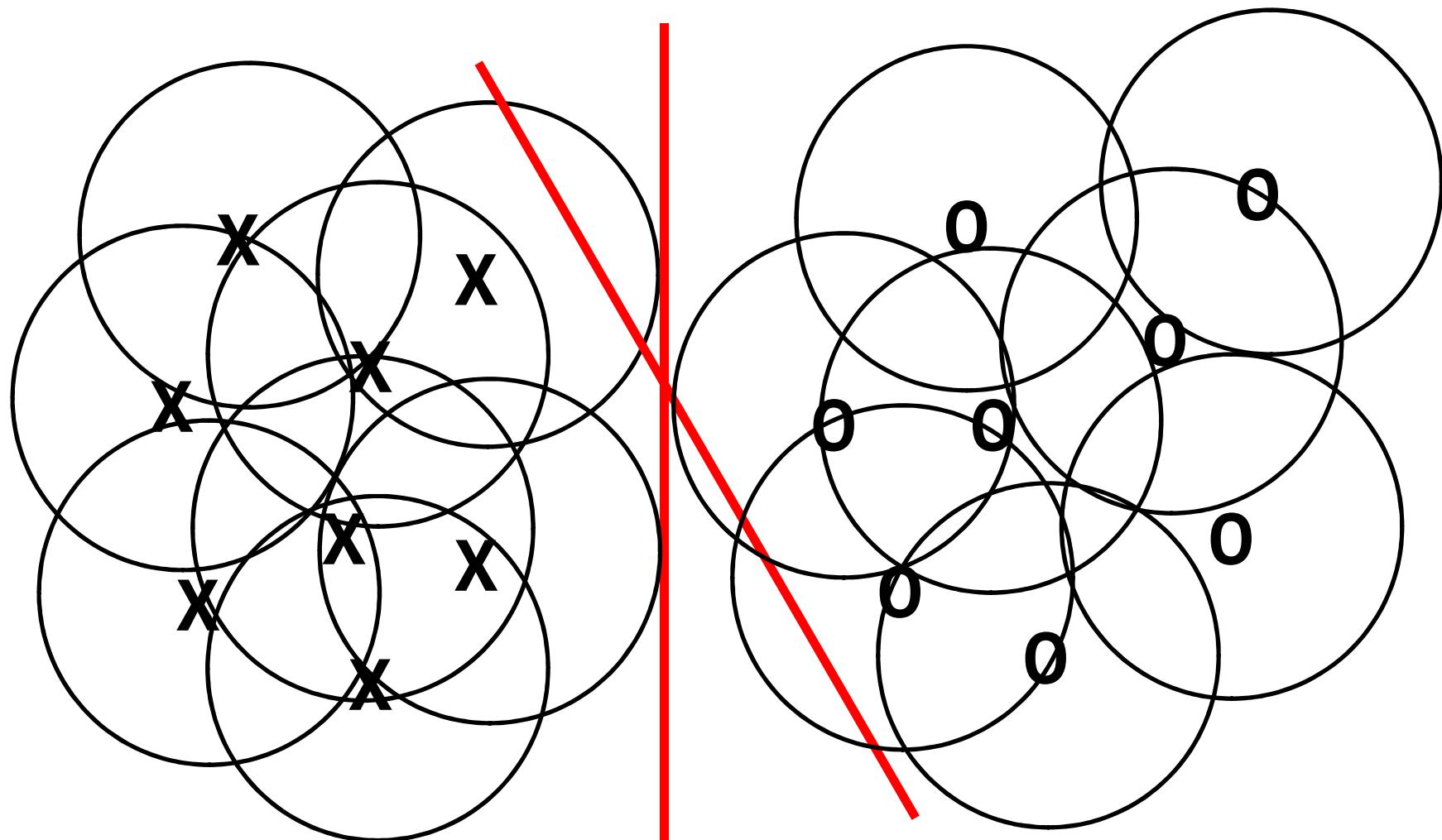
Noise in the Observations



Ruling Out Some Separators

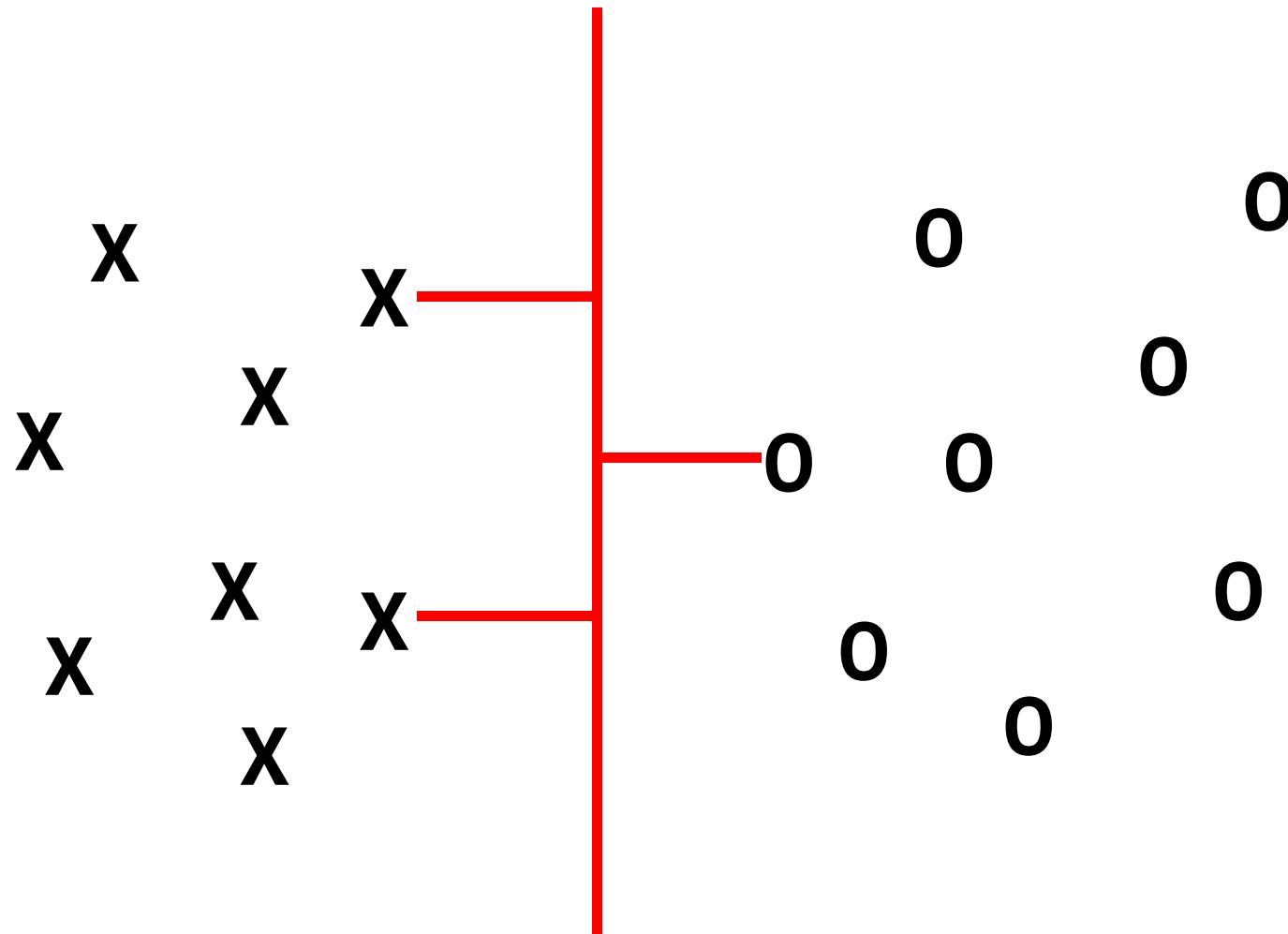


Lots of Noise



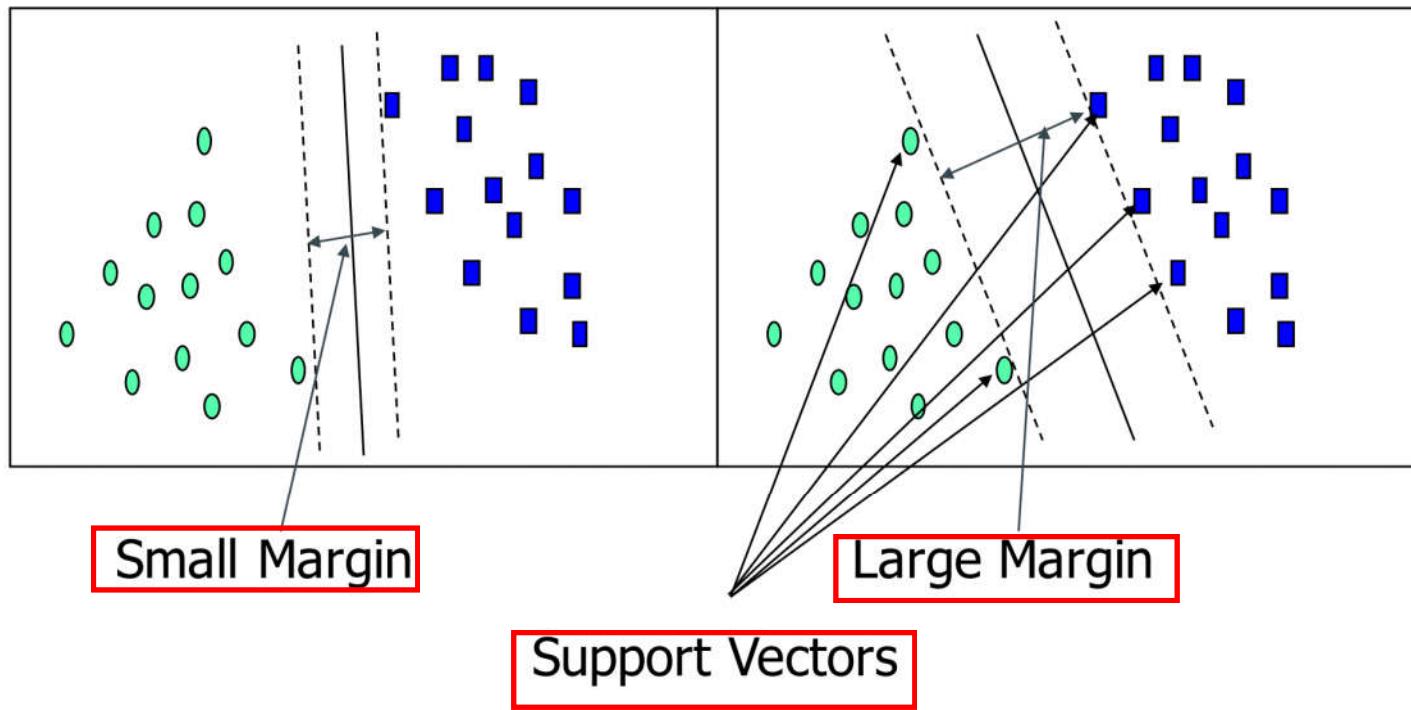
Maximizing the Margin

خط خوب خطی است که بیشترین فاصله
را از هر دو کلاس داشته باشد



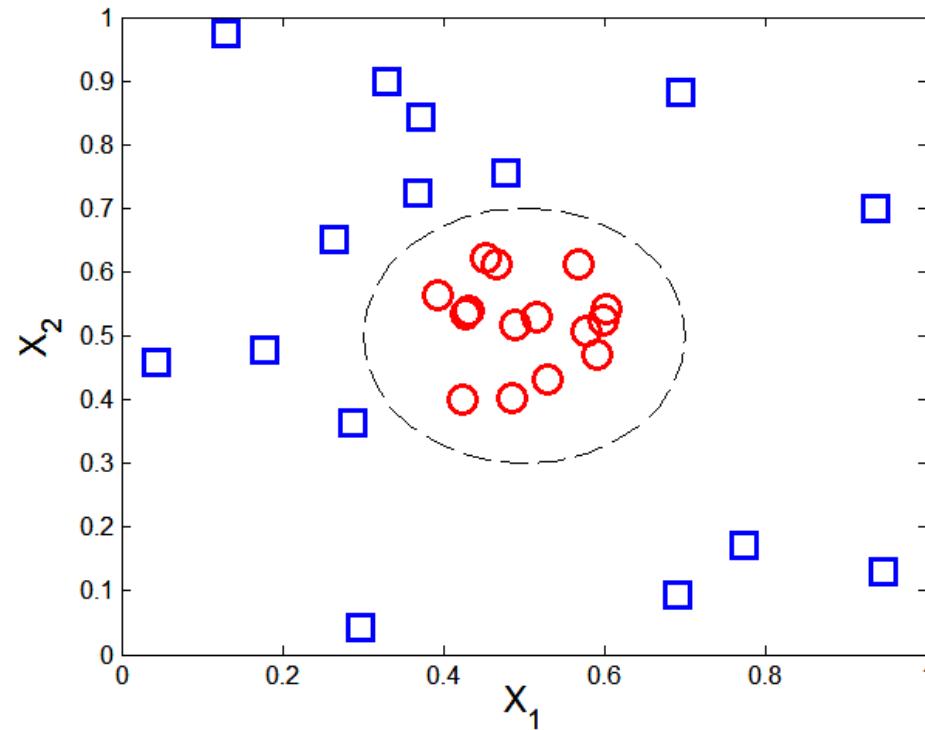
Maximizing the Margin

مارجین میشه فاصله
از کلاس ها



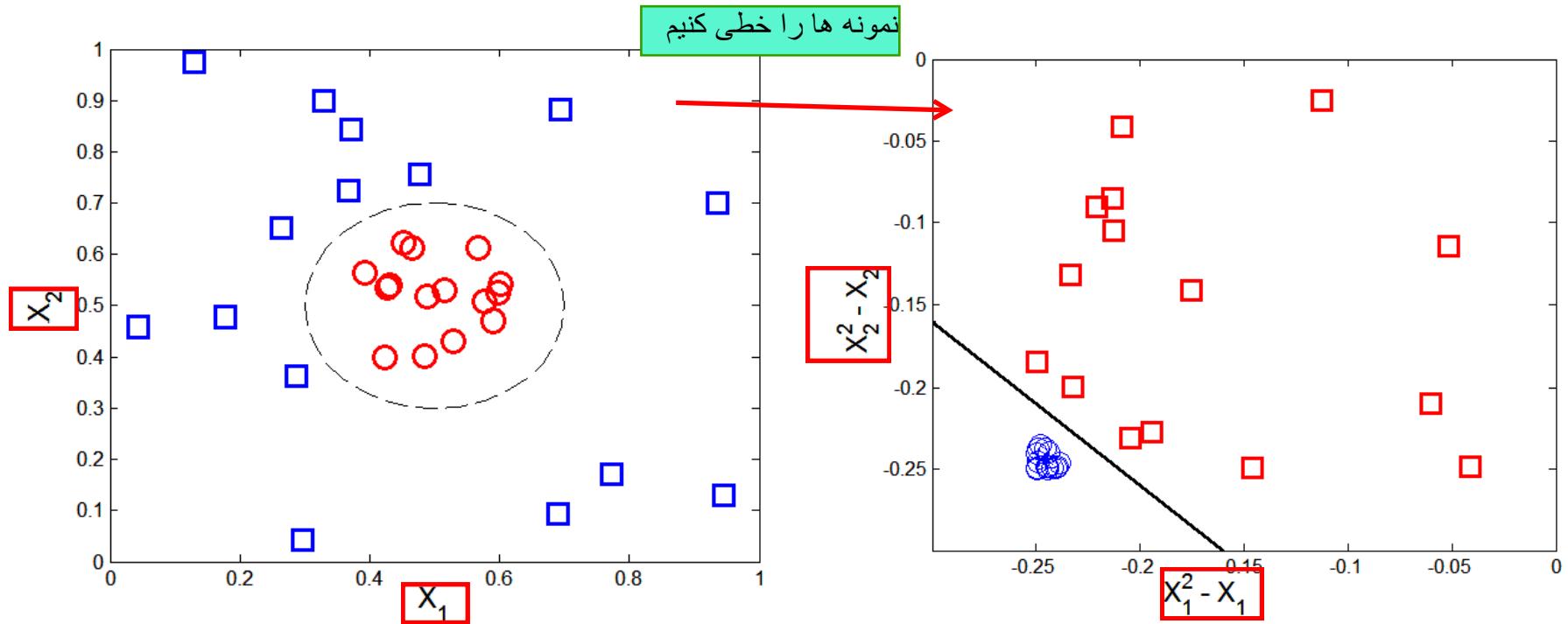
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Decision boundary:

$$\vec{w} \bullet \Phi(\vec{x}) + b = 0$$

Kernel Trick

داده ها را به فضای سطح بالاتری ببریم یا ابعاد بالاتر که راحت
تر بتوانیم داده هارا جدا کنیم

Learning Nonlinear SVM

- Kernel Trick:

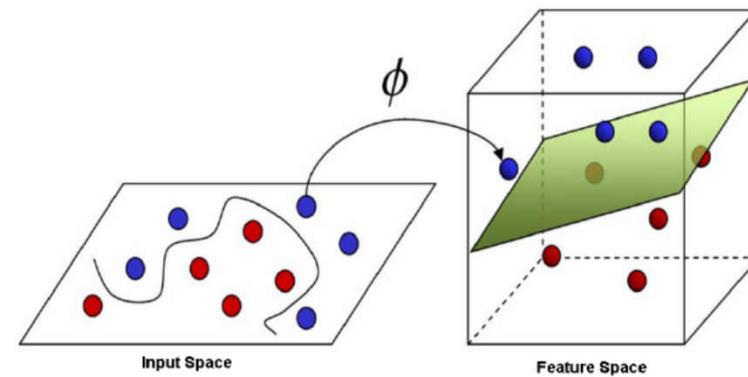
- $\Phi(x_i) \bullet \Phi(x_j) = K(x_i, x_j)$
- $K(x_i, x_j)$ is a kernel function (expressed in terms of the coordinates in the original space)

- ◆ Examples:

$$K(x, y) = (x \cdot y + 1)^p$$

$$K(x, y) = e^{-\|x-y\|^2/(2\sigma^2)}$$

$$K(x, y) = \tanh(kx \cdot y - \delta)$$



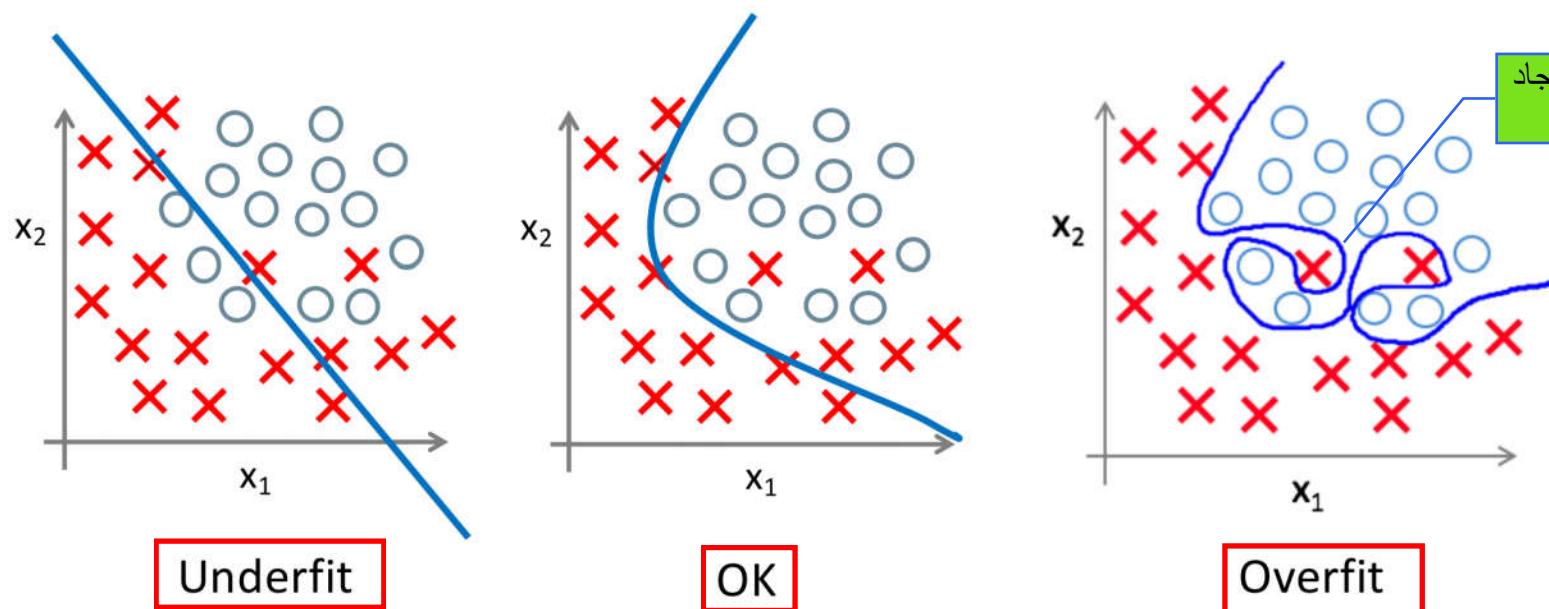
<http://thecaffeinedev.com/2-support-vector-machine-learning-math-behind-part2/>

Characteristics of SVM

- Robust to noise
- Overfitting is handled by maximizing the margin of the decision boundary,
- SVM can handle irrelevant and redundant attributes better than many other techniques
- The user needs to provide the type of kernel function and cost function
- Difficult to handle missing values
- What about categorical variables?

ویژگی های SVM مقاوم در برابر نویز تطبیق بیش از حد با به حداقل رساندن حاشیه مرز تصمیم گیری انجام می شود، SVM می تواند ویژگی های نامرتب و زائد را بهتر از بسیاری از تکنیک های دیگر مدیریت کند کاربر باید نوع عملکرد هسته وتابع هزینه را ارائه دهد رسیدگی به مقادیر از دست رفته مشکل است در مورد متغیر های طبقه بندی چطور؟

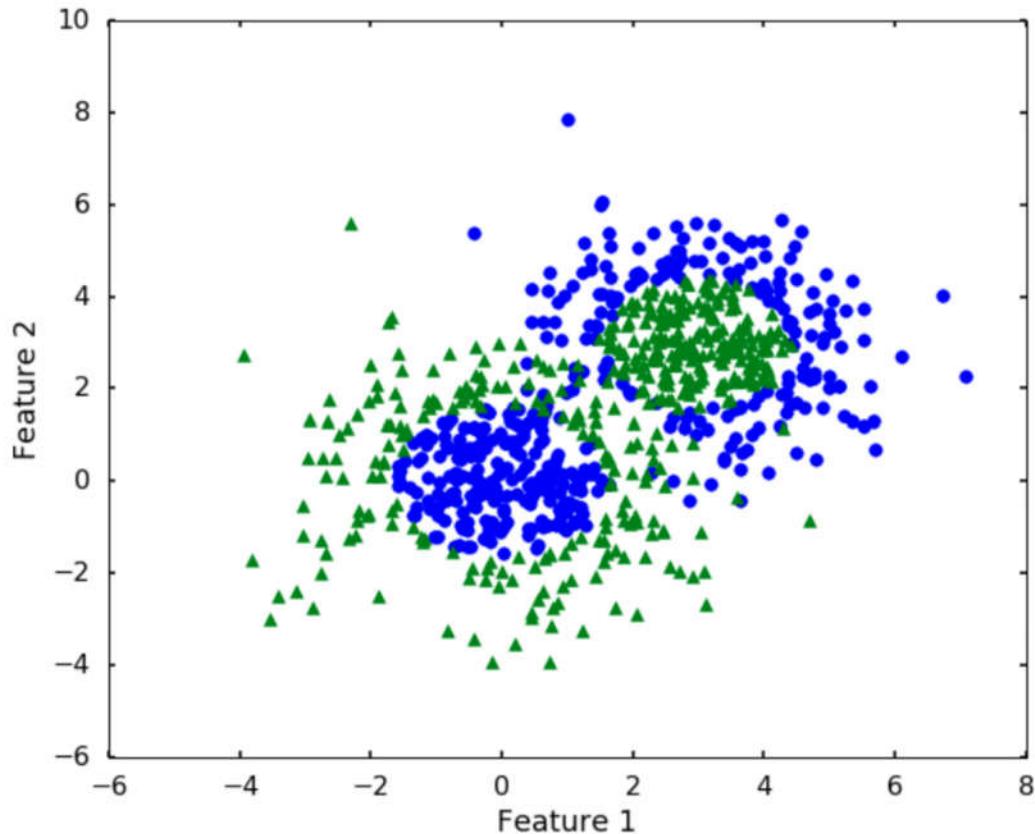
SVM



پیچیدگی زیادی ایجاد
کرده

In sklearn, this is controlled by **C** parameter of SVM and **gamma** parameter of rbf kernel

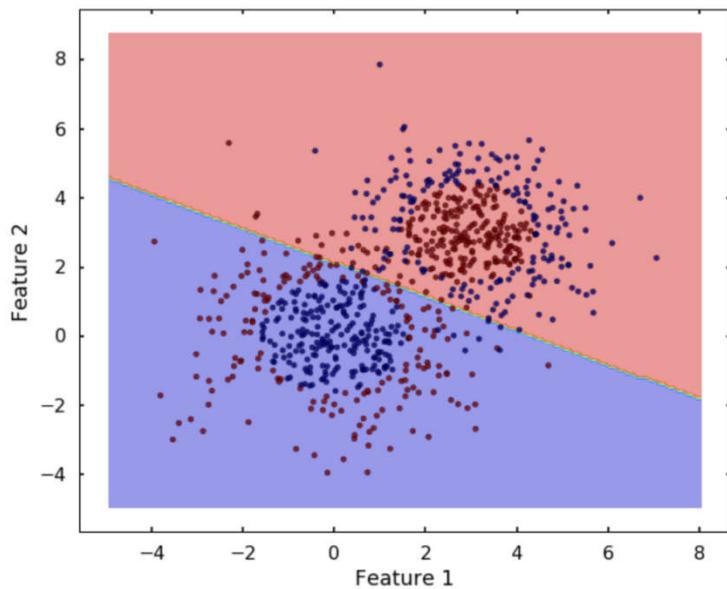
Python SVM



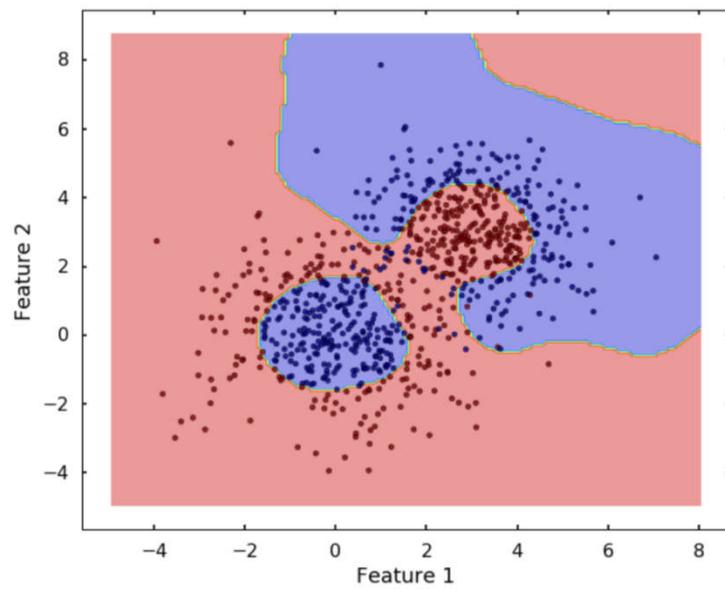
<http://qingkaikong.blogspot.com/2016/12/machine-learning-8-support-vector.html>

Python SVM

```
clf = svm.SVC(kernel='linear')
clf.fit(X,y)
```



```
clf = svm.SVC(kernel='rbf')
clf.fit(X,y)
```

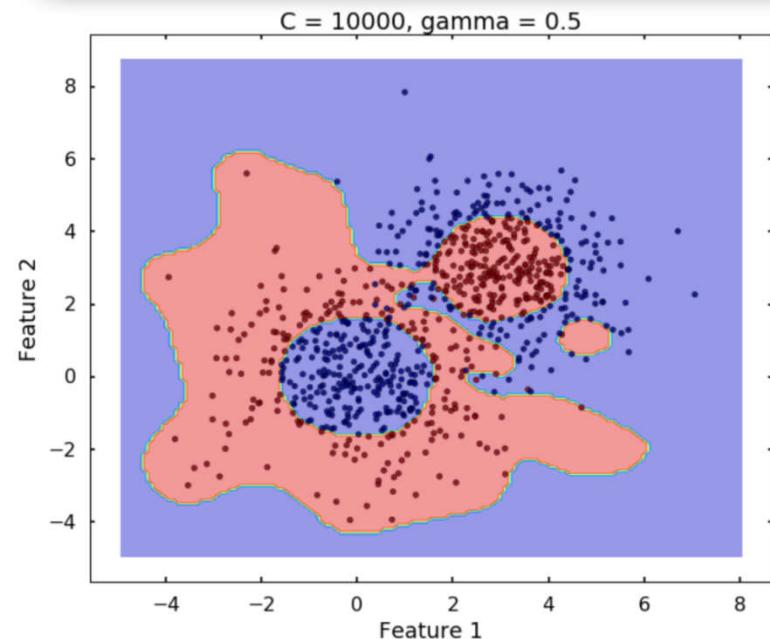


Python SVM

دنبال کم کردن خطای
روی داده های
اموزشی است

- A large C makes the cost of misclassification high.
- This will force the algorithm to fit the data with more flexible model.

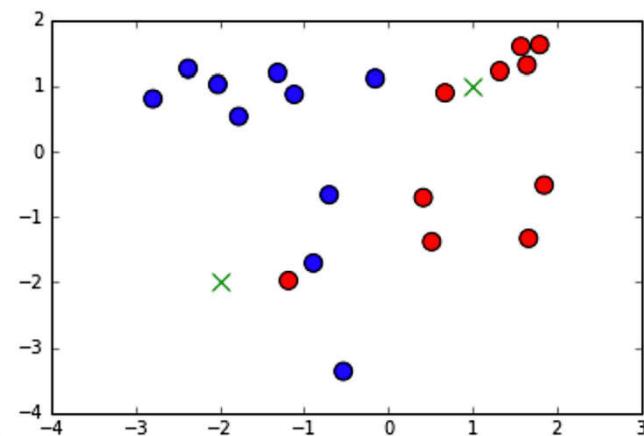
```
clf = svm.SVC(kernel='rbf', C = 10000, gamma = 0.5)
clf.fit(X,y)
```



Python KNN

```
from sklearn.datasets import make_classification
X, y = make_classification(n_features=2, n_redundant=0,
                           n_informative=2, n_samples=20)
```

```
plt.scatter(X[:,0], X[:,1], cmap='bwr', s=100, c=y)
plt.scatter([-2,1], [-2,1], marker='x', s=100, c='g')
plt.show()
```



Python KNN

```
from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier(n_neighbors=1)
clf.fit(X,y)
print('With k=1: ', clf.predict([[[-2,-2], [1,1]]]))

clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X,y)
print('With k=3: ', clf.predict([[[-2,-2], [1,1]]]))
```

```
With k=1:  [1 1]
With k=3:  [0 1]
```

ENSEMBLE LEARNING

Ensemble learning

- Motivations:

- Ensemble model improves accuracy and robustness over single model methods.
- A complex problem can be decomposed into multiple sub-problems that are easier to understand and solve (divide-and-conquer approach).

یادگیری گروهی
انگلیزه ها:

• مدل گروهی دقیق و استحکام را نسبت به روش های تک مدل بهبود می بخشد.

• یک مسئله پیچیده را می توان به چندین مساله فرعی که درک و حل آن آسان تر است تجزیه کرد (رویکرد تفرقه بیاندار و حکومت کن).

Ensemble learning is a machine learning technique that combines multiple individual models, called base learners or weak learners, to create a stronger and more accurate predictive model. It leverages the diversity and collective wisdom of multiple models to improve the overall performance and generalization ability. Ensemble learning has become popular in data mining and machine learning due to its ability to handle complex problems and enhance predictive accuracy.

Here are some different ways to implement ensemble learning:

Bagging (Bootstrap Aggregating): Bagging involves creating multiple base models, each trained on different random subsets of the training data (with replacement). The predictions of these models are combined, typically through voting or averaging, to make the final prediction. Examples of bagging ensembles include Random Forest and Extra Trees.

Example: Suppose we have a dataset of 1000 images, and we want to classify whether each image contains a cat or a dog. In bagging, we create multiple base models (e.g., decision trees) and train each model on a randomly sampled subset of the dataset. Each model predicts whether an image contains a cat or a dog. The final prediction is made by taking the majority vote of the predictions from all the models.

Boosting: Boosting is an iterative process that trains multiple base models sequentially. Each subsequent model focuses more on the misclassified instances from the previous models. The final prediction is made by combining the predictions of all the models, typically through weighted voting. Popular boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost.

Example: Let's consider the same image classification task. In boosting, we start by training a base model (e.g., decision tree) on the entire dataset. We then identify the misclassified instances and assign them higher weights. In the next iteration, we train another base model that focuses more on the misclassified instances. We repeat this process for several iterations, with each subsequent model giving more importance to the misclassified instances. Finally, we combine the predictions of all the models, typically by weighted voting, to make the final prediction.

Stacking: Stacking, also known as Stacked Generalization, involves training multiple base models on the same dataset and then using another model, called a meta-learner or blender, to combine their predictions. The meta-learner is trained on the predictions made by the base models. Stacking allows models to learn from each other's strengths and weaknesses, improving overall performance.

Example: Let's continue with the image classification task. In stacking, we train several base models (e.g., decision tree, random forest, support vector machine) on the entire dataset. Each model makes predictions on the images. Then, we create a new dataset using the predicted outputs of the base models as features. The meta-learner, such as logistic regression or neural network, is trained on this new dataset to make the final prediction.

Ensemble learning is a powerful approach to improve model performance, increase robustness, and enhance generalization in data mining and machine learning tasks.

python

 Copy code

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingClassifier, AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)

# Bagging
bagging_model = BaggingClassifier(n_estimators=10, random_state=42)
bagging_model.fit(X_train, y_train)
bagging_pred = bagging_model.predict(X_test)
bagging_accuracy = accuracy_score(y_test, bagging_pred)

# Boosting
boosting_model = AdaBoostClassifier(n_estimators=10, random_state=42)
boosting_model.fit(X_train, y_train)
boosting_pred = boosting_model.predict(X_test)
boosting_accuracy = accuracy_score(y_test, boosting_pred)
```

```
# Stacking
base_models = [LogisticRegression(), DecisionTreeClassifier(), RandomForestClassifier()]
stacking_model = LogisticRegression()
stacking_pred = np.zeros_like(y_test)

for model in base_models:
    model.fit(X_train, y_train)
    stacking_pred += model.predict(X_test)

stacking_pred /= len(base_models)
stacking_pred = np.round(stacking_pred).astype(int)
stacking_accuracy = accuracy_score(y_test, stacking_pred)

# Compare results
print("Bagging Accuracy:", bagging_accuracy)
print("Boosting Accuracy:", boosting_accuracy)
print("Stacking Accuracy:", stacking_accuracy)
```

Ensemble learning

Popular methods:

- Bagging
- Boosting
- Stacking

۳تا رویکرد برای شکستن مساله به بخش‌های کوچک‌تر

تکنیک‌هایی که نظر چندتا کلسیفایر را می‌پرسند. ایده: چون داریم از چندتا مدل استفاده می‌کنیم انتظار داریم نتایج بهتر باشند اور فیت شدن یکی کمتر می‌شود. یه مساله دیگه هم اینه که بعضی از مسائل اونقدر سخت هستند که نباید انتظار داشته باشیم یه کلسیفایر بتواند حلشون کنه به تنهایی

Bagging: Bagging involves training multiple base models on different subsets of the training data (with replacement). Each base model is trained independently, and their predictions are combined through voting or averaging to make the final prediction. The main focus is on reducing variance and improving model stability.

Boosting: Boosting also trains multiple base models, but in a sequential manner. Each subsequent model is trained to correct the mistakes made by the previous models. The predictions of these models are combined using weighted voting, with more weight given to the models that perform better. Boosting aims to reduce bias and improve overall accuracy.

Stacking: Stacking involves training multiple base models on the same dataset. The predictions of these models are then used as input features for a meta-learner or blender model. The meta-learner is trained to learn how to combine the predictions effectively, leveraging the diverse perspectives of the base models. Stacking aims to capture the strengths of different models and improve overall predictive performance.

Bagging: Bootstrap Aggregation

- Training

- Given a set D of d tuples, at each iteration i , a training set D_i of d tuples is sampled with replacement from D (i.e., bootstrap)
- A classifier model M_i is learned for each training set D_i
- Classification:** to classify an unknown sample X Each classifier M_i returns its class prediction • The bagged classifier M^* counts the votes and assigns the class with the most votes to X
- Regression:** take the average value instead of voting رگرسیون: به جای رای دادن، مقدار متوسط را بگیرید
- Bagging produces a combined model that often performs significantly better than the single model built from the original training data, and is never substantially worse.

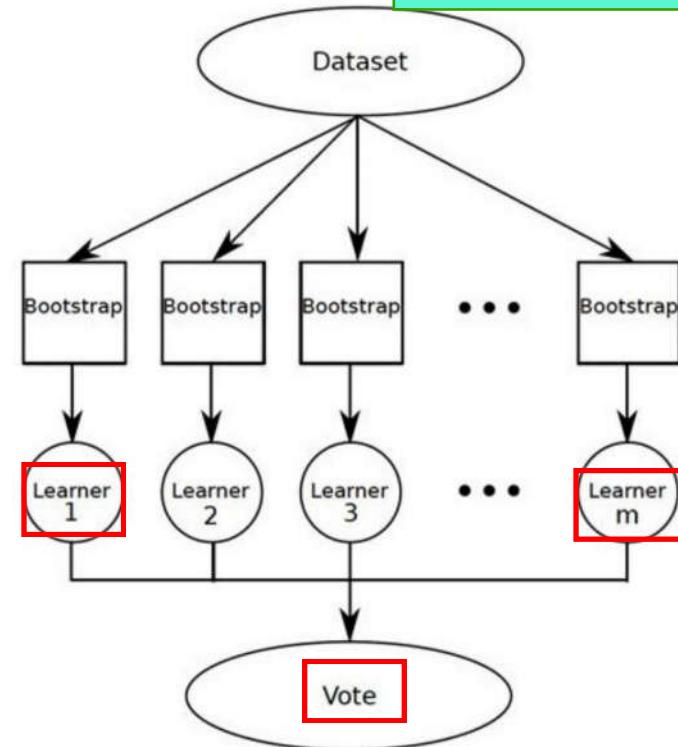
- Example

- Random forest

با یک تعداد زیادی درخت،
داده هامون را مدل میکنیم و
هر درختی مسئول یه بخشی از
داده هاست.

در مسائل رگرسن
یک میانگین از
خروجی این مدل ها
میگیریم به عنوان
نتیجه نهایی

کل داده ها را به چندین بخش تقسیم میکنیم و هر بخش را به یک مدل میدهیم. وقتی داده های جدیدی اومد بین این m تا مدل رای گیری میکنیم هرجوابی که بیشترین رای را اورد را به عنوان نتیجه نهایی در نظر میگیریم.

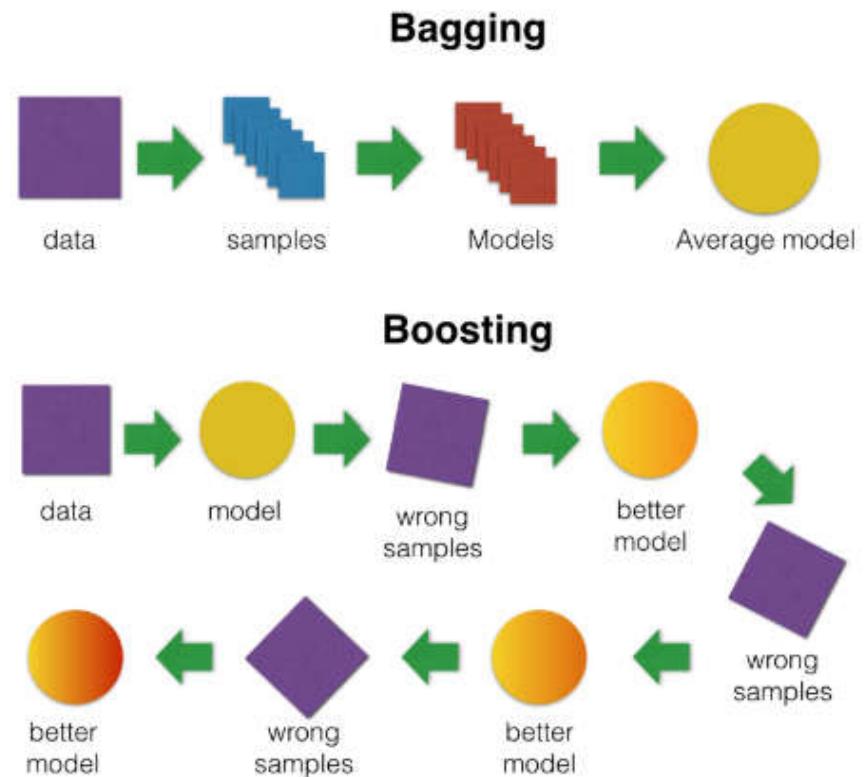


طبقه بندی: طبقه بندی یک نمونه ناشناخته X هر طبقه بندی کننده M_i پیشینی کلاس خود را بر میگرداند. طبقه بندی کننده کیسه ای M^* آرا را می شمارد و کلاسی را که بیشترین آرا را دارد به X اختصاص می دهد.

طبقه بندی: طبقه بندی یک نمونه ناشناخته X هر طبقه بندی کننده M_i پیشینی کلاس خود را بر میگرداند. طبقه بندی کننده کیسه ای M^* آرا را می شمارد و کلاسی را که بیشترین آرا را دارد به X اختصاص می دهد.

2. Boosting

- Comparing with bagging:
- The same base classifiers are used in both
- Boosting uses weighted voting/averaging
- Bagging is parallel while boosting is sequential
- Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data.



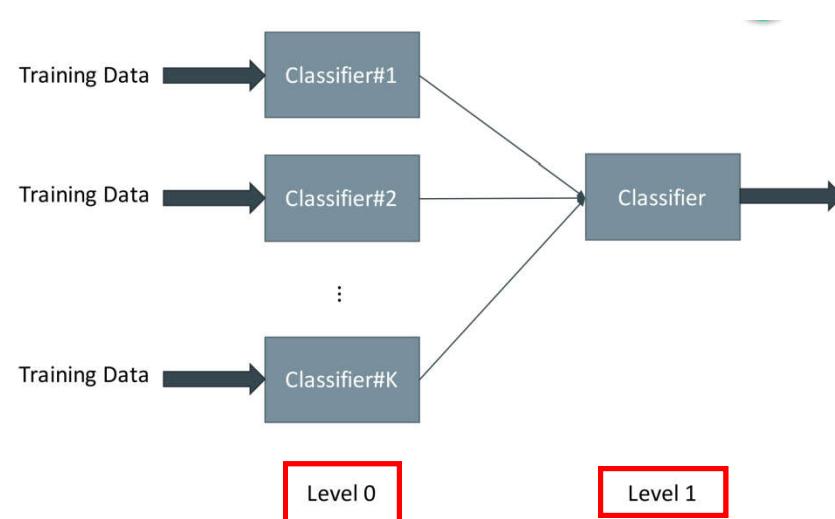
Source: bradzzz.gitbooks.io

2. تقویت
ماقیسه با bagging

طبقه بندی کننده های پایه یکسان در هر دو استفاده می شود
تقویت از رای گیری وزنی / میانگین استفاده می کند
بسته بندی موازی است در حالی که تقویت متوالی است
تقویت تمایل به دقیق بیشتری دارد، اما همچنین خطر تطبیق بیش از حد مدل
را با داده های طبقه بندی نادرست دارد.

3. Stacking

- It introduces the concept of a **metalearner**, which replaces the voting procedure.
- Normally is used to **combine** models of **different types**.
- Because most of the work is already done by the level-0 learners, it makes sense to choose a rather **simple algorithm** for the level-1 classifier.
- Use **out-of-fold predictions (OOF)** as the **training data** for the level 1 classifier.



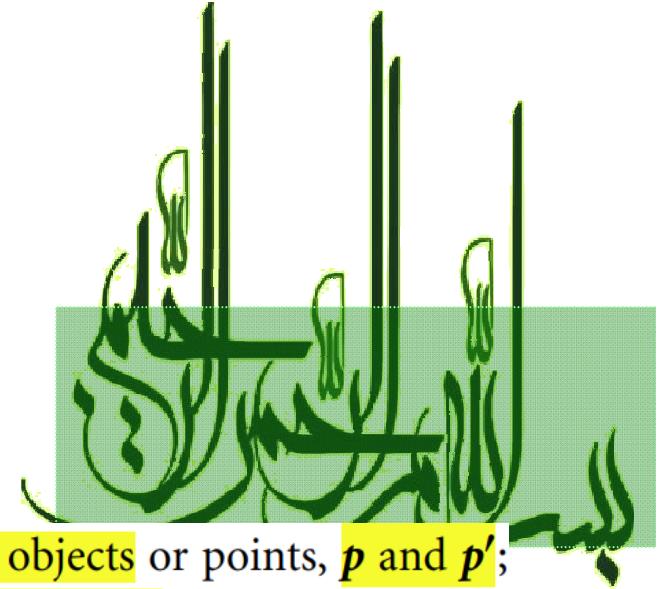
Stacking involves training multiple base models on the same dataset.

The predictions of the base models are used as input features for a meta-learner or blender model.

The meta-learner is trained on the predictions of the base models to make the final prediction.

Stacking aims to capture the strengths of different models and learn how to effectively combine their predictions.

The meta-learner can be a simple model like logistic regression or a more complex model like a neural network.



\mathbf{m}_i is the mean for cluster, C_i ; — distance between two objects or points, \mathbf{p} and \mathbf{p}' ;

Minimum distance:
$$dist_{min}(C_i, C_j) = \min_{\mathbf{p} \in C_i, \mathbf{p}' \in C_j} \{|\mathbf{p} - \mathbf{p}'|\}$$

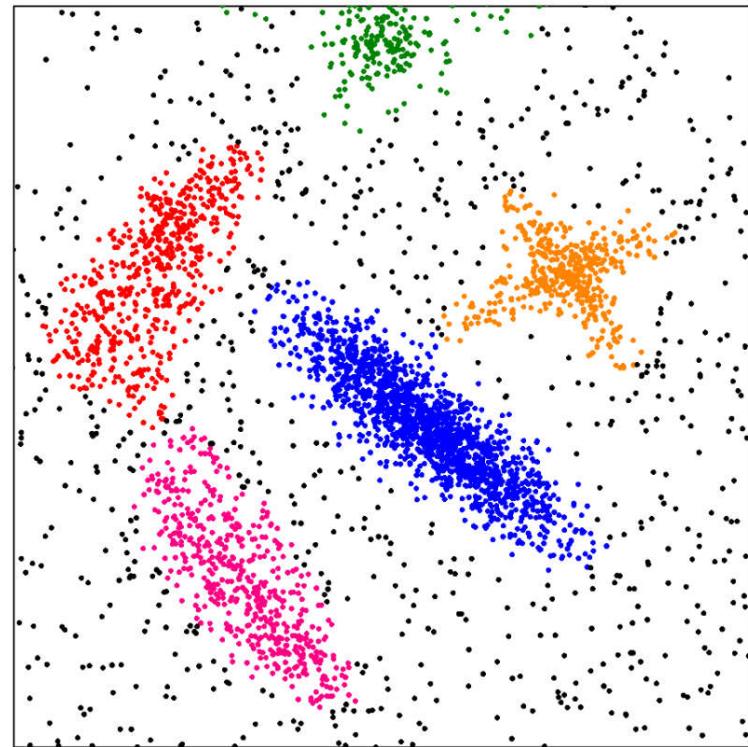
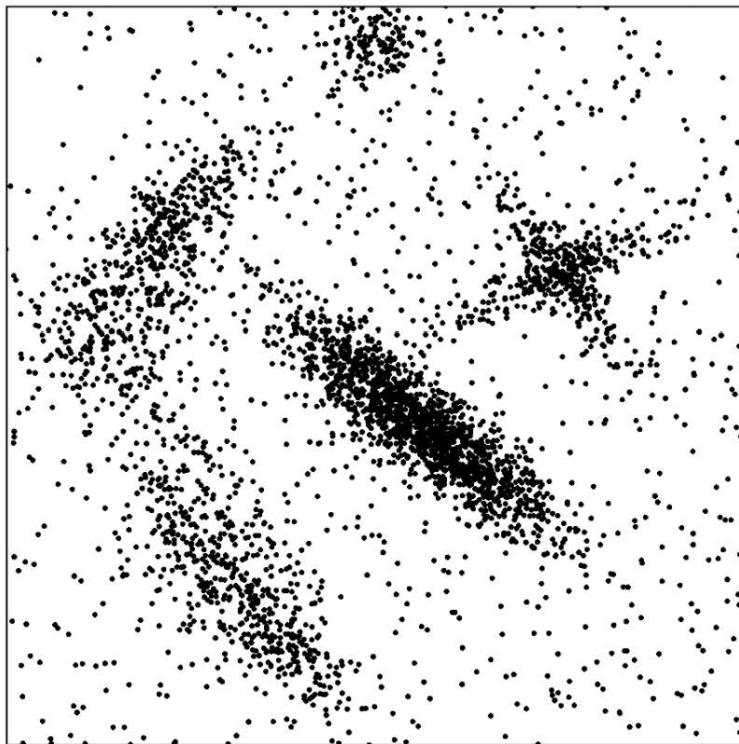
Maximum distance:
$$dist_{max}(C_i, C_j) = \max_{\mathbf{p} \in C_i, \mathbf{p}' \in C_j} \{|\mathbf{p} - \mathbf{p}'|\}$$

Mean distance:
$$dist_{mean}(C_i, C_j) = |\mathbf{m}_i - \mathbf{m}_j|$$

Average distance:
$$dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{\mathbf{p} \in C_i, \mathbf{p}' \in C_j} |\mathbf{p} - \mathbf{p}'|$$

What is Cluster Analysis?

میشه بگی هر کدام از این رکوردها
توی چه دسته هایی قرار دارند؟
چند تا دسته داریم کلا؟
براساس توده ها دسته بندی میکنیم

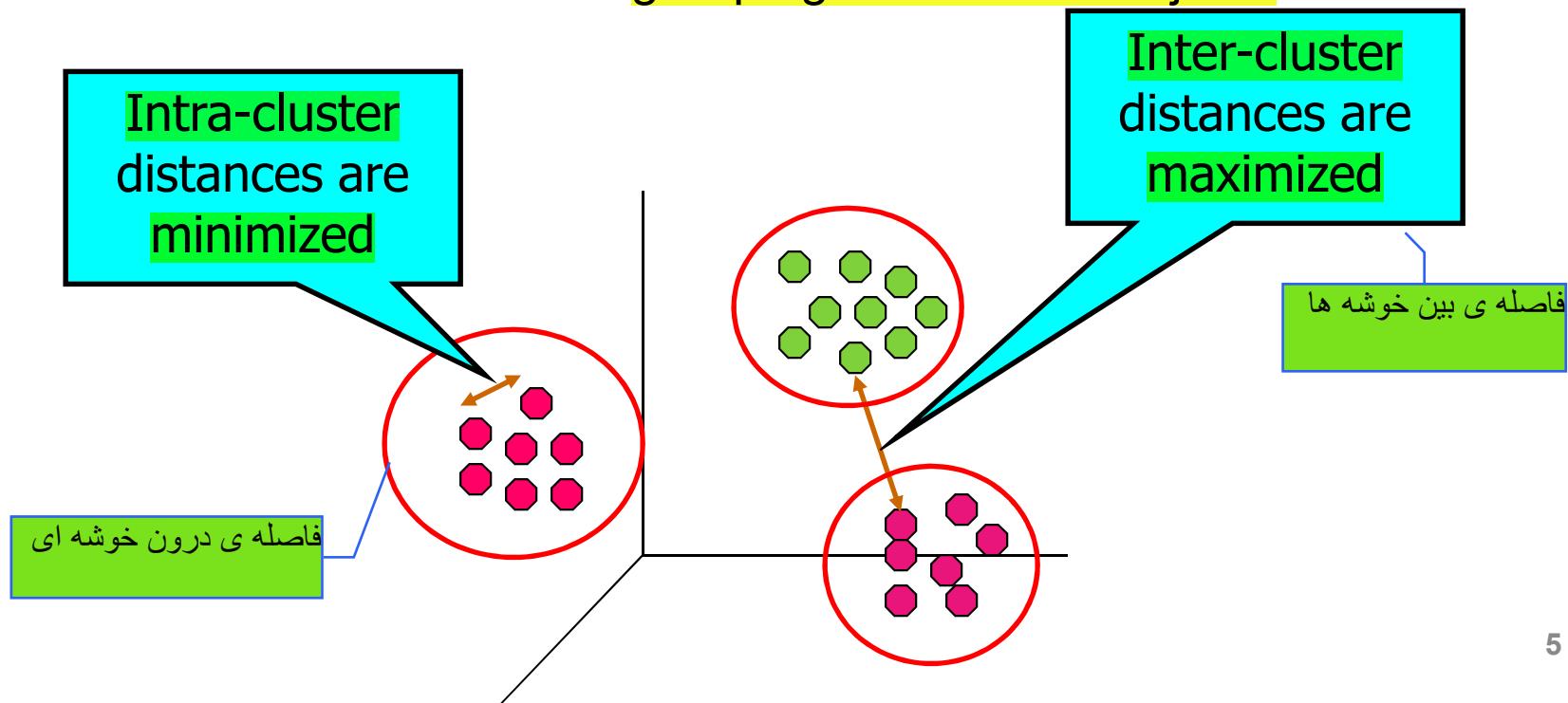


پس یه مشت داده میدن و میگن به یه تعداد دسته تقسیم کن و هر کدام از رکوردها
برای کدام دسته میشه را باید بگیم.

What is Cluster Analysis?

- **Cluster**: A collection of data objects
 - similar (or related) to one another within the same group
 - dissimilar (or unrelated) to the objects in other groups
- **Cluster analysis** (or *clustering, data segmentation, ...*)
 - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters

کلاستر: مجموعه ای از اجکت ها که دو تا ویژگی مهم دارند: ۱. این مجموعه از ویژگی ها توانی گروه شون شیوه به هم هستند.
۲. نسبت به بقیه گروه ها متفاوت هستند.



What is Cluster Analysis?

- **Unsupervised learning**: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical **applications**
 - As a **stand-alone tool** to get **insight** into **data distribution**
 - As a **preprocessing step** for other algorithms

وقتی یه دیتابی رو بهمون میدن دیگه نتیجشو بهمون نمیدن مدل باید از اختلاف ها و تشابه ها تشخیص بده

مثال از کمک کلاسترینگ در مبحث پری پراسسینگ:
اگر به جای ۲ میلیون رکورد بخاییم ۱۰۰ رکورد برداریم از داده هامون و باونها کار کنیم کدوم ۱۰۰ را برداریم تا نماینده خوبی از داده ها باشند؟
مثلای زمانی که ابعاد داده ها زیاد است بهتر است تعداد رکورد کمتری انتخاب کنیم

اماده کردن یه
برچسب اولیه از داده
ها

Clustering can be used as a preprocessing step in data mining to improve the performance of subsequent algorithms by organizing and grouping similar instances together. By clustering the data, we can discover underlying patterns and structures, reduce the dimensionality, and enhance the interpretability of the data. Here's an example to illustrate how clustering can be used as a preprocessing step:

Let's consider a customer segmentation problem for a retail company. The company has a dataset with various customer attributes such as age, income, spending habits, and purchase history. The goal is to group similar customers together to gain insights for targeted marketing strategies.

Clustering as Preprocessing:

As a preprocessing step, clustering can be applied to group similar customers together based on their attributes. One popular clustering algorithm is K-means. Let's assume we want to create four customer segments.

python

 Copy code

```
from sklearn.cluster import KMeans  
  
# Assume 'X' is the dataset containing customer attributes  
  
# Apply K-means clustering  
kmeans = KMeans(n_clusters=4, random_state=42)  
clusters = kmeans.fit_predict(X)
```

The resulting clusters assign each customer to one of the four segments based on their similarities.

Analyzing Cluster Characteristics:

Once the customers are clustered, we can analyze the characteristics of each segment. For example, we can calculate the average income and spending habits for each segment or visualize the distribution of age across clusters. This analysis provides insights into the different customer profiles present in the dataset.

Applying Algorithms on Segmented Data:

After clustering, we can use the identified customer segments as input for subsequent algorithms. For instance, we can apply classification algorithms separately to each segment to predict customer churn, or we can apply recommendation algorithms to provide personalized product recommendations within each segment.

python

 Copy code

```
from sklearn.ensemble import RandomForestClassifier
# Assume 'y' represents the target variable for customer churn prediction

# Apply RandomForestClassifier to each customer segment
for segment in range(4):
    segment_X = X[clusters == segment]
    segment_y = y[clusters == segment]

    model = RandomForestClassifier()
    model.fit(segment_X, segment_y)
    # Make predictions or evaluate model performance within each segment
```

By treating each cluster as a separate subset, we can apply algorithms on more homogeneous groups, potentially improving the accuracy and effectiveness of the subsequent models.

Using clustering as a preprocessing step allows us to organize the data into meaningful groups and tailor subsequent algorithms accordingly. It helps in discovering patterns, reducing complexity, and enabling targeted analysis or modeling on each identified cluster, leading to more accurate and personalized results.

Let's consider a dataset containing information about customers of an e-commerce platform. The dataset includes attributes such as age, income, and browsing behavior. The goal is to perform customer segmentation for targeted marketing campaigns.

Clustering as Preprocessing:

As a preprocessing step, we can apply a clustering algorithm to group similar customers together based on their attributes. Let's use K-means clustering as an example.

python

 Copy code

```
from sklearn.cluster import KMeans
# Assume 'X' is the dataset containing customer attributes

# Apply K-means clustering
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X)
```

The clustering algorithm assigns each customer to one of the clusters based on the similarity of their attributes.

Feature Engineering using Cluster Assignments:

Once the clustering is performed, we can create new features based on the cluster assignments. These new features capture the customer's cluster membership, which can be useful in subsequent algorithms.

python

 Copy code

```
import pandas as pd
# Assume 'X' is a DataFrame containing customer attributes

# Add cluster assignment as a new feature
X['Cluster'] = clusters

# Perform one-hot encoding on the 'Cluster' feature
X_encoded = pd.get_dummies(X, columns=['Cluster'])
```

By encoding the cluster assignments as new features, we provide additional information to the subsequent algorithms.

3. Applying Other Algorithms on Enhanced Data:

After the clustering and feature engineering steps, we can apply other algorithms on the enhanced dataset. These algorithms can benefit from the additional insights gained through clustering.

python

 Copy code

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Assume 'y' represents the target variable for classification

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# Apply RandomForestClassifier on the enhanced data
model = RandomForestClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
```

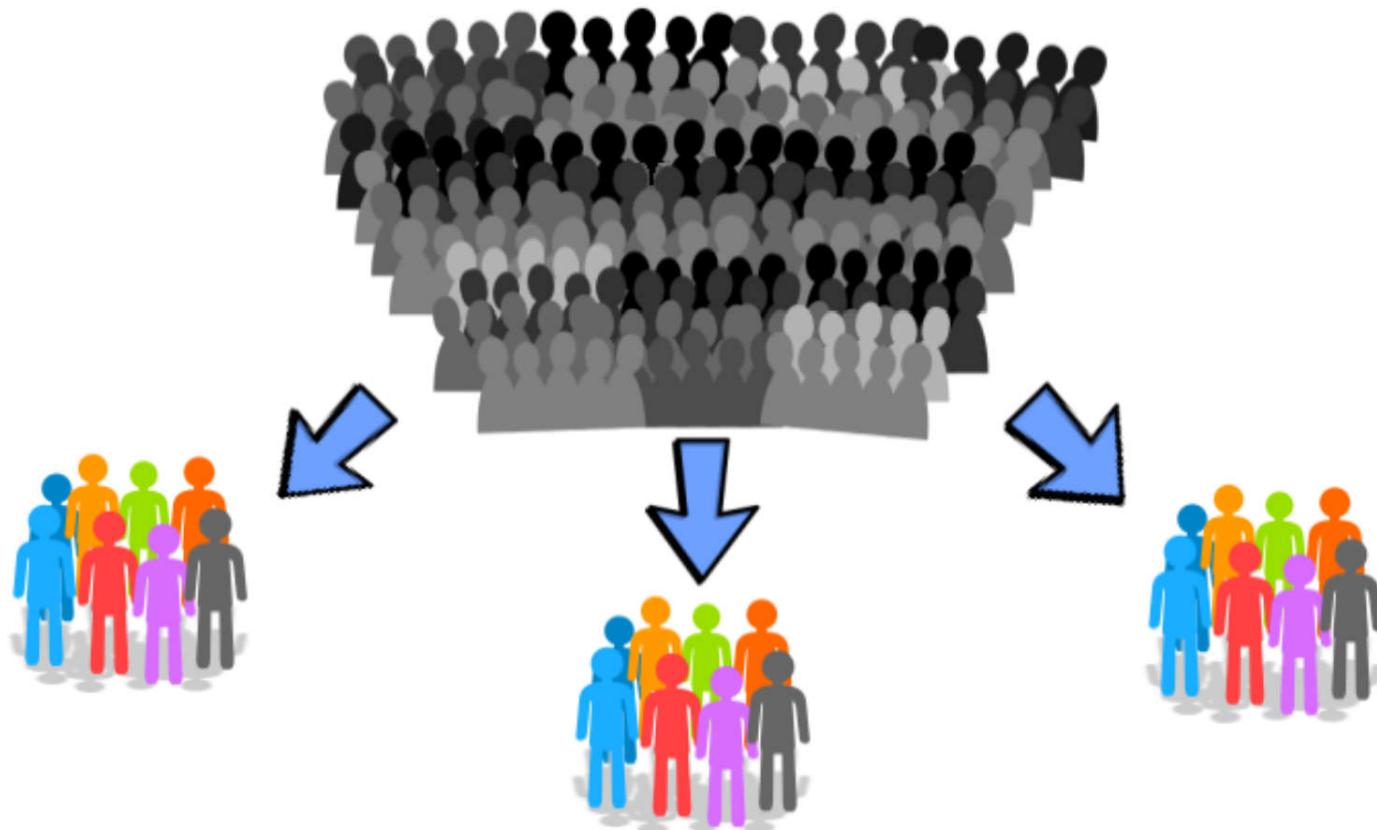
By using the enhanced dataset with cluster information as additional features, the subsequent algorithm (in this case, a RandomForestClassifier) can potentially achieve better performance and accuracy.

In this example, clustering is used as a preprocessing step to group similar customers together. Then, the cluster assignments are encoded as new features, which are subsequently used in conjunction with other algorithms for classification or other tasks. This approach can help capture underlying patterns and relationships in the data, leading to improved results in subsequent algorithms.

Example: customer segmentation

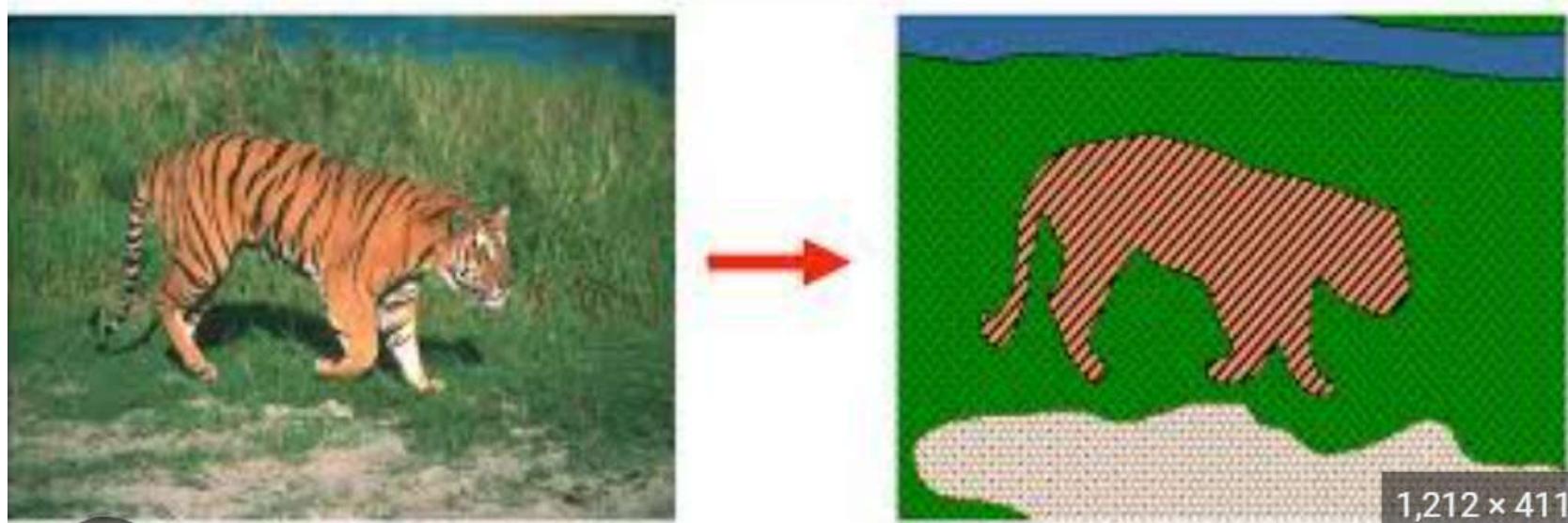
به دلیل تنوع بالای مشتری ها از کلاسترینگ استفاده میکنیم

خیلی خوب میشه که دسته بندی کنیم مشتری ها را برای هر دسته یه استراتژی جداگانه در نظر بگیریم.



Example: image segmentation

در پردازش تصویر خیلی مهم میشه که اطلاعات
مهمون کجای عکسه؟
مثلًا جداکردن بکگراند از فورگراند.
مثلًا در مثل زیر جداکردن حیوان از سبزه ها



Example: information retrieval

الآن مهمه که موتور
جستجو هر دو تا موضوع
جگوار را به ما پیشنهاد
ده هم ماشین هم حیوانش
را

Google jaguar

All Images News Videos Books More Tools

car cat f type f pace drawing wallpaper leopard GU

W Wikipedia Jaguar - Wikipedia

Yellow National Geographic Jaguar, facts and photos

Blue Encyclopedia Britannica Jaguar | Habitat, Diet, & Fact...

Black St. Louis Zoo Saint Louis Zoo | Jaguar

Green ia.wikipedia.org Jaguar - Wikipedia, le encyclopedia libere

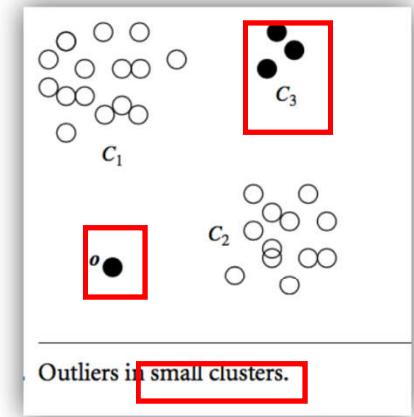
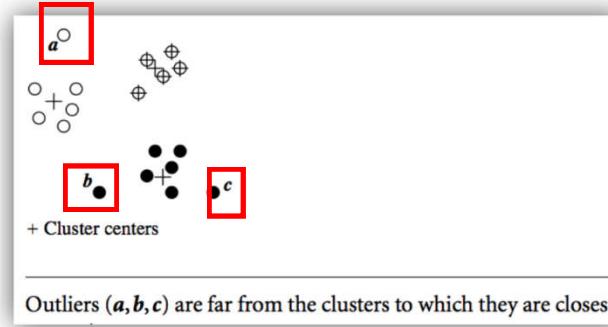
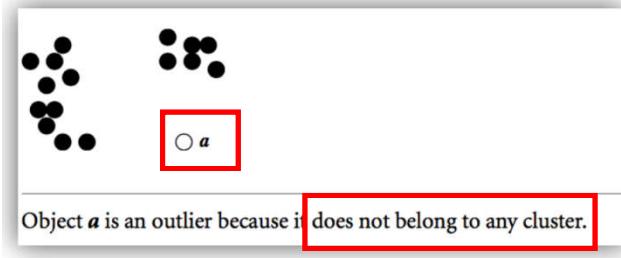
Red CarWale Jaguar XF Price - Images, Colours ...

Blue www.jaguar.co.uk Luxury Sports Cars, Executive Saloons ...

Example: outlier detection

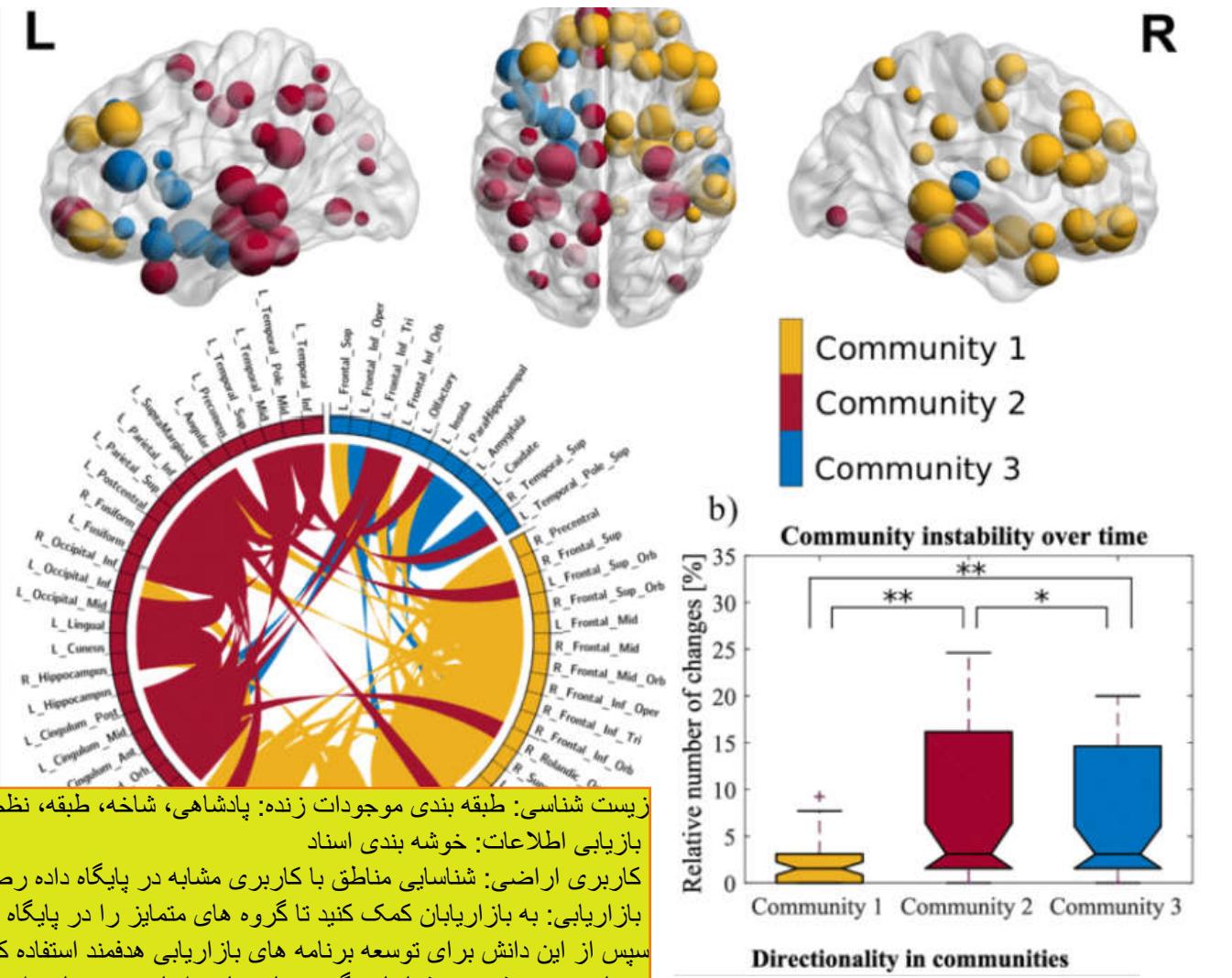
- What are outliers?
- The set of objects are considerably dissimilar from the remainder of the data
- Outlier detection is useful in fraud detection applications such as creditcard fraud detection.
- It is also useful as a preprocessing tool.
- One way for outlier detection is using clustering:
 - Objects that do not belong to any cluster
 - Objects far from other objects in the same cluster
 - Clusters of very small cardinality

Example: outlier detection



Refer to chapter 12 of the third edition of “Data Mining: Concepts and Techniques” for more information.

Example: community detection



ریست شناسی: طبقه بندی موجودات زنده: پادشاهی، شاخه، طبقه، نظم، خانواده، جنس و گونه

بازیابی اطلاعات: خوش بندی اسناد

کاربری اراضی: شناسایی مناطق با کاربری مشابه در پایگاه داده رصد زمین

بازاریابی: به بازاریابان کمک کنید تا گروه های متمایز را در پایگاه منتشریان خود کشف کنند و سپس از این دانش برای توسعه برنامه های بازاریابی هدفمند استفاده کنند.

برنامه ریزی شهری: شناسایی گروه خانه ها بر اساس نوع خانه، ارزش و موقعیت جغرافیایی.

مطالعات زمین لرزه: کانون های زمین لرزه مشاهده شده باید در امتداد گسل های قاره قرار گیرند.

آب و هوا: درک آب و هوای زمین، یافتن الگوهای جوی و اقیانوسی

علوم اقتصادی: تحقیقات بازار

Clustering for Data Understanding and Applications

- **Biology:** taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- **Information retrieval:** document clustering
- **Land use:** Identification of areas of similar land use in an earth observation database
- **Marketing:** Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **City-planning:** Identifying groups of houses according to their house type, value, and geographical location
- **Earthquake studies:** Observed earth quake epicenters should be clustered along continent faults
- **Climate:** understanding earth climate, find patterns of atmospheric and ocean
- **Economic Science:** market research

Clustering as a **Preprocessing Tool** **(Utility)**

- **Summarization:**
 - Preprocessing for **regression**, **PCA**, **classification**, and **association analysis**
- **Compression:**
 - **Image processing**: vector quantization
- **Finding K-nearest Neighbors**
 - **Localizing search** to one or a small number of clusters
- **Outlier detection**
 - Outliers are often viewed as those “**far away**” from any cluster

Clustering can be used as a preprocessing tool to find the K-nearest neighbors (KNN) more efficiently by reducing the search space. Here's an example to demonstrate how clustering can be used for finding K-nearest neighbors:

1. Clustering Step:

First, we apply a clustering algorithm, such as K-means, to group similar data points together.

Let's assume we have a dataset of points represented by their (x, y) coordinates.

python

 Copy code

```
from sklearn.cluster import KMeans
# Assume 'X' is the dataset containing data points

# Apply K-means clustering
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X)
```

The clustering algorithm assigns each data point to one of the clusters based on their similarity.

2. Indexing and Nearest Neighbor Search:

After clustering, we build an index structure for each cluster to efficiently search for K-nearest neighbors within that cluster.

python

 Copy code

```
import numpy as np
from scipy.spatial import KDTree

# Create a KDTree index for each cluster
cluster_indexes = []
for cluster_id in range(kmeans.n_clusters):
    cluster_points = X[clusters == cluster_id]
    cluster_kdtree = KDTree(cluster_points)
    cluster_indexes.append(cluster_kdtree)
```

By creating KDTree indexes for each cluster, we can perform fast nearest neighbor searches within specific clusters.

3. Finding K-nearest Neighbors:

When searching for the K-nearest neighbors of a given query point, we can utilize the clustering information to narrow down the search space to the relevant clusters, reducing the overall computation.

python

 Copy code

```
# Assume 'query_point' is the point for which we want to find K-nearest neighbors

# Find the relevant clusters for the query point
query_cluster = kmeans.predict(np.array([query_point]))[0]

# Perform K-nearest neighbor search within the relevant cluster
cluster_kdtree = cluster_indexes[query_cluster]
dists, indices = cluster_kdtree.query(np.array([query_point]), k=k)
nearest_neighbors = cluster_points[indices[0]]
```

By first identifying the relevant cluster for the query point, we can limit the nearest neighbor search to that cluster's index structure, resulting in a more efficient search process.

In this example, clustering is used as a preprocessing step to group similar data points together. Then, an index structure (KDTree) is built for each cluster, allowing for efficient K-nearest neighbor search within the relevant clusters. This approach reduces the computational complexity by limiting the search space to clusters likely to contain the nearest neighbors, thus improving the efficiency of finding K-nearest neighbors.

Traditional Hierarchical Clustering and Non-traditional Hierarchical Clustering differ in the way they construct hierarchical clusters and the criteria they use for merging or splitting clusters. Here's a comparison between the two approaches:

Traditional Hierarchical Clustering:

Traditional Hierarchical Clustering starts with each data point as a separate cluster and iteratively merges or agglomerates clusters until a stopping criterion is met. Two commonly used methods for agglomerative clustering are Single Linkage and Complete Linkage.

Single Linkage: This method merges the two clusters that have the closest pair of points based on the minimum distance between any two points from different clusters.

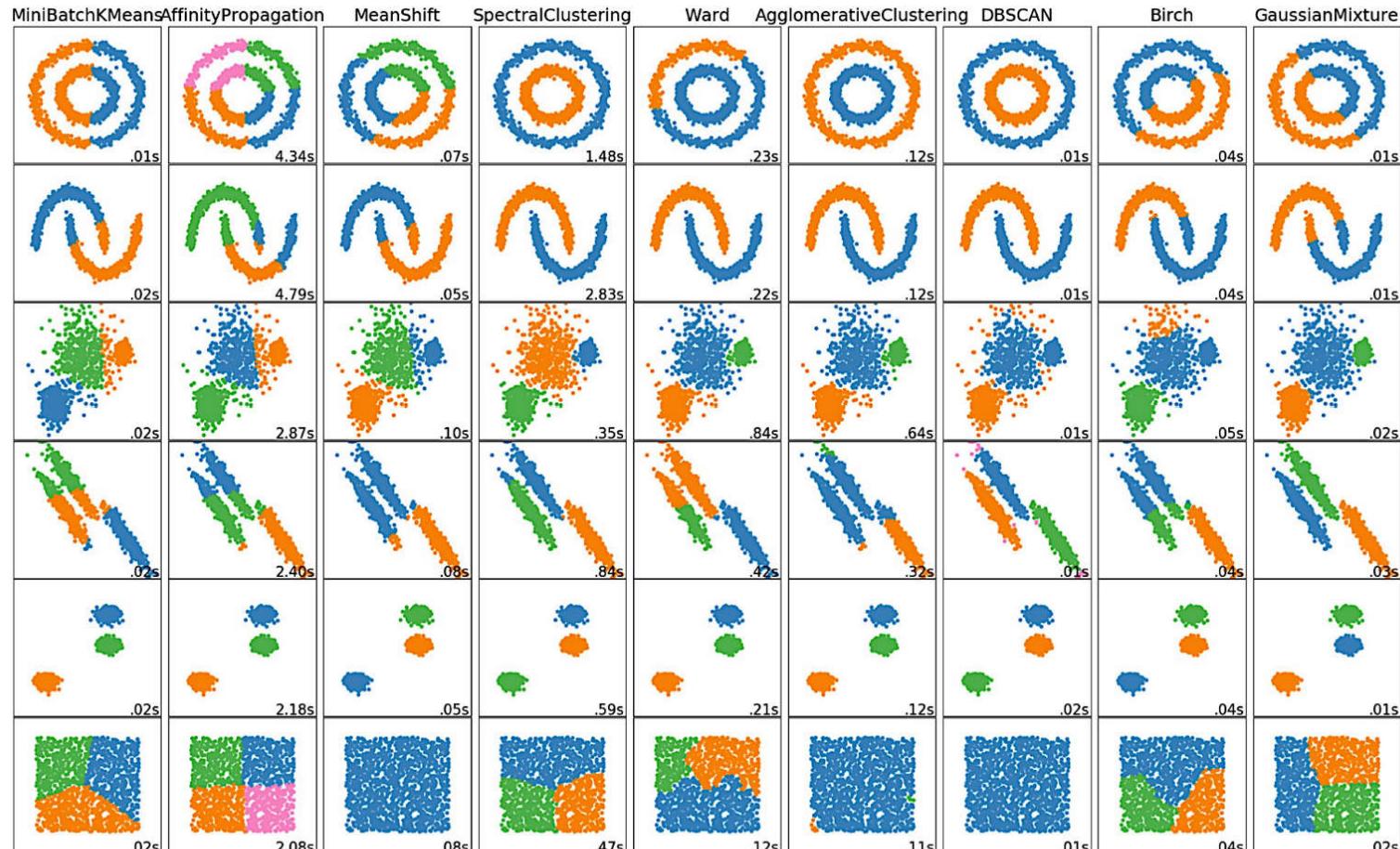
Complete Linkage: This method merges the two clusters that have the closest pair of points based on the maximum distance between any two points from different clusters.

Non-traditional Hierarchical Clustering:

Non-traditional Hierarchical Clustering approaches focus on more specific clustering techniques that may deviate from the traditional agglomerative or divisive methods. These approaches often involve different criteria or algorithms designed to capture specific characteristics of the data or overcome limitations of traditional methods. Examples of non-traditional hierarchical clustering include Density-based Spatial Clustering of Applications with Noise (DBSCAN) and OPTICS (Ordering Points to Identify Clustering Structure).

DBSCAN: DBSCAN is a density-based clustering algorithm that groups together data points that are close to each other and have a sufficient number of nearby neighbors. It defines clusters based on dense regions separated by sparser areas of the data.

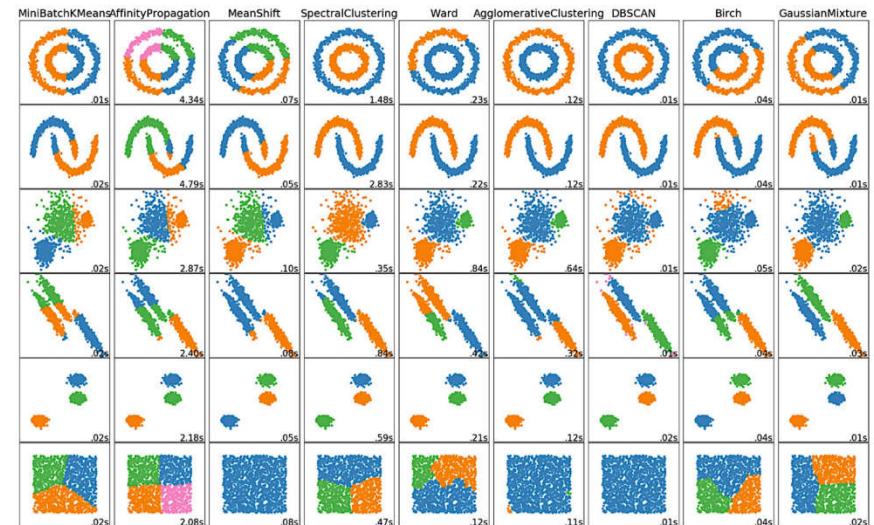
Clustering Toy Data



http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

Clustering Toy Data

- These are toy 2D datasets.
- The last dataset is an example of a ‘null’ situation for clustering.
- With the exception of the last dataset, the parameters of each of these dataset-algorithm pairs has been tuned to produce good clustering results.
- Note that the intuitions might not apply to very high dimensional data.



http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

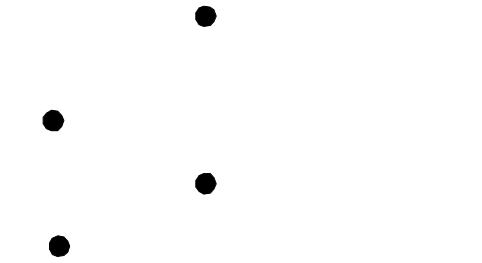
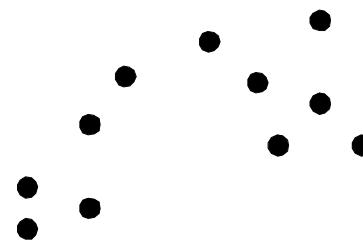
Types of Clusterings

- A **clustering** is a **set of clusters**
- Important distinction between **hierarchical** and **partitional** sets of clusters
 - **Partitional Clustering**
 - ◆ A **division** of data objects into **non-overlapping subsets** (**clusters**)
 - **Hierarchical clustering**
 - ◆ A set of **nested clusters** organized as a **hierarchical tree**

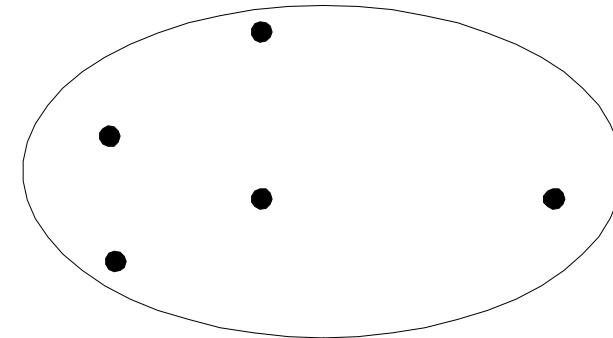
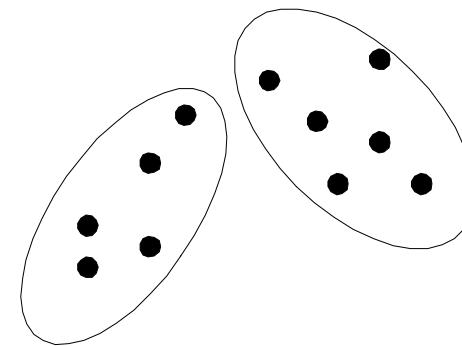
۱. تکه تکه کردن یا پارتیشن
کردن کلاسترها
۲. کلاسترینگ سلسله مراتبی

یه خوشی میتواند عضو یه خوشی
بزرگتر باشه

Partitional Clustering

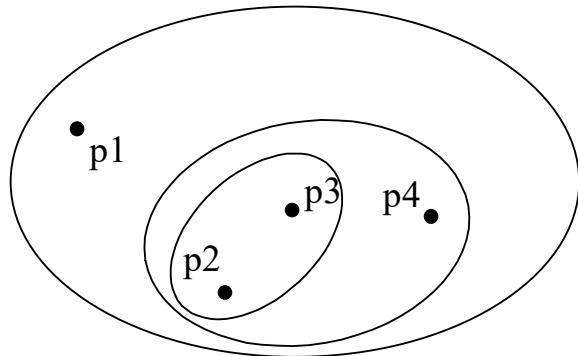


Original Points

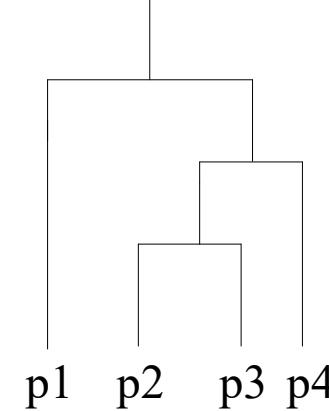


A Partitional Clustering

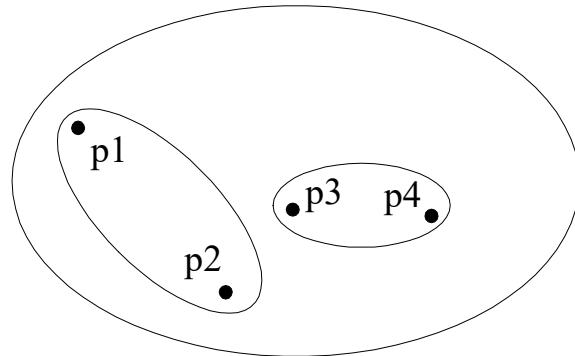
Hierarchical Clustering



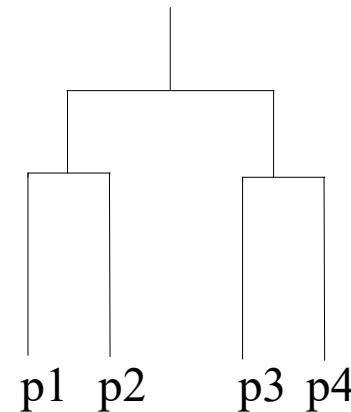
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Other Distinctions Between Sets of Clusters

● Exclusive versus non-exclusive

- In non-exclusive clusterings, points may belong to multiple clusters.
 - ◆ Can belong to multiple classes or could be ‘border’ points

— Fuzzy clustering (one type of non-exclusive)

- ◆ In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
- ◆ Weights must sum to 1
- ◆ Probabilistic clustering has similar characteristics

● Partial versus complete

- In some cases, we only want to cluster some of the data

این نمونه ای که داریم بررسی میکنیم قطعاً را یک خوش است یا نه؟ نسبی است؟ یه نمونه ای با یک حتمالی متعلق به یک خوش است

تکنیک خوش بندی روی کل داده ها اعمال میشه یا روی یه بخشی از داده ها؟

Types of Clusters

انواع خوشه ها
خوشه ها در فضا چگونه نسبت
به هم قرار میگیرند؟

- Well-separated clusters
- Prototype-based clusters
- Contiguity-based clusters
- Density-based clusters
- Described by an Objective Function

خوشه ها کاملا از هم جدا هستند یعنی داده ها به
گونه ای هستند که همه ای رکوردهای یک خوشه
از یک خوشه دیگر فاصله دارند

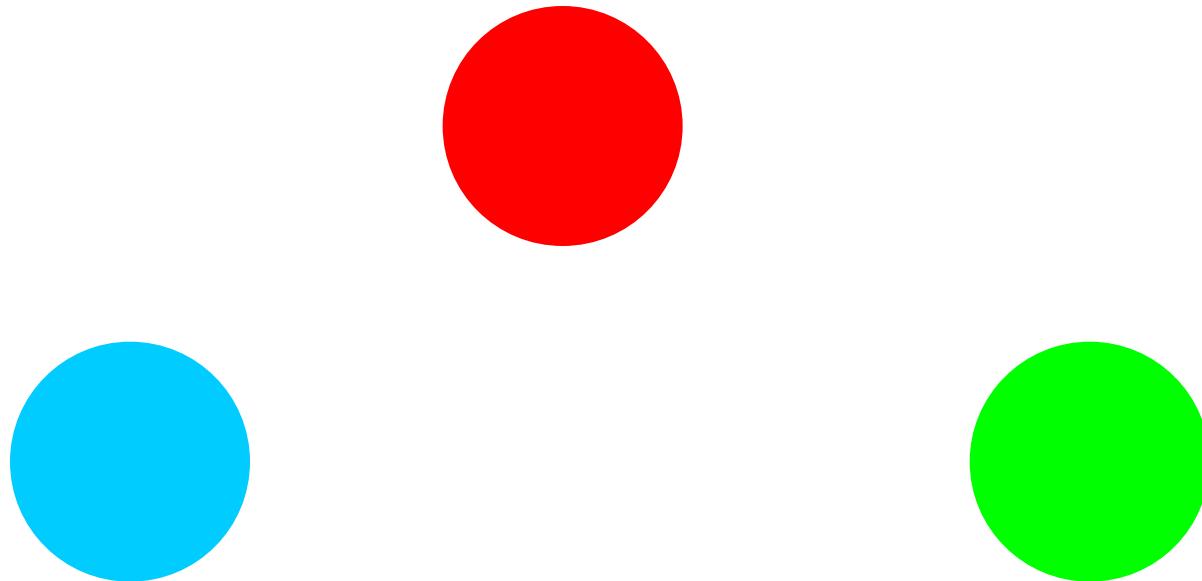
خوشه ها نسبت به یک نمونه ای اولیه از هم
گستته شدند.

خودمون تعریف کنیم
که یک خوشه چگونه
شکل میگیره؟

Types of Clusters: Well-Separated

- Well-Separated Clusters:

- A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



3 well-separated clusters

Types of Clusters: Prototype-Based

فاصله نقاط نسبت به یک نمونه
مثل مرکز متفاوت میشه

- Prototype-based

- A cluster is a set of objects such that an object in a cluster is closer (more similar) to the prototype or “center” of a cluster, than to the center of any other cluster
- The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most “representative” point of a cluster



4 center-based clusters

Types of Clusters: Contiguity-Based

پیوسته

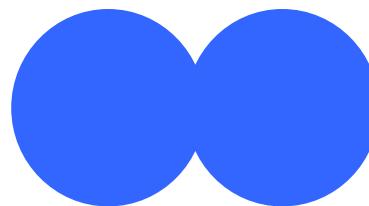
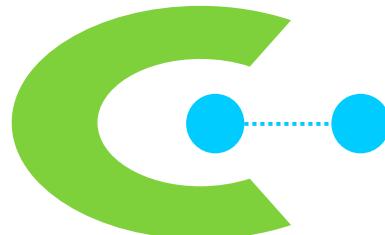
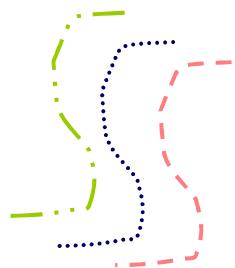
- Contiguous Cluster

(Nearest neighbor or Transitive)

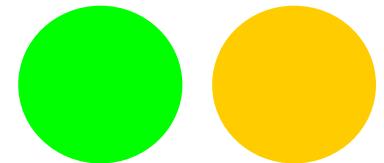
متعدد

یک درجه ای از نزدیک بودن
و مجاورت را تعریف میکنیم
نمونه های مجاور هم توی یک
خوش قرار میگیرند.

- A cluster is a set of points such that a point in a cluster is closer (or more similar) **to one or more other points in the cluster than to any point not in the cluster.**
- Used when the clusters are **irregular** or **intertwined**(non Noise)



در هم تنیده شده است



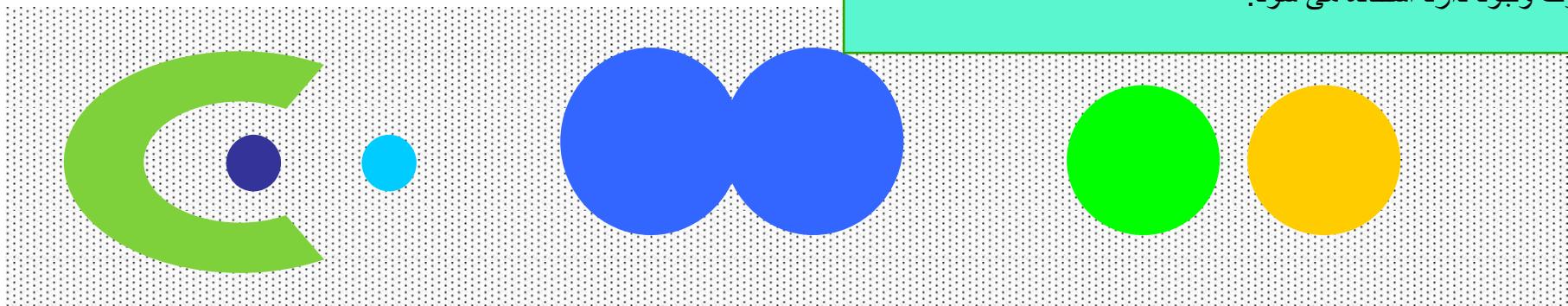
8 contiguous clusters

این تکنیک ها خیلی به نویز حساس هستند نویز مثل اینکه یه سری داده های تصادفی بریزیم توی داده های اصلی پس برای داده هایی که نویز دارند از این روش ها استفاده نمیکنیم.

Types of Clusters: Density-Based

● Density-based

- A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
- Used when the clusters are irregular or intertwined, and when noise and outliers are present.



6 density-based clusters

Introduction to Data Mining, 2nd Edition

برای داده های نویزی خیلی
خوبه
نگاه میکنیم به چگالی داده ها

Clustering Algorithms

- K-means and its variants

از الگوریتم های
پارتیشن بندی
عنی فضای پارتیشن
بندی میکنند

- Hierarchical clustering

شکل کروی

- Density-based clustering

Method	General Characteristics
Partitioning methods	<ul style="list-style-type: none">– Find mutually exclusive clusters of spherical shape– Distance-based– May use mean or medoid (etc.) to represent cluster center– Effective for small- to medium-size data sets
Hierarchical methods	<ul style="list-style-type: none">– Clustering is a hierarchical decomposition (i.e., multiple levels)– Cannot correct erroneous merges or splits– May incorporate other techniques like microclustering or consider object “linkages”
Density-based methods	<ul style="list-style-type: none">– Can find arbitrarily shaped clusters– Clusters are dense regions of objects in space that are separated by low-density regions– Cluster density: Each point must have a minimum number of points within its “neighborhood”– May filter out outliers
Grid-based methods	<ul style="list-style-type: none">– Use a multiresolution grid data structure– Fast processing time (typically independent of the number of data objects, yet dependent on grid size)

Algorithm: k -means. The k -means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- k : the number of clusters,
- D : a data set containing n objects.

Output: A set of k clusters.

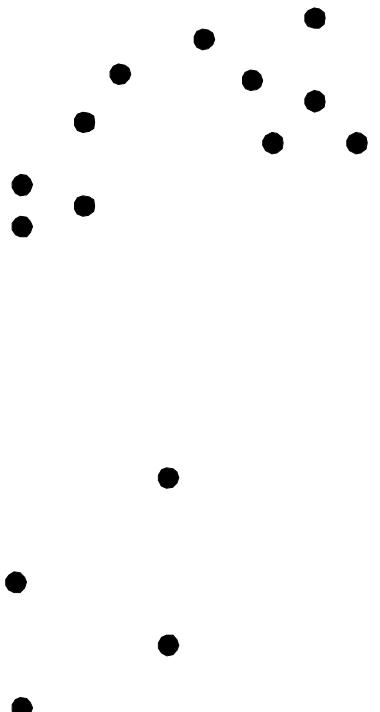
Method:

- (1) arbitrarily choose k objects from D as the initial cluster centers;
- (2) repeat
- (3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
- (4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
- (5) until no change;

K-MEANS CLUSTERING

Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database D of n objects into a set of k clusters,



The centroid of a cluster is simply the mean (average) value of all the data points in that cluster. To calculate the centroid, follow these steps:

Calculate the mean value of the x-coordinates and y-coordinates separately for all the data points in the cluster.

Use the resulting mean values as the x-coordinate and y-coordinate of the centroid.
Here's an example to illustrate how to calculate the centroid of a cluster with three data points:

Data Points:

(2, 4), (3, 6), (5, 8)

Calculate the mean value of the x-coordinates:

Mean(x) = $(2 + 3 + 5) / 3 = 10 / 3 = 3.33$ (rounded to two decimal places)

Calculate the mean value of the y-coordinates:

Mean(y) = $(4 + 6 + 8) / 3 = 18 / 3 = 6$

Use the resulting mean values as the x-coordinate and y-coordinate of the centroid:
Centroid = (3.33, 6)

So the centroid of this cluster is (3.33, 6).

Partitioning Algorithms: Basic Concept

هدف : بهینه کردن یک تابع

- Partitioning method: Partitioning a database D of n objects into a set of k clusters,
such that the sum of squared distances is minimized (where c_i is the centroid or medoid of cluster C_i)

تابعی تعریف میکنیم که فاصله از میانگین را تعریف میکنه
تابع را طوری تعریف میکنیم که زمانی که خوش بندی خوب انجام بشه مقداری که تابع میده کمینه میشه

$$E = \sum_{i=1}^k \sum_{p \in C_i} (p - c_i)^2$$

نمونه هایی که توی هر خوش هستند چقدر مرکز هاشون نزدیک هستند؟

Partitioning Algorithms: Basic Concept

- Given k , find a partition of k clusters that optimizes the chosen partitioning criterion
 - Global optimal: exhaustively enumerate all partitions
 - Heuristic methods: k -means and k -medoids algorithms
 - k -means (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
 - k -medoids or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster


$$E = \sum_{i=1}^k \sum_{p \in C_i} (p - c_i)^2$$

The *K-Means* Clustering Method

- Given k , the *k-means* algorithm is implemented in four steps:
 - Partition objects into k nonempty subsets
 - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
 - Assign each object to the cluster with the nearest seed point
 - Go back to Step 2, stop when the assignment does not change

در ابتدا به صورت تصادفی n تا ابجکت را به k تا خوشه تقسیم بندی میکنیم.
بعدش تلوی این کاتا مجموعه میایم میانگین هر خوشه را حساب میکنیم.
حالا تک تک ابجکت هامون را براساس فاصله از این میانگینی که بدست اوردهیم مجدداً به خوشه ها اساین میکنیم (خوشه ها را اپدیت میکنیم)

K-Means Clustering is a popular unsupervised machine learning algorithm used to cluster/group similar data points together. The "K" in K-Means refers to the number of clusters that we want to identify from the given dataset.

Here's how it works:

1. First, we randomly select K data points as centroids (center points) for each cluster.
2. Then, we calculate the distance between each data point and the centroids using a distance metric like Euclidean distance.
3. Next, we assign each data point to the nearest centroid, forming K clusters.
4. After that, we re-calculate the centroids for each cluster by taking the mean of all the data points in the cluster.
5. We repeat steps 2-4 until the centroids no longer move or the maximum number of iterations is reached.

Once the algorithm converges, we will have K clusters containing similar data points.

Here's an example: Let's assume we have a dataset of customer information containing age and income. We want to cluster these customers into two groups based on their age and income. We choose K = 2 and randomly select two points as centroids. Then, we calculate the distance between each data point and the centroids, assign each data point to the nearest centroid, and re-calculate the centroids for each cluster. We repeat this process until the centroids no longer move. Finally, we have two clusters consisting of customers with similar age and income.

Sure, let's say we have a dataset of 8 data points and we want to cluster them into two groups based on their values. Here's the dataset:

[2, 3, 6, 7, 8, 10, 13, 14]

We choose K = 2 and randomly select two points as centroids: 5 and 11. Then, we calculate the distance between each data point and the centroids using Euclidean distance and assign each data point to the nearest centroid. After that, we re-calculate the centroids for each cluster by taking the mean of all the data points in the cluster. We repeat this process until the centroids no longer move or the maximum number of iterations is reached.

Here's how the clustering process might look like:

Iteration 1:

Cluster 1: [2, 3, 6, 7, 8] Centroid: 5

Cluster 2: [10, 13, 14] Centroid: 11.67

SSE: $(2-5)^2 + (3-5)^2 + (6-5)^2 + (7-5)^2 + (8-5)^2 + (10-11.67)^2 + (13-11.67)^2 + (14-11.67)^2 = 66.33$

New centroids: Cluster 1 mean = 5.2, Cluster 2 mean = 12.33

Iteration 2:

Cluster 1: [2, 3, 6, 7, 8] Centroid: 5.2

Cluster 2: [10, 13, 14] Centroid: 12.33

SSE: $(2-5.2)^2 + (3-5.2)^2 + (6-5.2)^2 + (7-5.2)^2 + (8-5.2)^2 + (10-12.33)^2 + (13-12.33)^2 + (14-12.33)^2 = 36.47$

New centroids: Cluster 1 mean = 4.8, Cluster 2 mean = 12.33

Iteration 3:

Cluster 1: [2, 3, 6, 7, 8] Centroid: 5.2

Cluster 2: [10, 13, 14] Centroid: 12.33

SSE: $(2-4.8)^2 + (3-4.8)^2 + (6-4.8)^2 + (7-4.8)^2 + (8-4.8)^2 + (10-12.33)^2 + (13-12.33)^2 + (14-12.33)^2 = 30.53$

New centroids: Cluster 1 mean = 5.2, Cluster 2 mean = 12.33

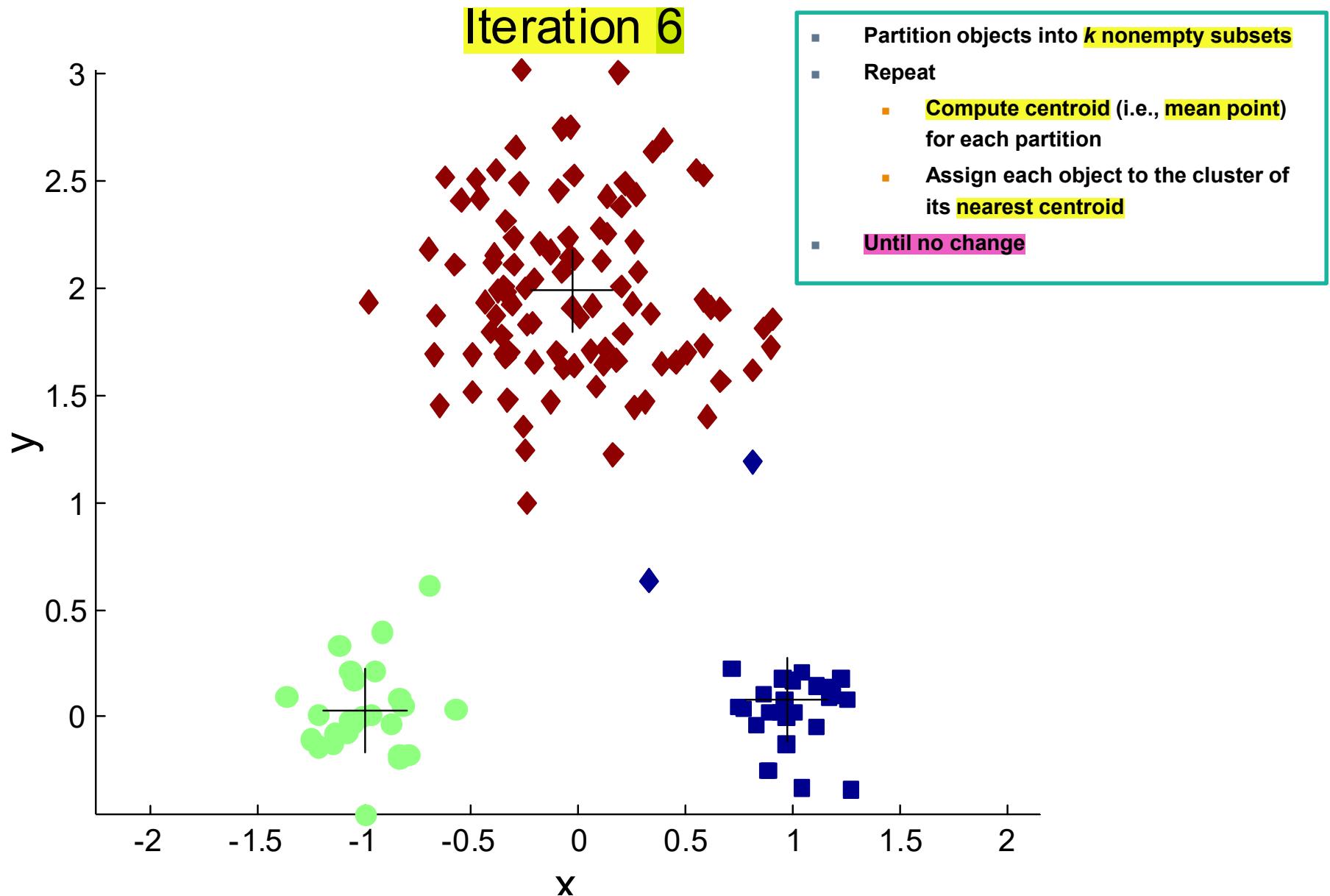
Since the centroids no longer move after the third iteration, we stop and conclude that the optimal clustering result for this dataset is:

Cluster 1: [2, 3, 6, 7, 8]

Cluster 2: [10, 13, 14]

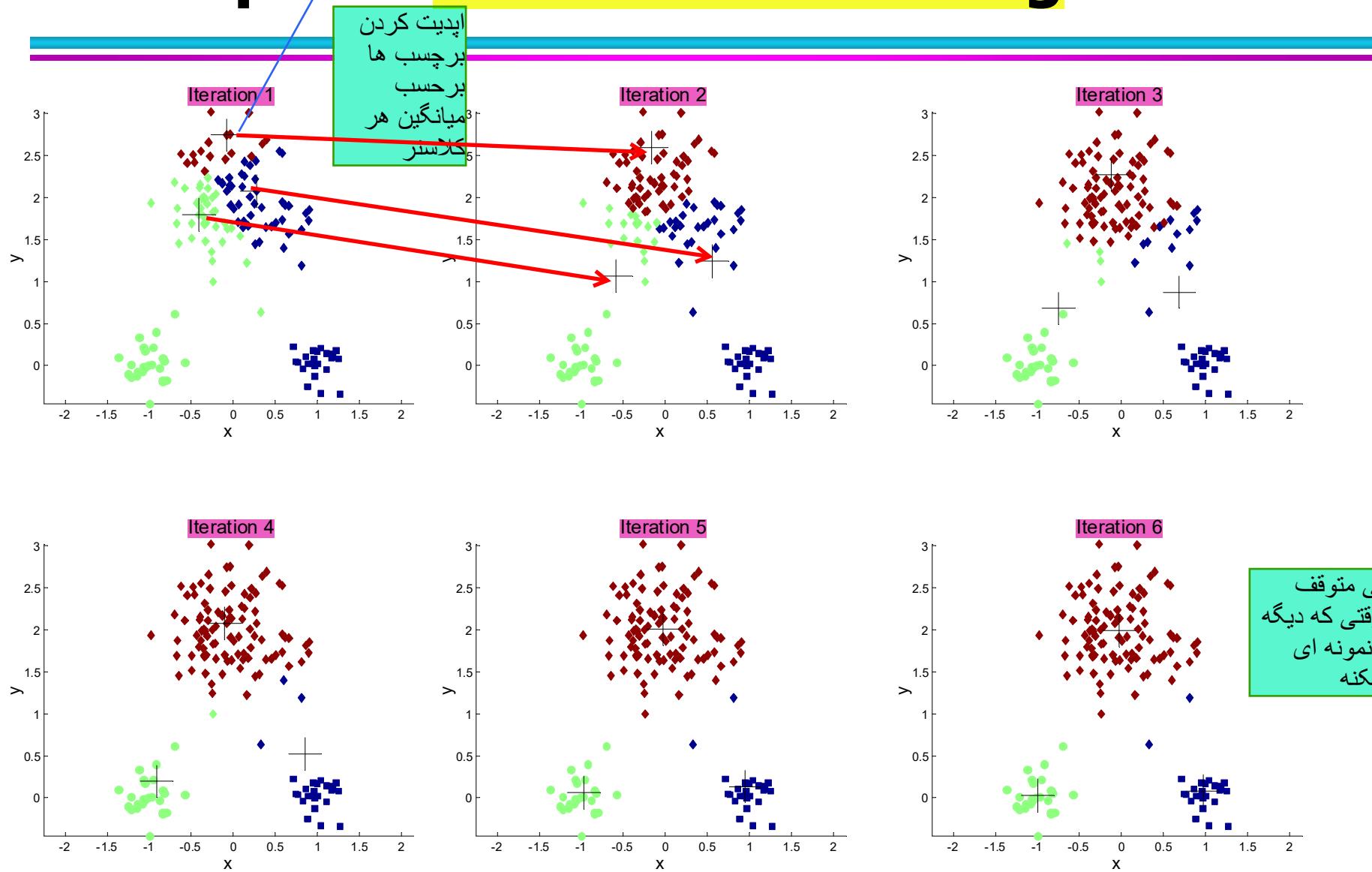
with a Sum of Squared Error (SSE) of 30.53.

Example of K-means Clustering



Example of K-means Clustering

centroid of cluster



K-means Clustering – Details

- Simple iterative algorithm.
 - Choose initial centroids;
 - repeat {assign each point to a nearest centroid; re-compute cluster centroids}
 - until centroids stop changing.
- Initial centroids are often chosen randomly.
 - Clusters produced can vary from one run to another
- The centroid is (typically) the mean of the points in the cluster, but other definitions are possible (see Table 5.2).

K-means Clustering – Details

- K-means will converge for common proximity measures with appropriately defined centroid (see Table 5.2)
- Most of the convergence happens in the first few iterations.
 - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is $O(n * K * I * d)$
 - n = number of points, K = number of clusters,
I = number of iterations, d = number of attributes

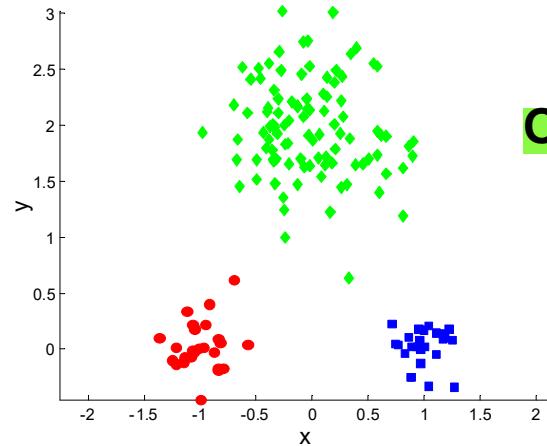
K-means Objective Function

- A common objective function (used with Euclidean distance measure) is Sum of Squared Error (SSE)
 - For each point, the error is the distance to the nearest cluster center
 - To get SSE, we square these errors and sum them.

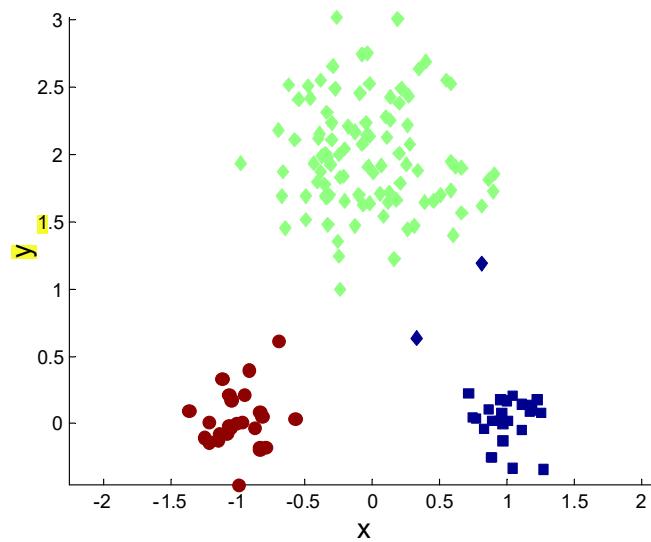
$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

- x is a data point in cluster C_i and m_i is the centroid (mean) for cluster C_i
- SSE improves in each iteration of K-means until it reaches a local or global minima.

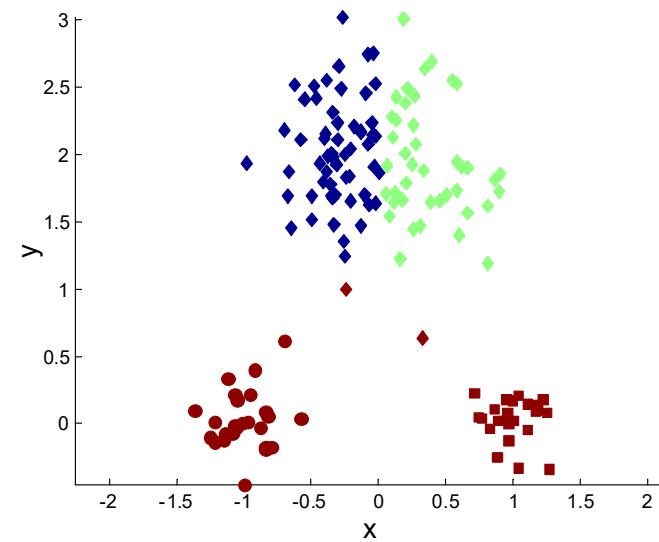
Two different K-means Clusterings



Original Points

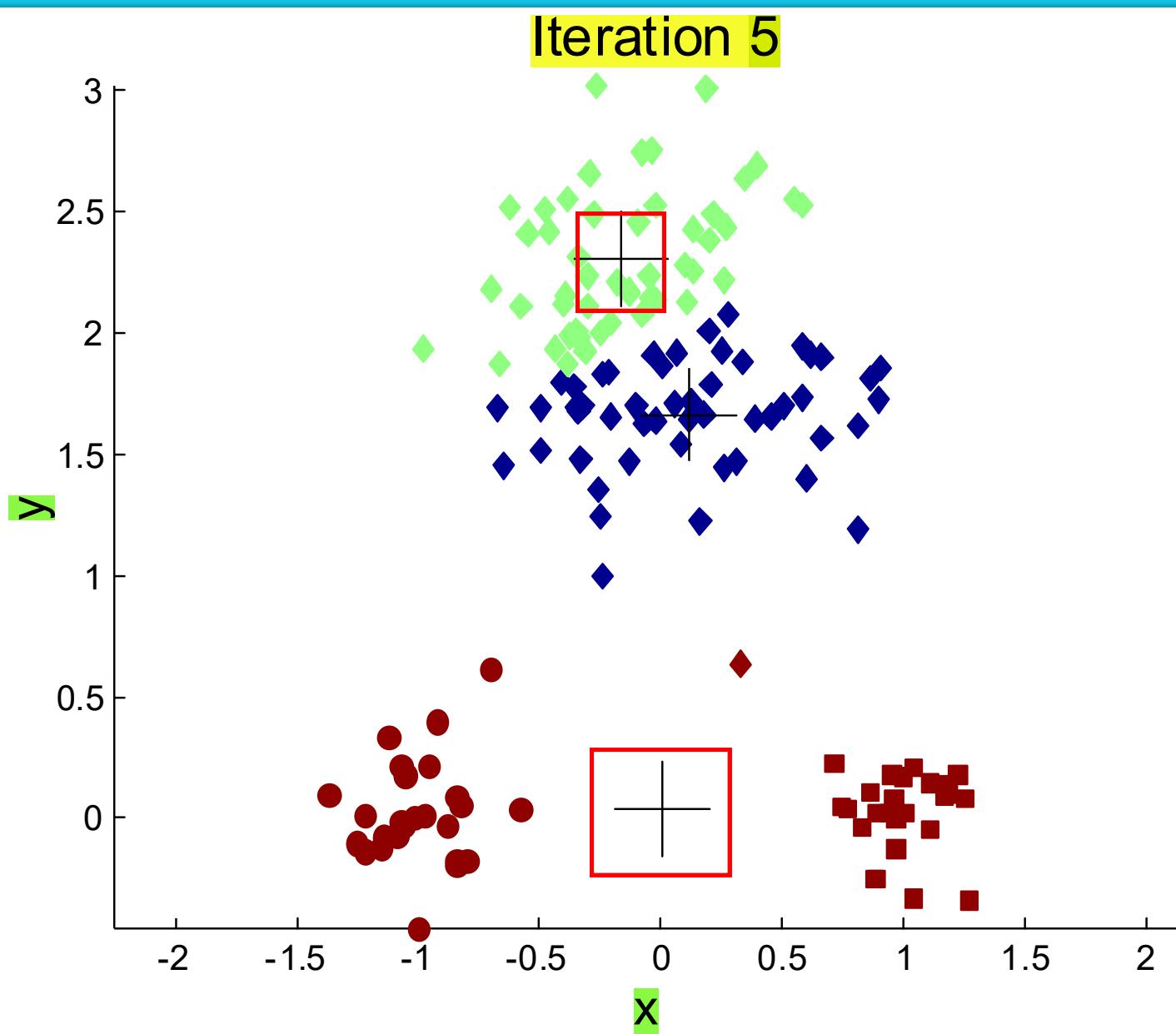


Optimal Clustering

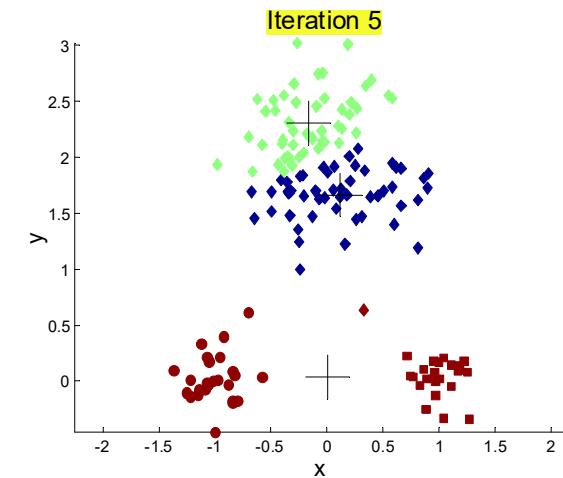
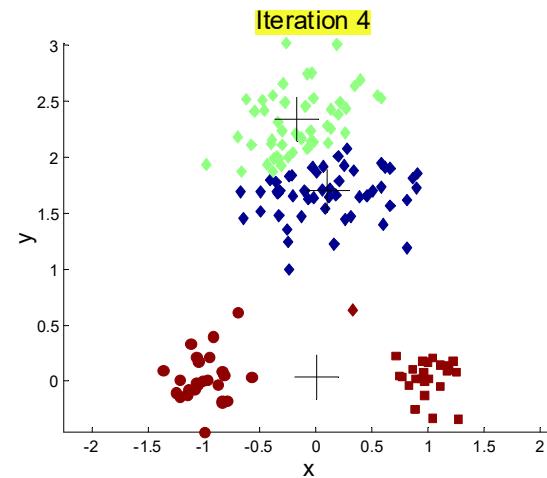
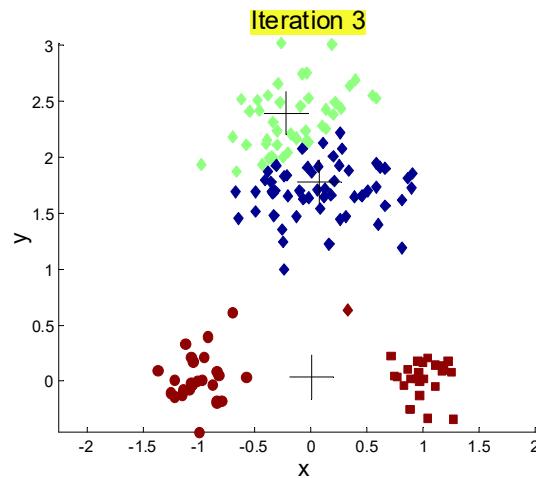
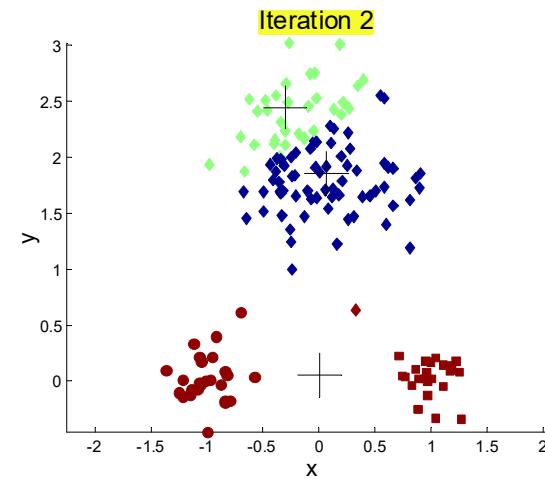
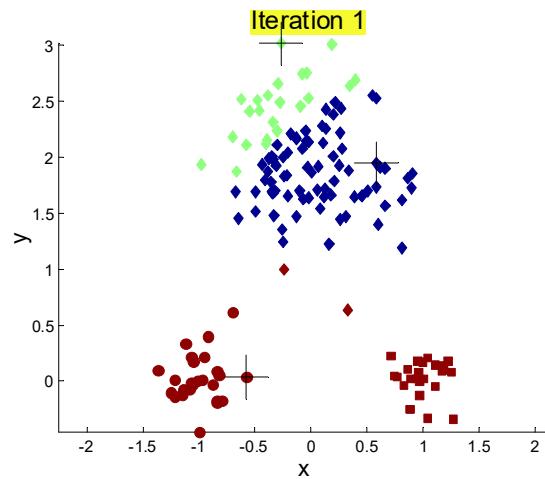


Sub-optimal Clustering

Importance of Choosing Initial Centroids ...

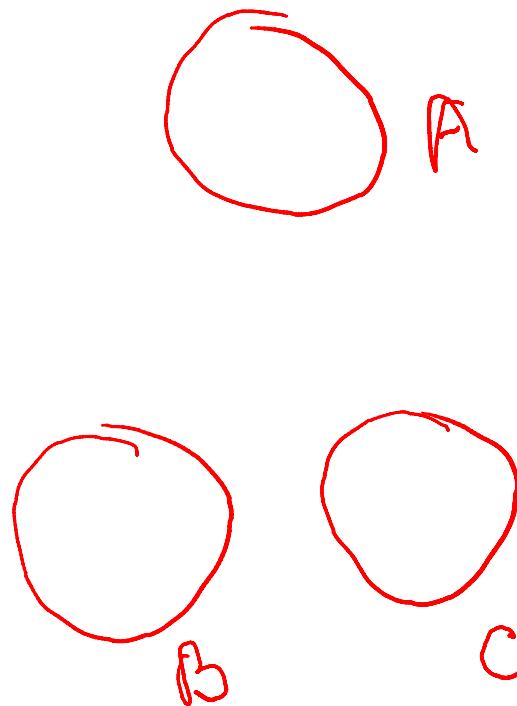


Importance of Choosing Initial Centroids ...

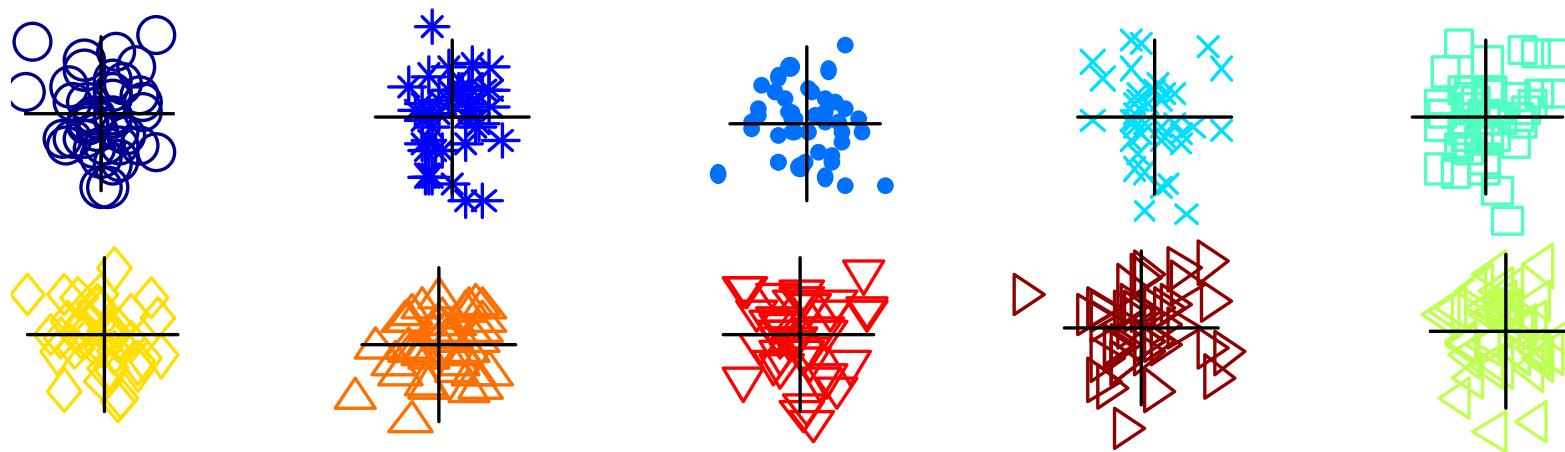


Importance of Choosing Initial Centroids

- Depending on the choice of initial centroids, B and C may get merged or remain separate

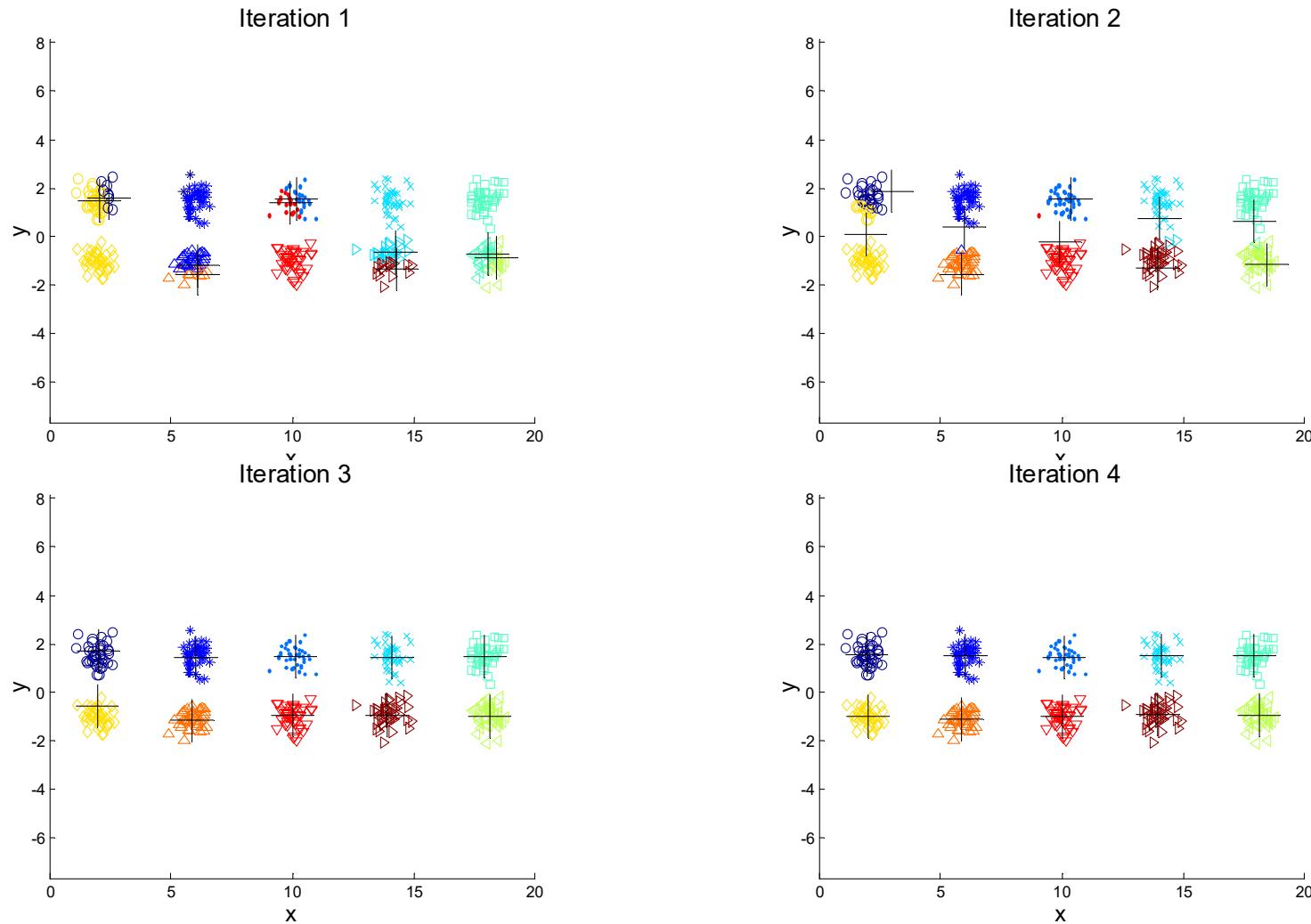


10 Clusters Example



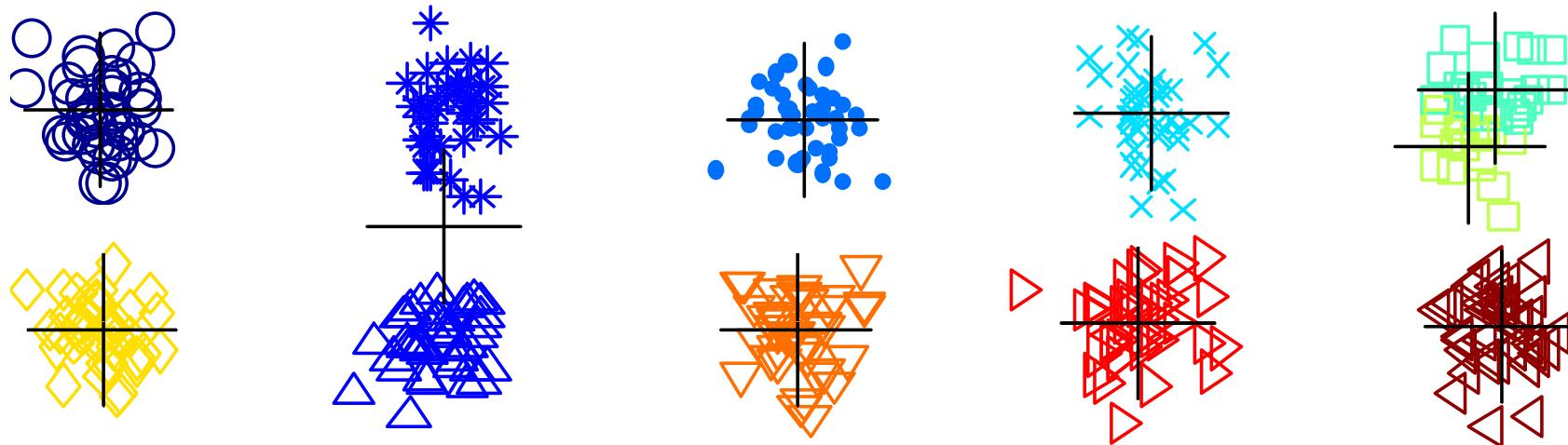
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



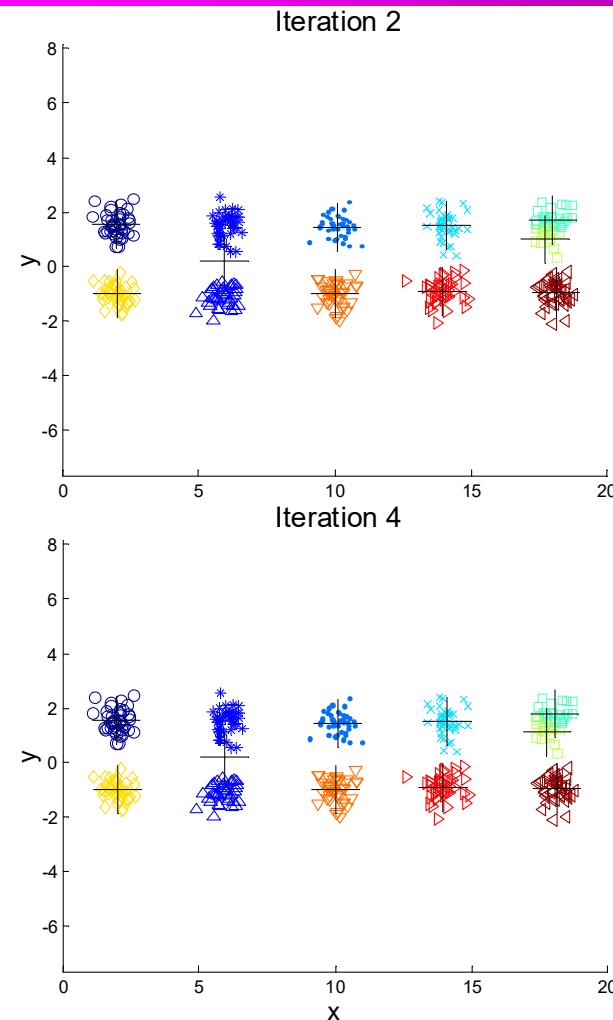
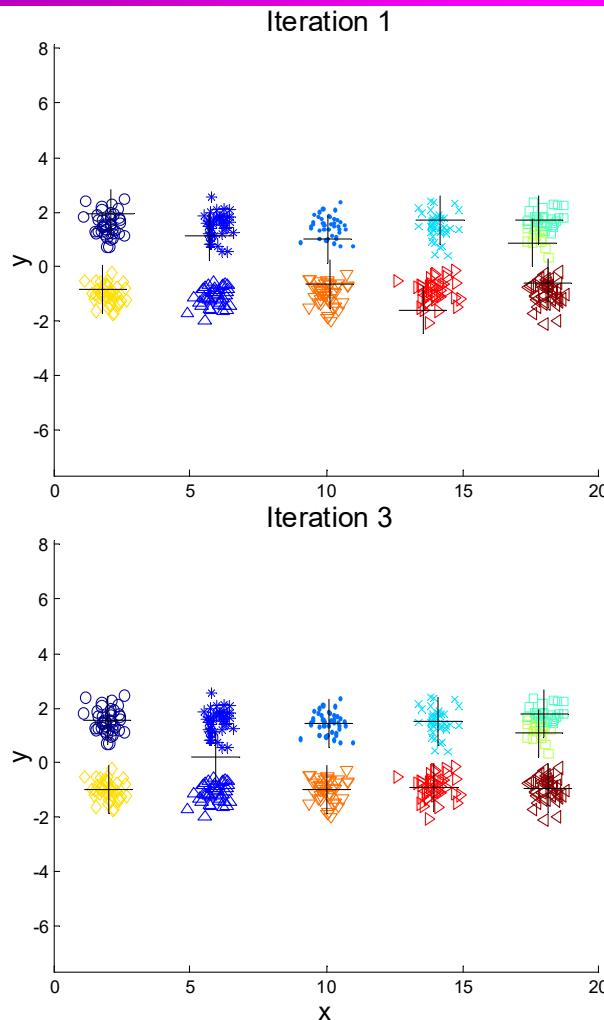
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



Starting with **some pairs of clusters** having **three initial centroids**, while other have only one.

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

Solutions to Initial Centroids Problem

- Multiple runs
 - Helps, but probability is not on your side
- Use some strategy to select the k initial centroids and then select among these initial centroids
 - Select most widely separated
 - ◆ K-means++ is a robust way of doing this selection
 - Use hierarchical clustering to determine initial centroids
- Bisecting K-means
 - Not as susceptible to initialization issues

دو نیم کردن K
به مسائل اولیه سازی حساس نیست

K-means++

- This approach can be slower than random initialization, but very consistently produces better results in terms of SSE
 - The k-means++ algorithm guarantees an approximation ratio $O(\log k)$ in expectation, where k is the number of centers
- To select a set of initial centroids, C , perform the following

Algorithm 5.2 K-means++ initialization algorithm.

- 1: For the first centroid, pick one of the points at random.
- 2: **for** $i = 1$ to *number of trials* **do**
- 3: Compute the distance, $d(x)$, of each point to its closest centroid.
- 4: Assign each point a probability proportional to each point's $d(x)^2$.
- 5: Pick new centroid from the remaining points using the weighted probabilities.
- 6: **end for**

K-means++ is an initialization algorithm used in the K-means clustering algorithm. It aims to select initial cluster centroids in a way that improves the convergence and quality of the final clustering result. K-means++ ensures that the initial centroids are well-spaced and representative of the underlying data distribution.

Here's a step-by-step explanation of how K-means++ works using a numeric example:

Let's assume we have a dataset with the following points in a two-dimensional space:

```
scss
```

 Copy code

```
(1, 1), (1, 2), (2, 1), (4, 4), (5, 5), (5, 6), (6, 5)
```

Step 1: Select the first centroid randomly from the data points.

```
yaml
```

 Copy code

```
Centroid 1: (1, 2)
```

Step 2: Calculate the squared Euclidean distance between each point and the chosen centroid.

Assign probabilities to the points based on their distance squared.

```
scss
```

 Copy code

Point	Squared Distance	Probability
(1, 1)	1	0.1
(1, 2)	0	0 (chosen centroid)
(2, 1)	1	0.1
(4, 4)	18	0.8
(5, 5)	32	1.0
(5, 6)	34	1.0
(6, 5)	32	1.0

Step 3: Select the next centroid based on the probabilities calculated in the previous step. Higher probabilities imply a higher chance of being chosen.

```
yaml
```

 Copy code

```
Centroid 1: (1, 2)
```

```
Centroid 2: (5, 5)
```

Step 4: Calculate the squared Euclidean distance between each point and the chosen centroids.

Assign probabilities based on their distances.

scss

Copy code

Point	Squared Distance to Centroid 1	Squared Distance to Centroid 2	Probability
(1, 1)	1	18	0.05
(1, 2)	0	8	0.01
(2, 1)	1	18	0.05
(4, 4)	18	0	0
(5, 5)	32	0	0
(5, 6)	34	2	0.01
(6, 5)	32	2	0.01

Step 5: Repeat Steps 3 and 4 until the desired number of centroids is reached. In this case, let's say we want to find three centroids.

yaml

Copy code

```
Centroid 1: (1, 2)
Centroid 2: (5, 5)
Centroid 3: (4, 4)
```

After the initialization step, the K-means algorithm can proceed with the standard K-means clustering process, where points are assigned to the nearest centroid, and centroids are updated iteratively until convergence.

K-means++ ensures that the initial centroids are well-distributed and representative, leading to improved clustering results compared to randomly choosing initial centroids. By considering the distances and probabilities, it reduces the chance of initialization leading to suboptimal clustering solutions.

Limitations of K-means

- K-means has **problems** when **clusters** are of differing

- **Sizes**
 - **Densities**
 - **Non-globular shapes**

حدودیت های K-means زمانی که خوشه ها متفاوت هستند مشکل دارد

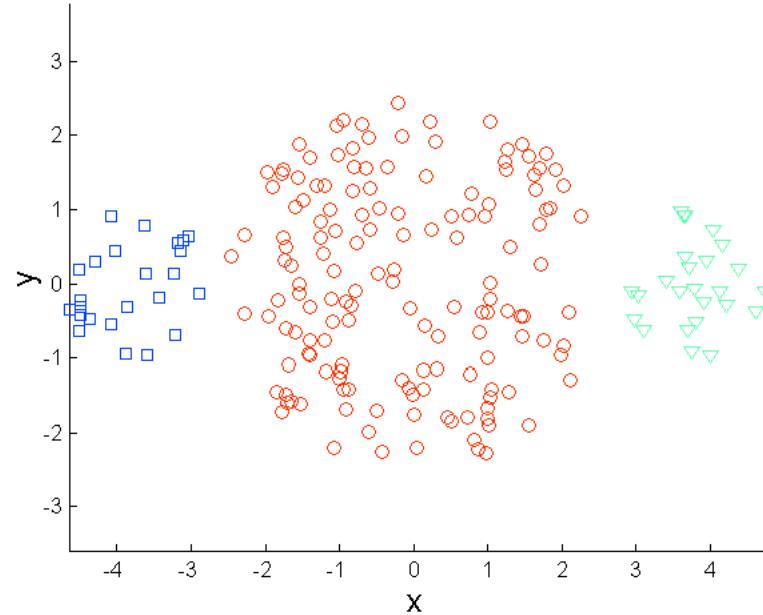
- اندازه ها
- تراکم ها
- اشکال غیر کروی

هنگامی که داده ها حاوی مقادیر پرت باشد، K-means مشکل دارد.

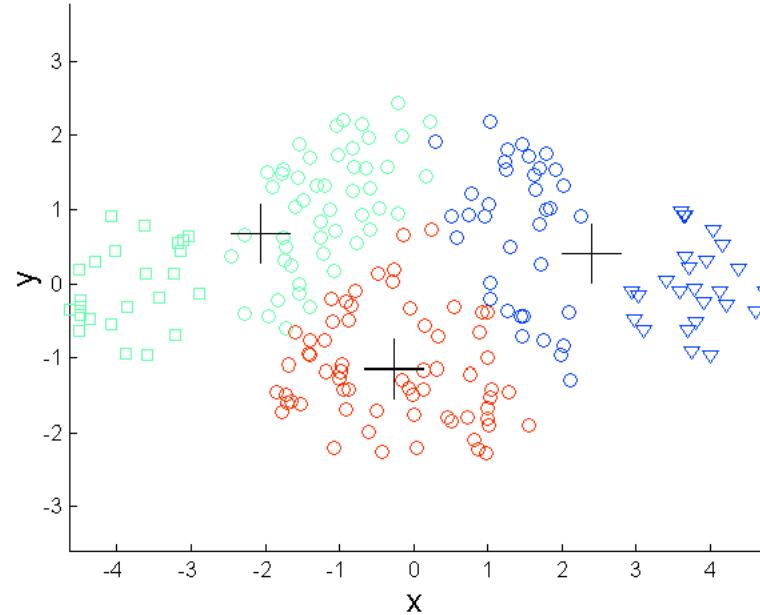
یک راه حل ممکن حذف نقاط پرت قبل از خوشه بندی است

- K-means has problems when the data **contains outliers**.
- One possible solution is to **remove outliers before clustering**

Limitations of K-means: Differing Sizes



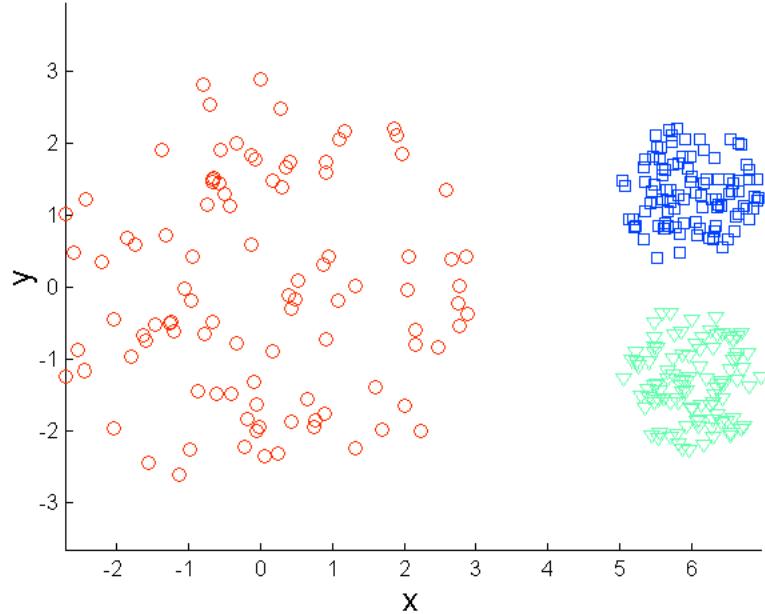
Original Points



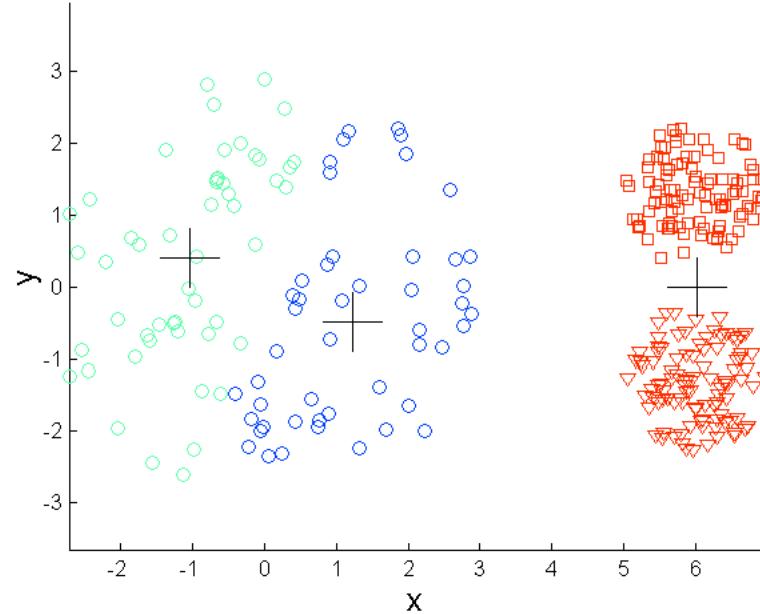
K-means (3 Clusters)

Limitations of K-means: Differing Density

چگالی: تجمع نقاط
حول میانگین

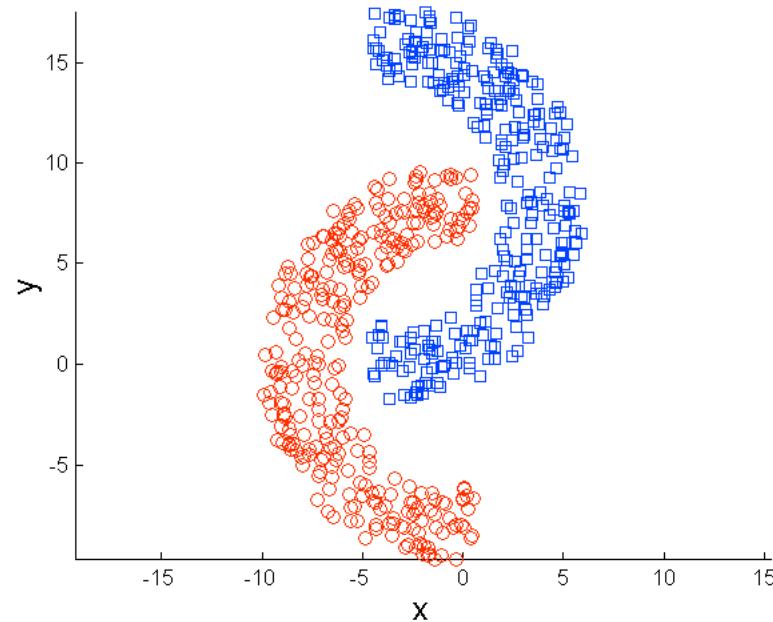


Original Points

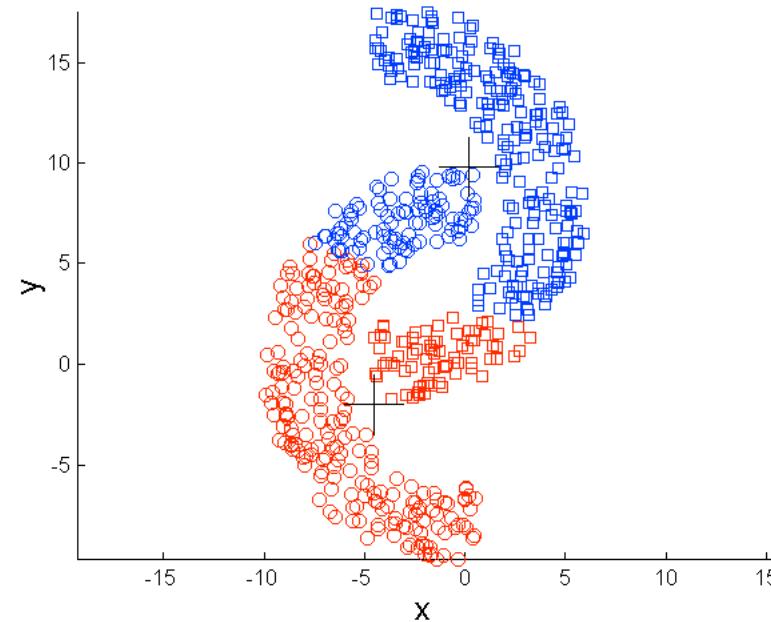


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

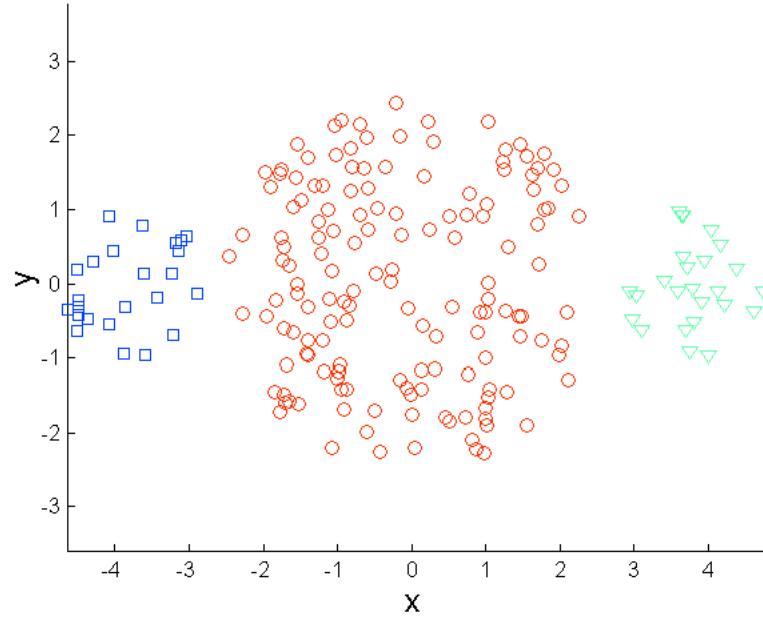


Original Points

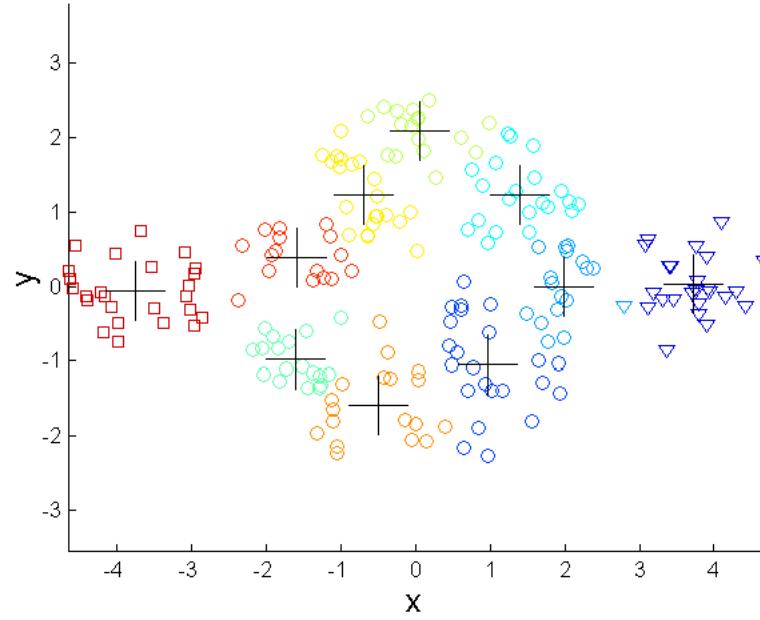


K-means (2 Clusters)

Overcoming K-means Limitations



Original Points

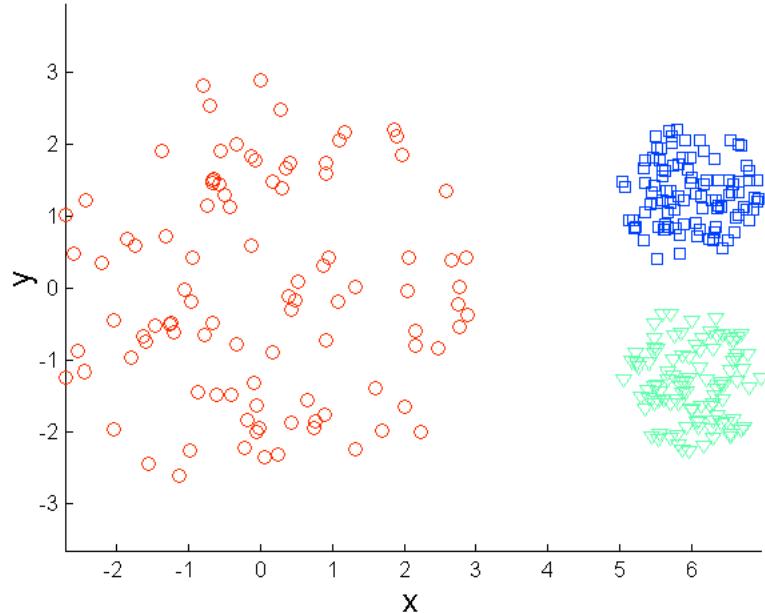


K-means Clusters

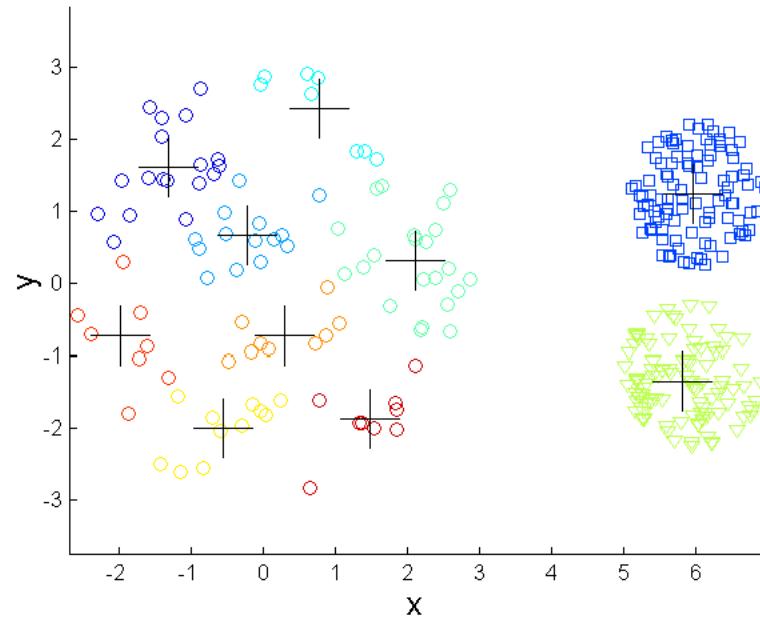
One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a post-processing step.

یک راه حل این است که تعداد زیادی خوشه پیدا کنیم به طوری که هر یک از آنها بخشی از یک خوشه طبیعی را نشان دهد. اما این خوشه های کوچک باید در یک مرحله پس از پردازش کنار هم قرار گیرند.

Overcoming K-means Limitations



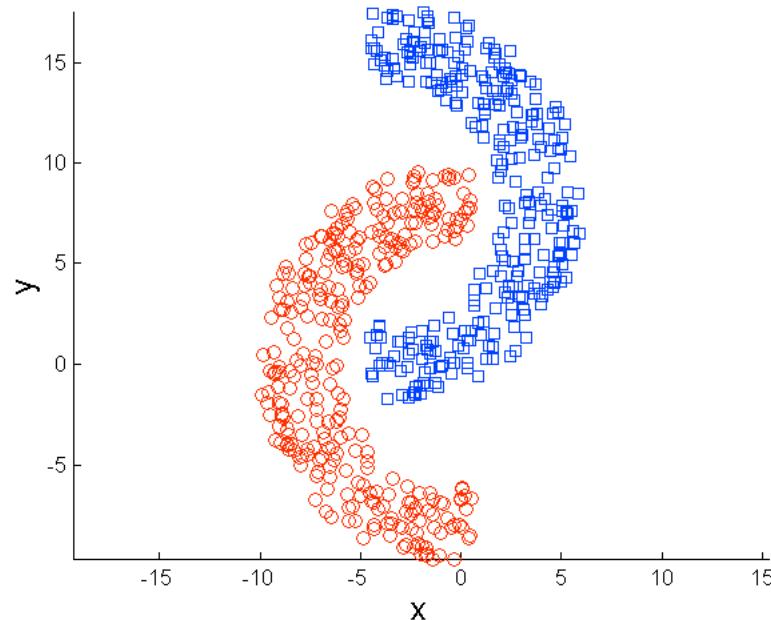
Original Points



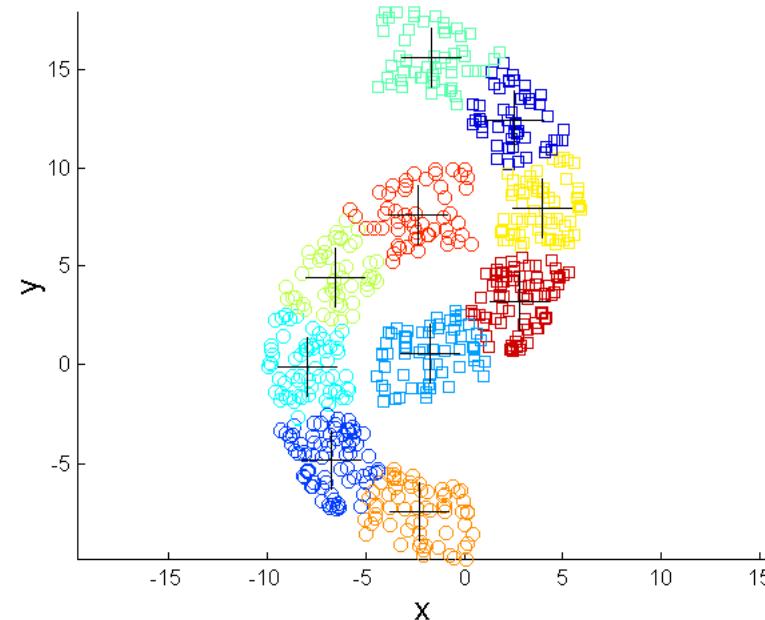
K-means Clusters

One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a **post-processing step**.

Overcoming K-means Limitations



Original Points



K-means Clusters

One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a **post-processing step**.

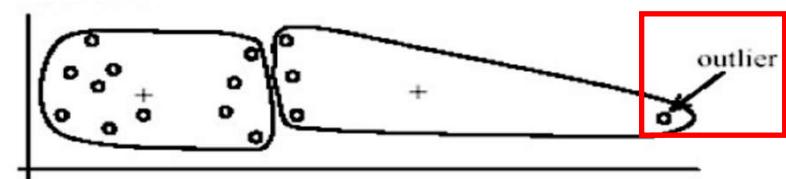
K-means and outlier!

- K-Medoids: Instead of taking the mean value of the object in a cluster as a reference point, medoids can be used, which is the most centrally located object in a cluster

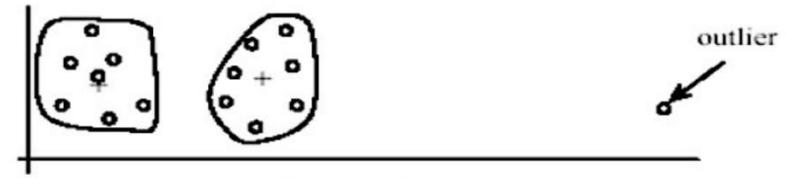
به جای در نظر گرفتن مقدار میانگین شی در یک خوش به عنوان نقطه مرجع، می توان از medoids استفاده کرد که مرکزی ترین شی در یک خوش است.



<https://www.cese.nsw.gov.au/effective-practices/unit-4-outliers>



(A): Undesirable clusters



(B): Ideal clusters

<https://www.slideshare.net/anilyadav5055/15857-cse422-unsupervisedlearning>

K-Medoids is a clustering algorithm that is similar to K-means but instead of using the mean (average) of data points as centroids, it uses actual data points, called medoids, as representatives of clusters. Medoids are the most centrally located points within a cluster, minimizing the dissimilarity or distance to other points within the same cluster.

In K-Medoids clustering, the algorithm aims to minimize the sum of dissimilarities between each data point and its assigned medoid. The dissimilarity or distance measure used can vary, commonly employing metrics such as Euclidean distance, Manhattan distance, or other distance functions based on the data type and problem domain.

The steps involved in the K-Medoids clustering algorithm are as follows:

Initialization: Randomly select K data points from the dataset as initial medoids or use a more advanced initialization method like K-Medoids++.

Assignment: Assign each data point to the closest medoid based on a chosen dissimilarity measure.

Update: For each cluster, compute the total dissimilarity of each data point to all other data points within the cluster. Select the data point with the lowest total dissimilarity as the new medoid for that cluster.

Repeat Steps 2 and 3 until convergence or until a maximum number of iterations is reached.

Finalize: Once the algorithm converges, each data point will be assigned to its corresponding cluster based on the final medoids.

K-Medoids clustering is advantageous when dealing with datasets where the mean may not be a suitable representation, such as categorical data or data with outliers. By using actual data points as medoids, K-Medoids provides more robustness and interpretability in clustering, as the medoids are directly interpretable as representative data points within each cluster.

Certainly! Let's consider a dataset with six data points in a two-dimensional space:

```
scss
```

 Copy code

```
(2, 4), (3, 6), (4, 8), (5, 2), (6, 4), (7, 6)
```

We will use the Manhattan distance (also known as L1 distance) as the dissimilarity measure.

Let's calculate the medoids step-by-step:

Step 1: Initialization: Randomly select K data points from the dataset as initial medoids.

Let's assume we randomly select the first two data points as the initial medoids:

```
yaml
```

 Copy code

```
Medoid 1: (2, 4)  
Medoid 2: (3, 6)
```

Step 2: Assignment: Assign each data point to the closest medoid based on the Manhattan

Calculate the Manhattan distance between each data point and the medoids:

scss

 Copy code

Point	Distance to Medoid 1	Distance to Medoid 2	Assigned Medoid
(2, 4)	0	2	Medoid 1
(3, 6)	2	0	Medoid 2
(4, 8)	4	2	Medoid 2
(5, 2)	5	5	Medoid 2
(6, 4)	6	4	Medoid 2
(7, 6)	8	6	Medoid 2

Step 3: Update: Compute the total dissimilarity of each data point to all other data points within the cluster. Select the data point with the lowest total dissimilarity as the new medoid for that cluster.

For Cluster 1 (assigned to Medoid 1):

java

 Copy code

Point	Total Dissimilarity
(2, 4)	0

For Cluster 2 (assigned to Medoid 2):

scss

 Copy code

Point	Total Dissimilarity
(3, 6)	0
(4, 8)	2
(5, 2)	10
(6, 4)	10
(7, 6)	14

Since the total dissimilarity is lowest for Medoid 1 (Cluster 1), Medoid 1 remains unchanged. However, we need to update Medoid 2 by selecting the data point with the lowest total dissimilarity, which is (3, 6).

yaml

 Copy code

```
Medoid 1: (2, 4)
Medoid 2: (3, 6)
```

Step 4: Repeat Steps 2 and 3 until convergence or until a maximum number of iterations is reached.

In this example, since the medoids remain unchanged after one iteration, the algorithm has converged. The final medoids are:

```
yaml
```

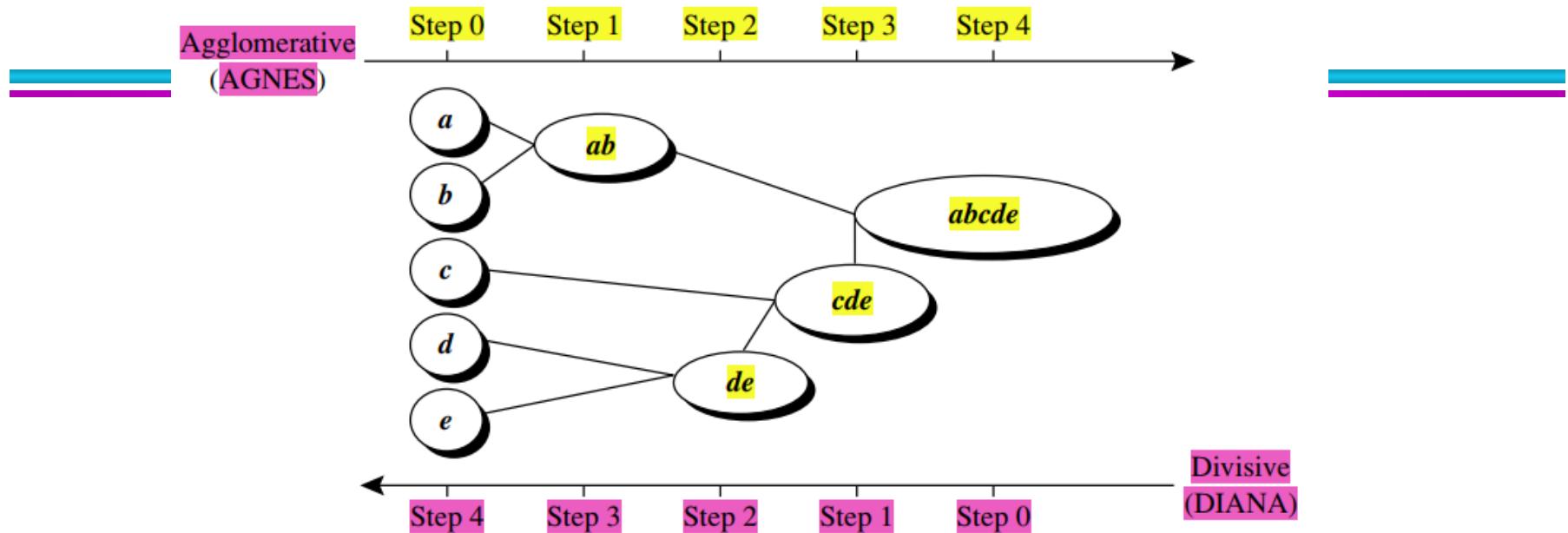
 Copy code

```
Medoid 1: (2, 4)
```

```
Medoid 2: (3, 6)
```

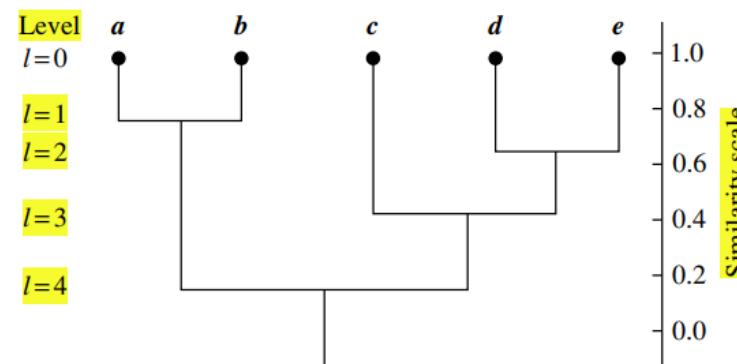
These medoids represent the most centrally located points within their respective clusters based on the Manhattan distance.

Note that in practical scenarios, the K-Medoids algorithm may require multiple iterations before convergence, and the medoids may change in each iteration as the clusters are updated. The example above demonstrates a single iteration for simplicity.



HIERARCHICAL CLUSTERING

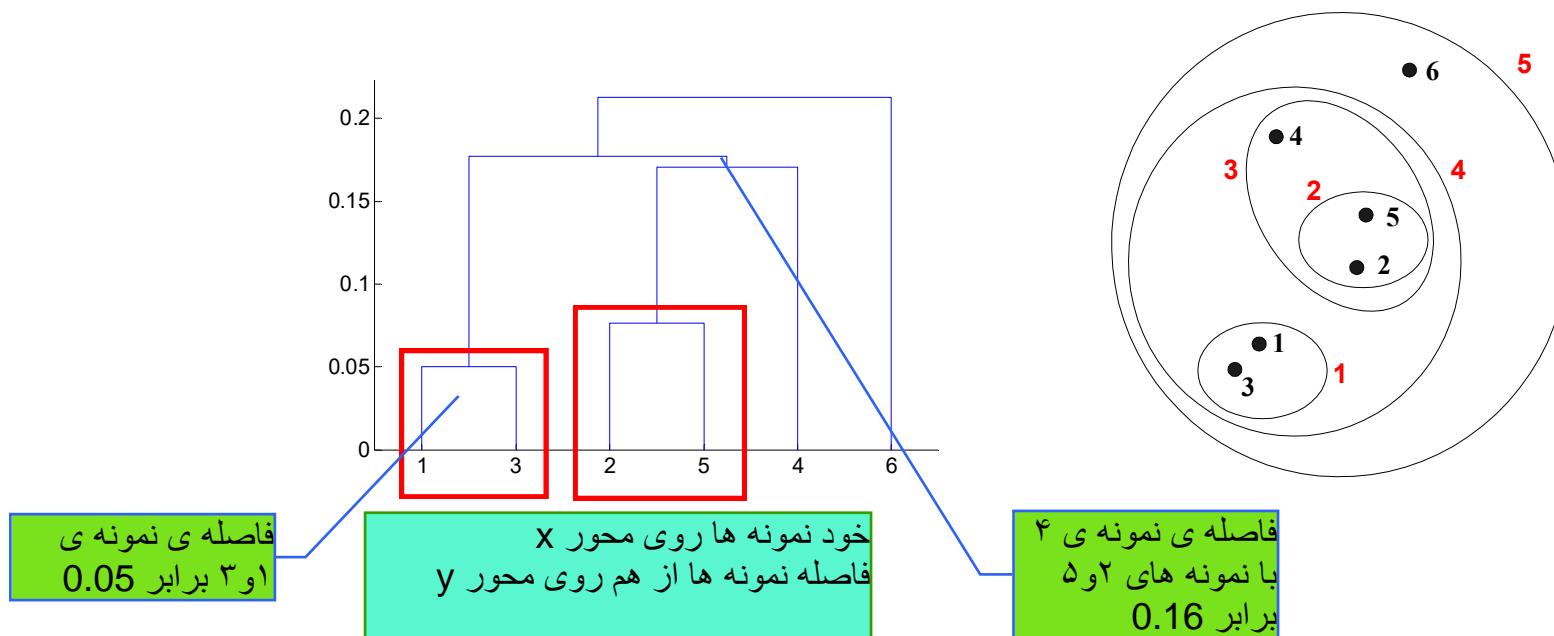
Agglomerative and divisive hierarchical clustering on data objects $\{a, b, c, d, e\}$.



Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
 - A tree like diagram that records the sequences of merges or splits

مجموعه ای از خوشه های تو در تو را که به صورت درخت سلسله مراتبی سازماندهی شده اند تولید می کند. یک نمودار مانند درخت که دنباله های ادغام یا تقسیم را ثبت می کند



Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
 - Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level
- They may correspond to meaningful taxonomies
 - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, ...)

نقاط قوت خوشبندی سلسله مراتبی
مجبور نیستید تعداد خاصی از خوشبندی را فرض کنید.
هر تعداد خوشبندی را می‌توان با "برش" دندروگرام
در سطح مناسب به دست آورد.
آنها ممکن است با طبقه بندی های معنی دار مطابقت
داشته باشند.
مثال در علوم زیستی (مانند پادشاهی حیوانات، بازسازی
فیلوجنی، ...)

۲ تا روش برای ایجاد خوشه ها داریم:
۱. از بالا به پایین
۲. از پایین به بالا

Hierarchical Clustering

- Two main types of hierarchical clustering
 - Agglomerative:
 - ◆ Start with the points as individual clusters
 - ◆ At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
 - Divisive:
 - ◆ Start with one, all-inclusive cluster
 - ◆ At each step, split a cluster until each cluster contains an individual point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
 - Merge or split one cluster at a time

دو نوع اصلی خوشه بندی سلسله مراتبی
- تجمعی:
با نقاط به عنوان خوشه های فردی شروع کنید
در هر مرحله، نزدیکترین جفت خوشه ها را
ادغام کنید تا تنها یک خوشه (یا k خوشه) باقی
بماند

- نفرقه افکن:
با یک خوشه فراگیر شروع کنید
در هر مرحله، یک خوشه را تا زمانی تقسیم کنید که هر خوشه دارای
یک نقطه جداگانه باشد (یا k خوشه وجود داشته باشد)

الگوریتم های سلسله مراتبی سنتی از یک شباهت یا ماتریس فاصله استفاده می کنند
- ادغام یا تقسیم یک خوشه در یک زمان

Agglomerative Clustering Algorithm

Key Idea: Successively merge closest clusters

- Basic algorithm
 1. Compute the proximity matrix
 2. Let each data point be a cluster
 3. Repeat
 - 4. Merge the two closest clusters
 - 5. Update the proximity matrix
 6. Until only a single cluster remains

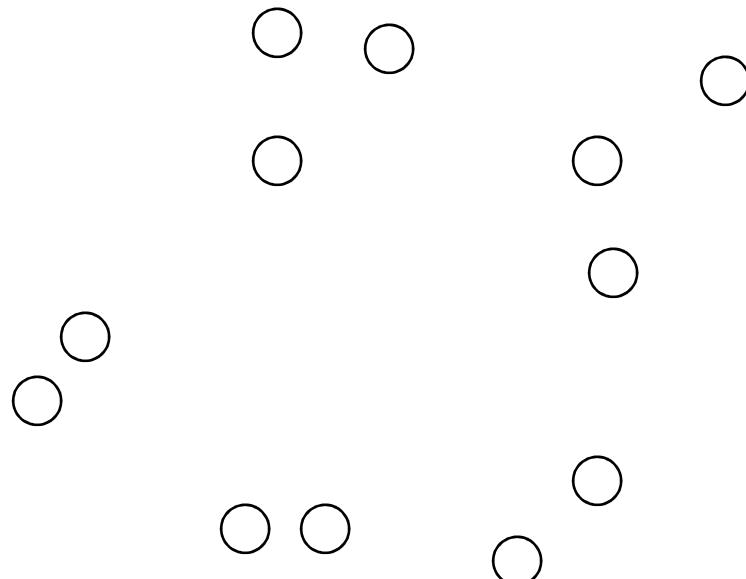
Key operation is the computation of the proximity of two clusters

- Different approaches to defining the distance between clusters distinguish the different algorithms

Steps 1 and 2

نقاطی که نزدیک به هم هستند را مرج میکنیم تا یک خوش پس باید فاصله ای بین هر دو نقطه را داشته باشیم.

- Start with clusters of individual points and a proximity matrix



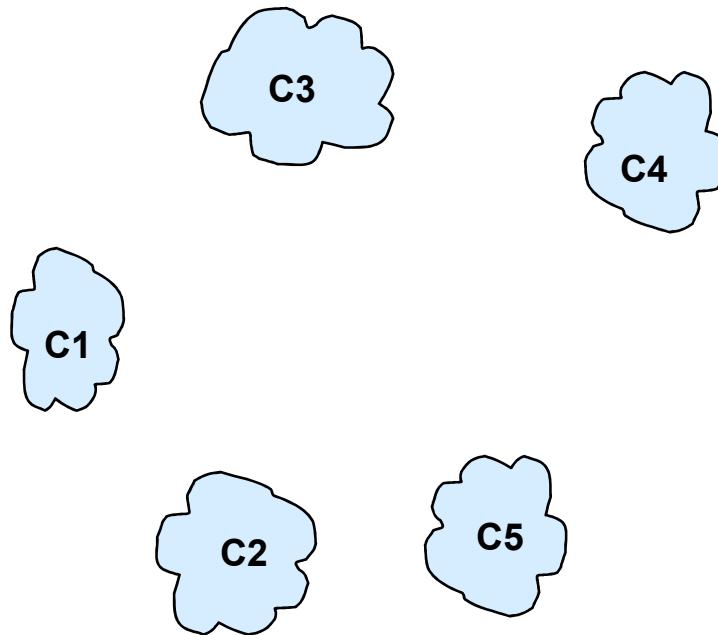
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						

Proximity Matrix

p1 p2 p3 p4 ... p9 p10 p11 p12

Intermediate Situation

- After some merging steps, we have some clusters

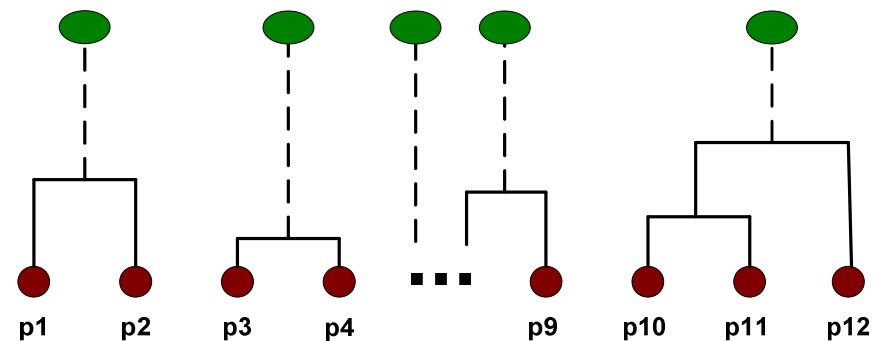


فاصله‌ی خوش‌های
بدست اومده از هم

	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

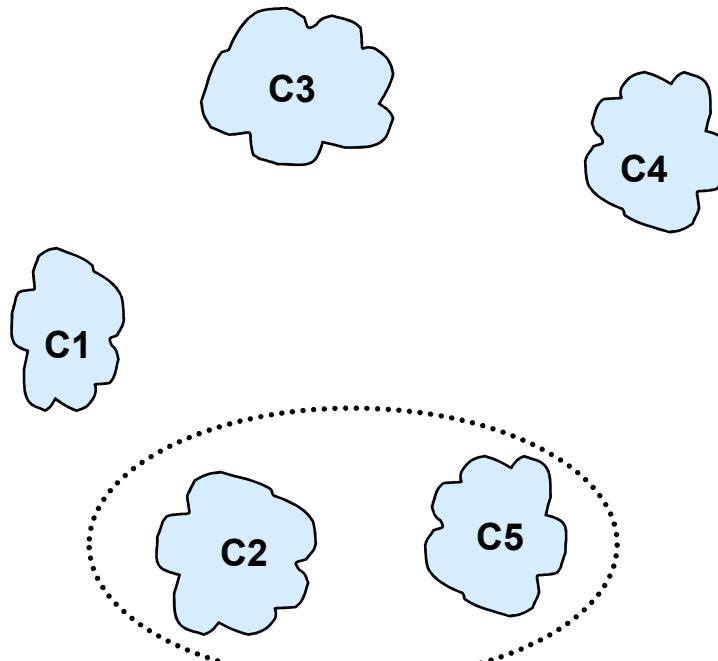
Proximity Matrix

ماتریس مجاورت



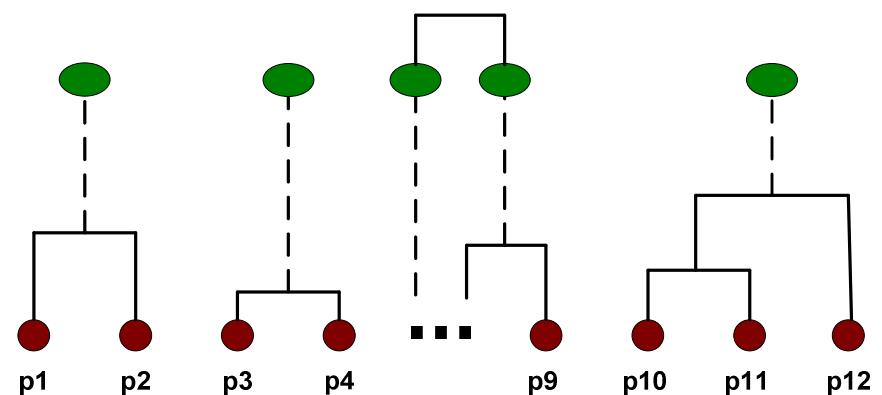
Step 4

- We want to merge the two closest clusters (**C2** and **C5**) and update the proximity matrix.



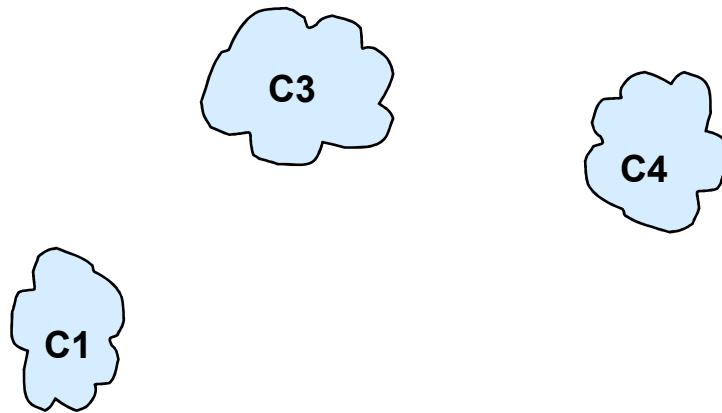
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



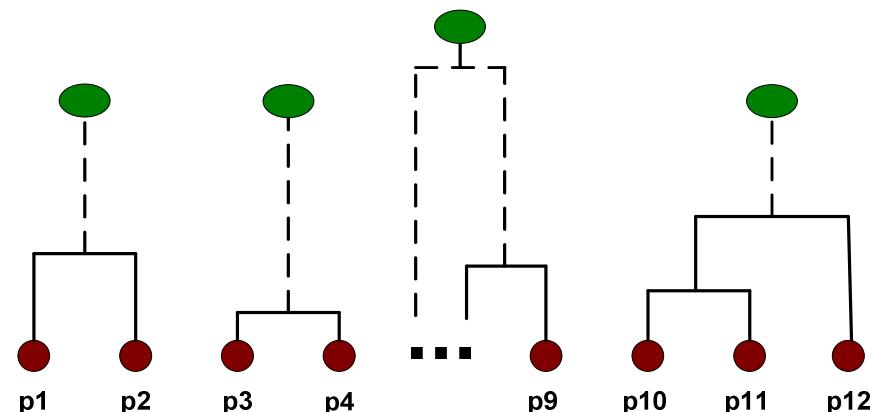
Step 5

- The question is “How do we update the proximity matrix?”

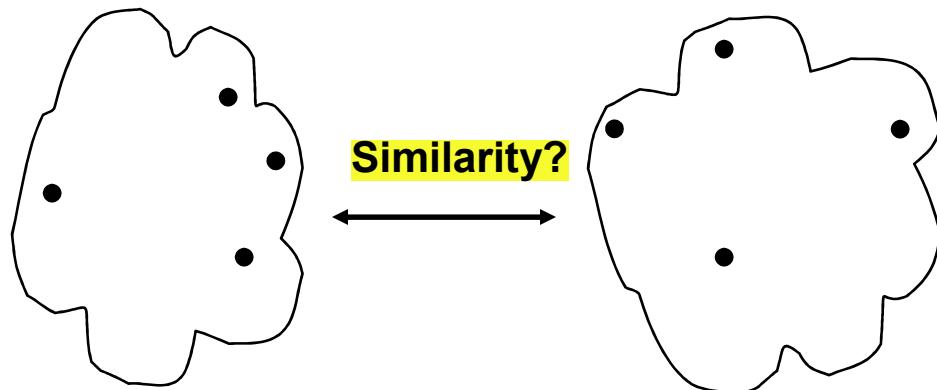


	C1	C5	C3	C4
C1	?			
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Distance

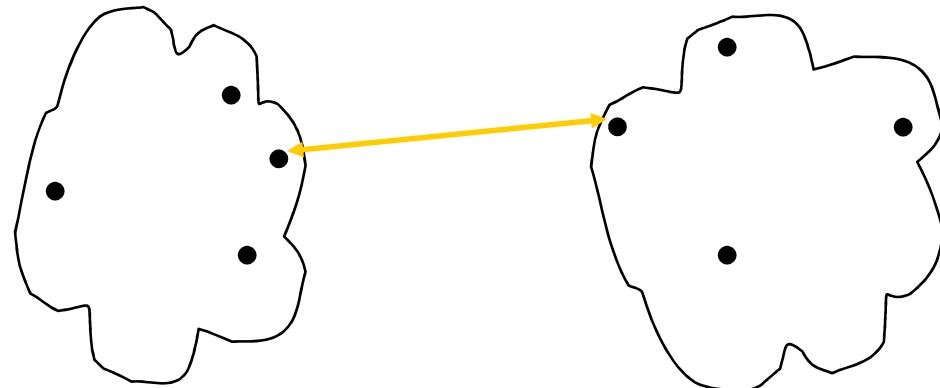


سوال: چطوری فاصله‌ی بین خوشه‌ها را
اپدیت کنیم؟

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

▪ Proximity Matrix

How to Define Inter-Cluster Similarity

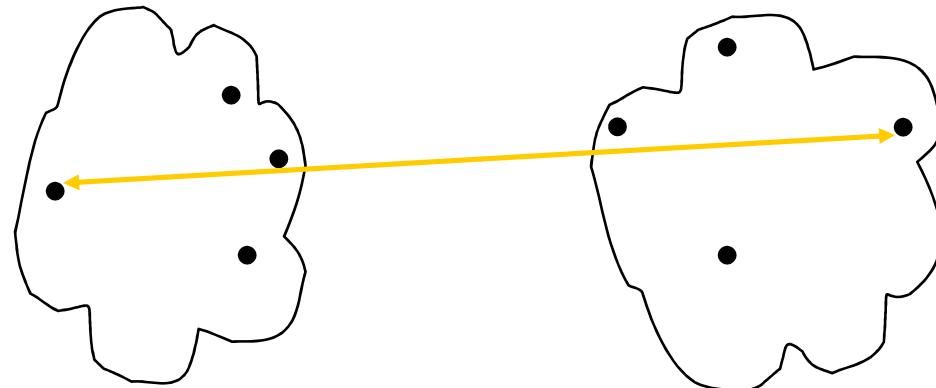


- MIN

	p1	p2	p3	p4	p5	...
p1						
.						

- Proximity Matrix

How to Define Inter-Cluster Similarity

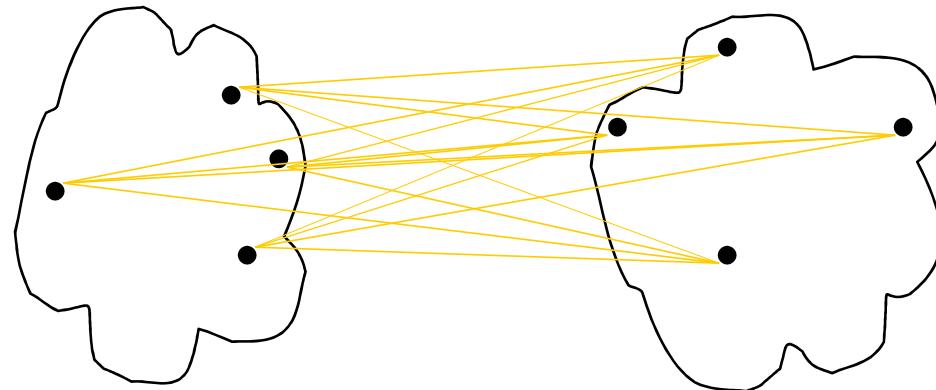


- MIN
- MAX

	p1	p2	p3	p4	p5	...
p1						
.

- Proximity Matrix

How to Define Inter-Cluster Similarity

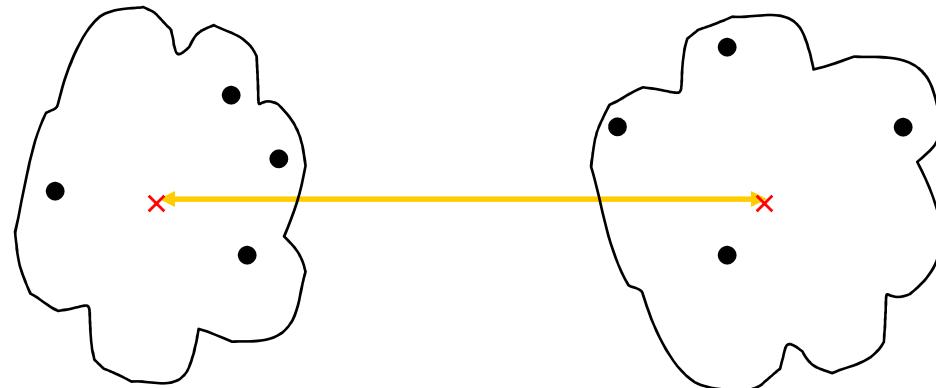


- MIN
- MAX
- Group Average

	p1	p2	p3	p4	p5	...
p1						
.						

· Proximity Matrix

How to Define Inter-Cluster Similarity



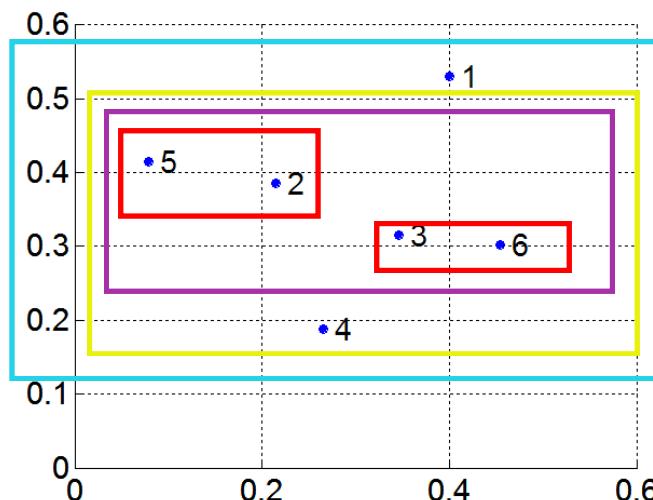
- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

· Proximity Matrix

MIN or Single Link

- Proximity of two clusters is based on the two closest points in the different clusters
 - Determined by one pair of points, i.e., by one link in the proximity graph
- Example:



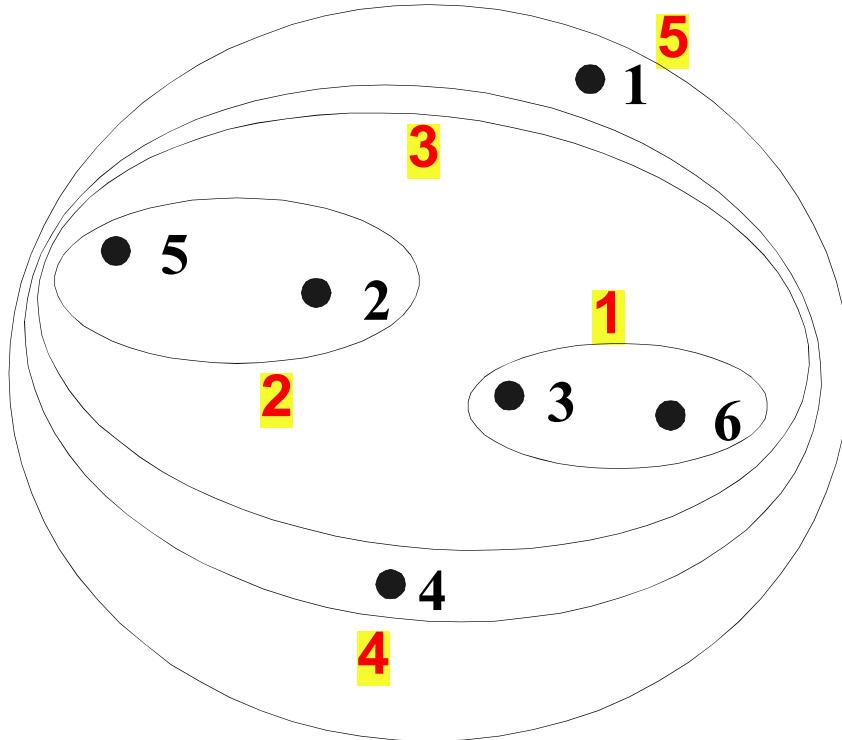
نقاط در یک فضای
دو بعدی نمایش داده
شدند

مجاورت دو خوشه بر اساس دو نزدیکترین نقطه در خوشه های مختلف است.
با یک جفت نقطه، به عنوان مثال، توسط یک پیوند در نمودار یا گراف مجاورت تعیین می شود

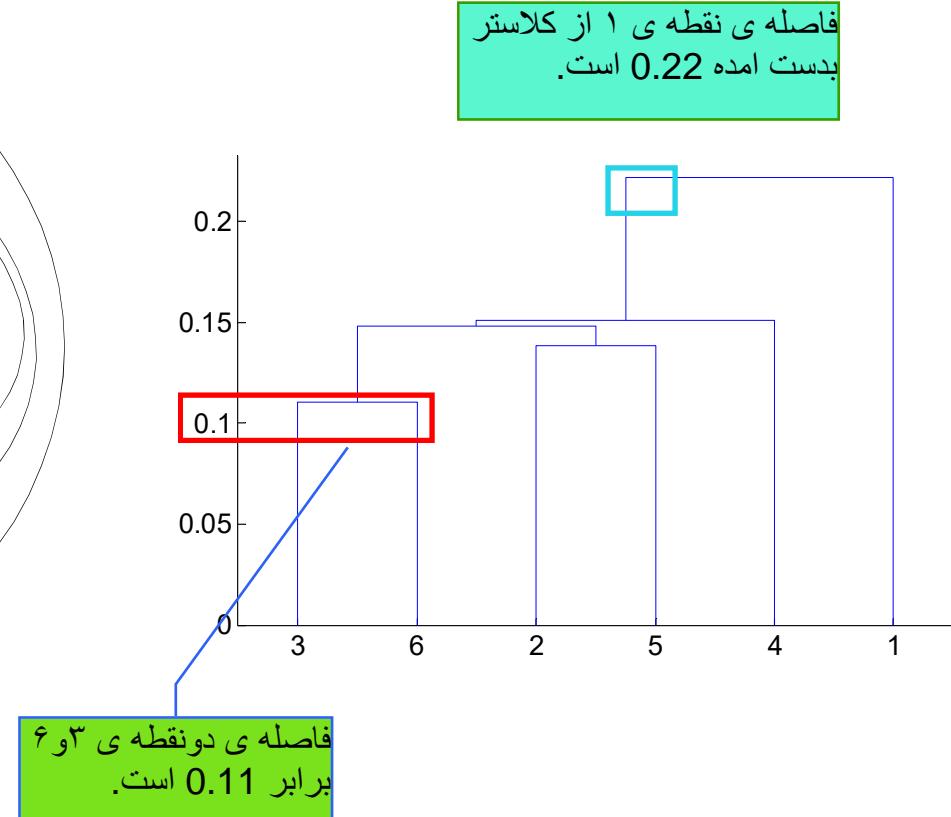
Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: MIN



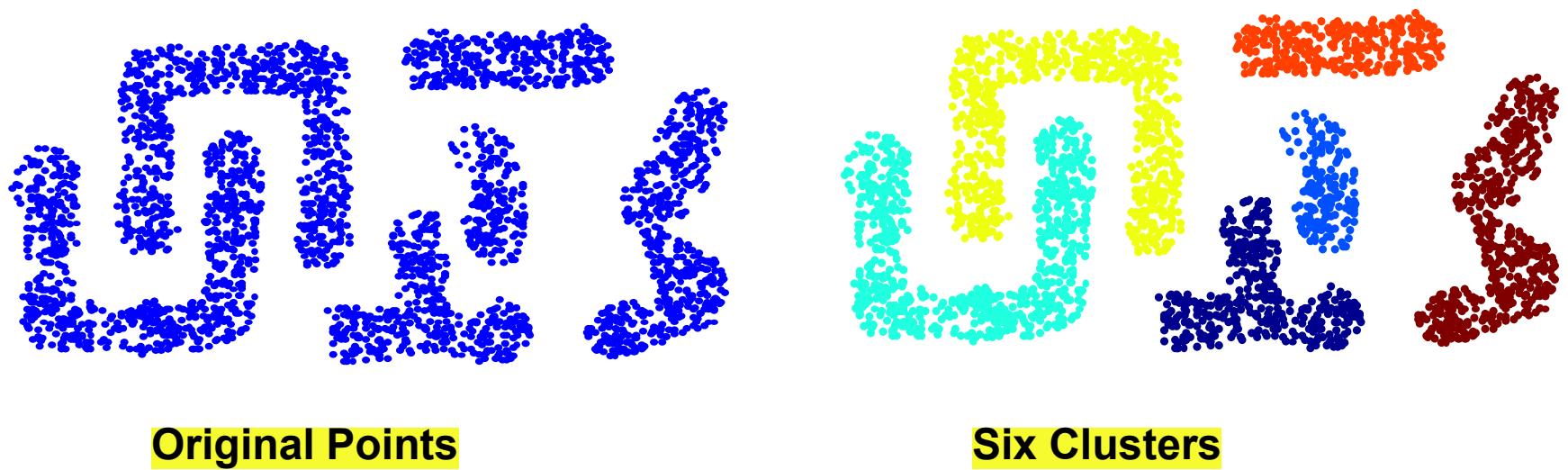
Nested Clusters



Dendrogram

دندروگرام:
یک نمودار درختی، به ویژه نموداری که
روابط طبقه بندی را نشان می دهد.

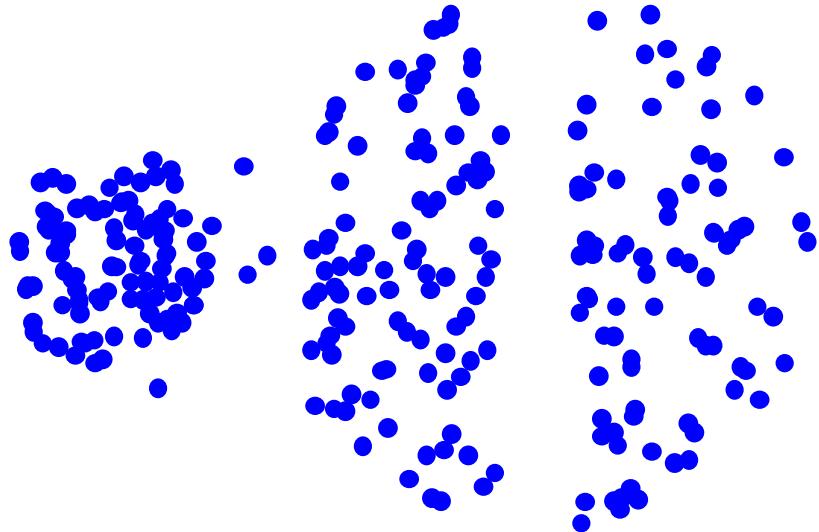
Strength of MIN



- Can handle non-elliptical shapes

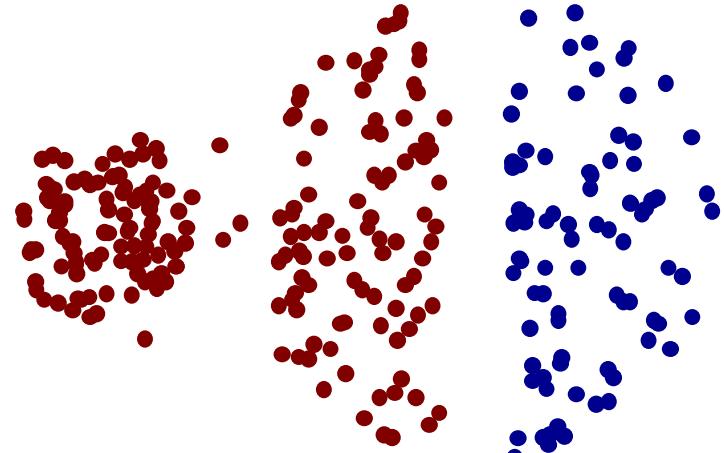
• می تواند اشکال غیر
بیضوی را اداره کند

Limitations of MIN

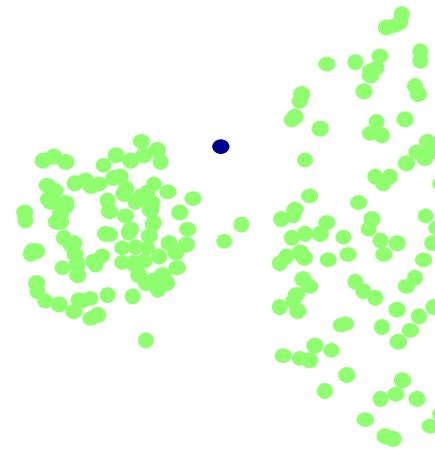


Original Points

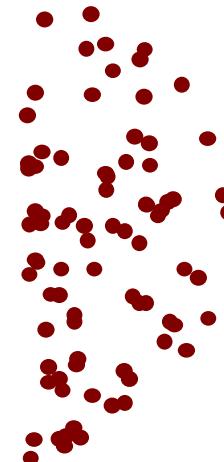
- Sensitive to noise



Two Clusters



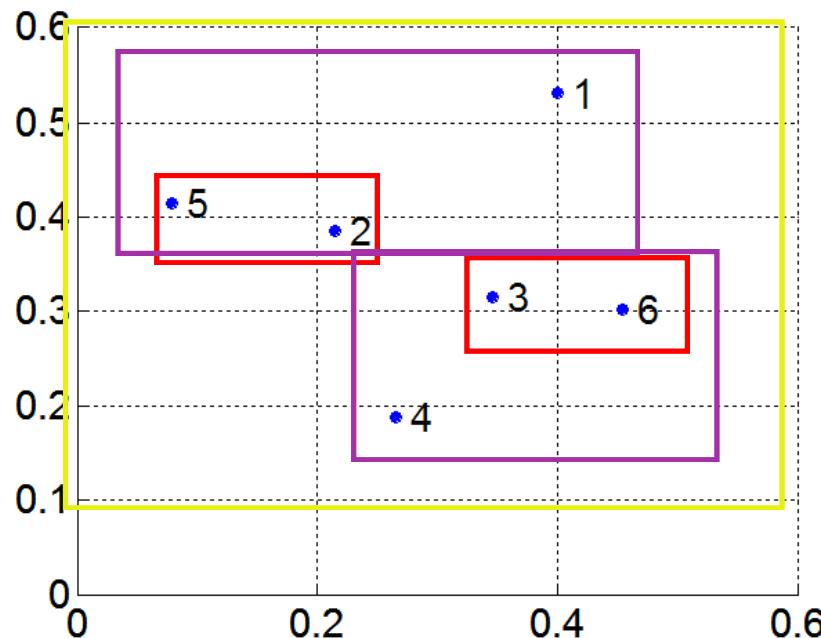
Three Clusters



مجاورت دو خوشه بر اساس دو دورترین نقطه در خوشه های مختلف است.
- توسط تمام جفت نقاط در دو خوشه تعیین می شود

MAX or Complete Linkage

- Proximity of two clusters is based on the two most distant points in the different clusters
 - Determined by all pairs of points in the two clusters



Distance Matrix:

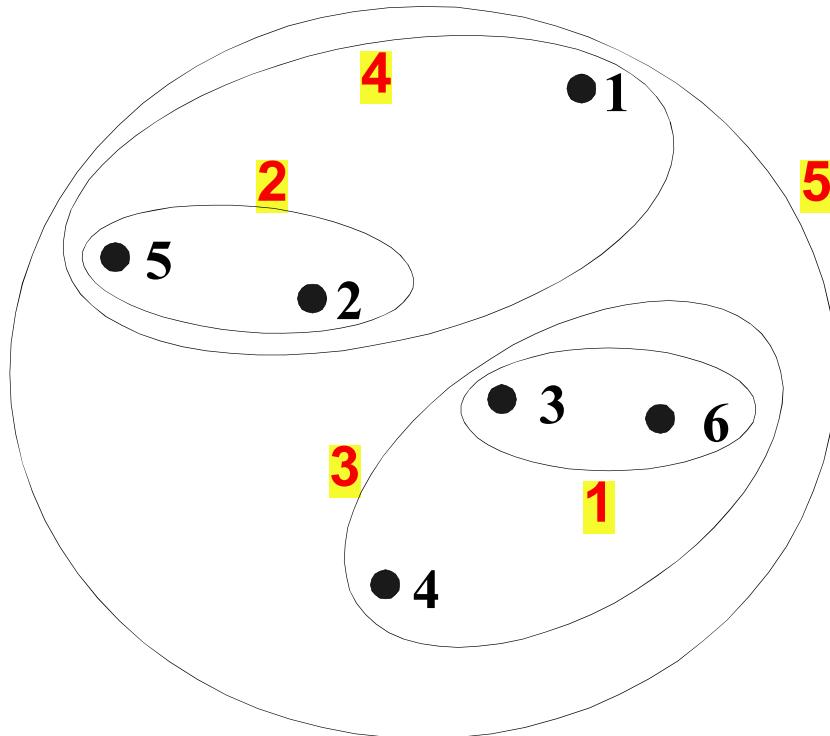
	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

معیار مون برای پیدا کردن نزدیک ترین خوشه ها، ماکس فاصله ای نقاط در دو خوشه است.

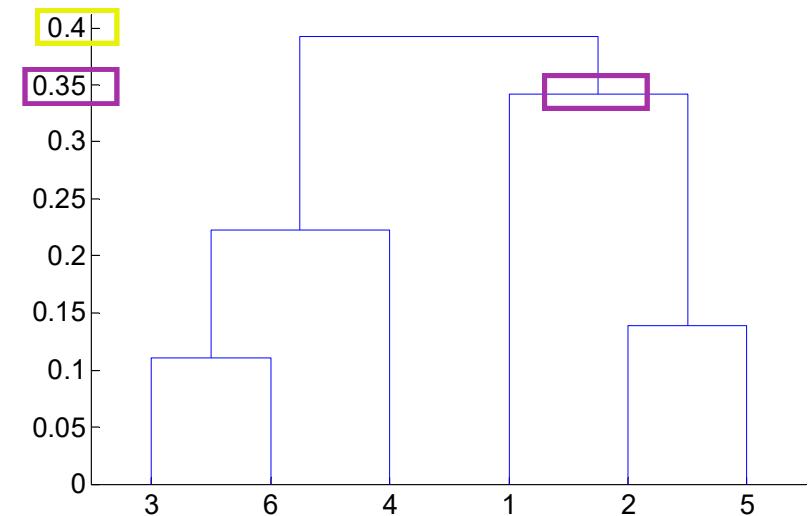
ین دو تا خوشه بین نفسی که درست شد، دو نقطه ای p5,p6 بیشترین فاصله را از هم دارند یعنی 0.39

در ابتدا که هیچ خوشه ای نداریم، نقاطی که کمترین فاصله از هم دارند را توانی یک خوشه قرار میدهیم چون p2,p5 کمترین فاصله را از هم دارند پس یک خوشه میشوند.

Hierarchical Clustering: MAX

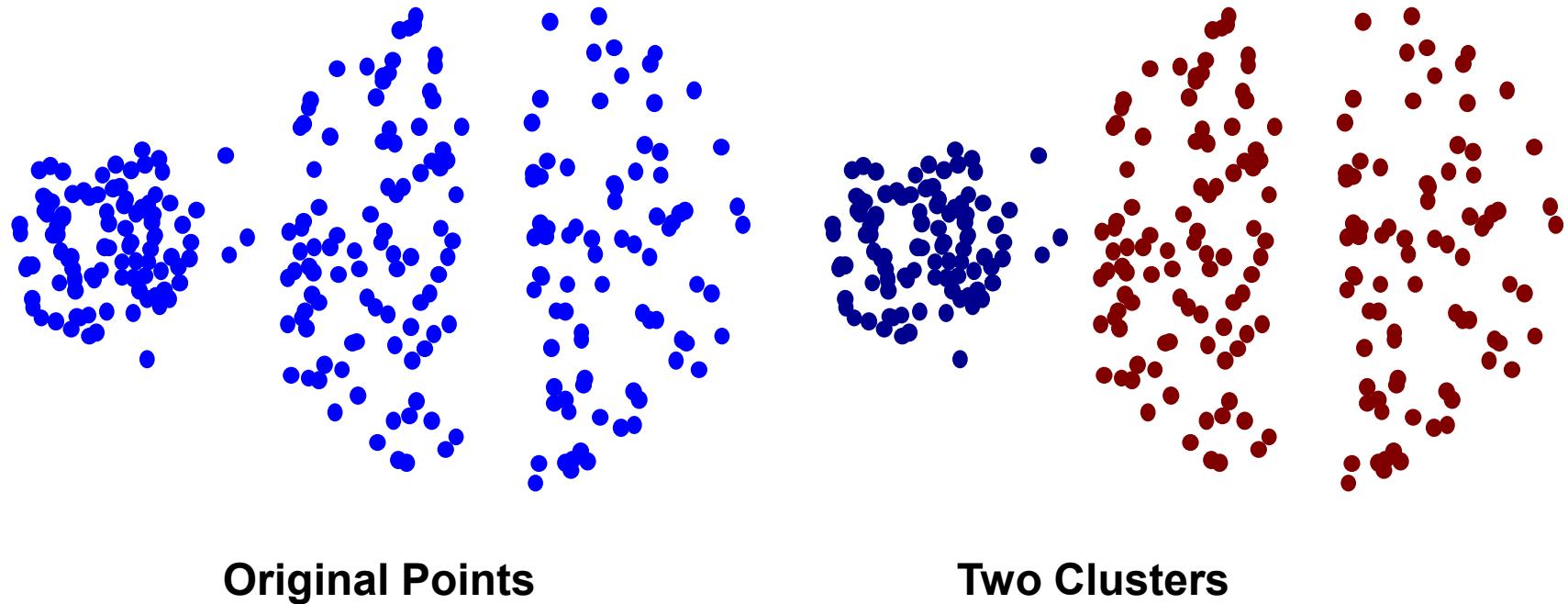


Nested Clusters



Dendrogram

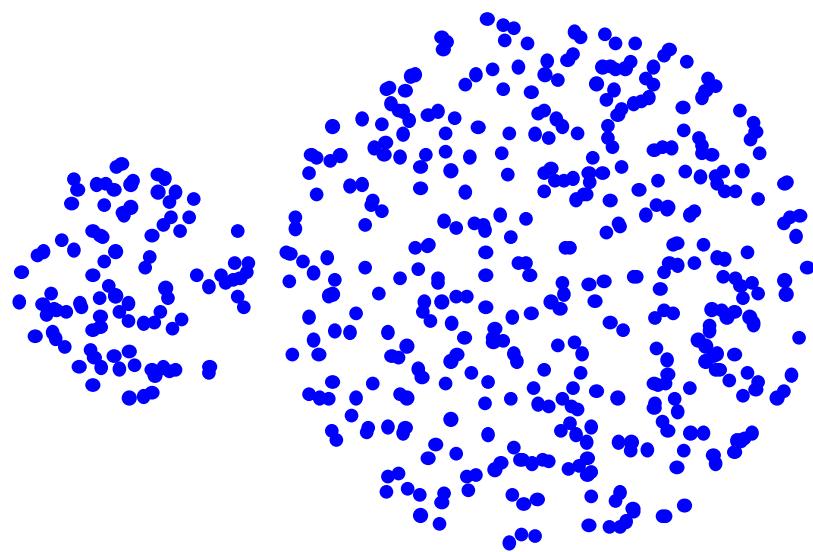
Strength of MAX



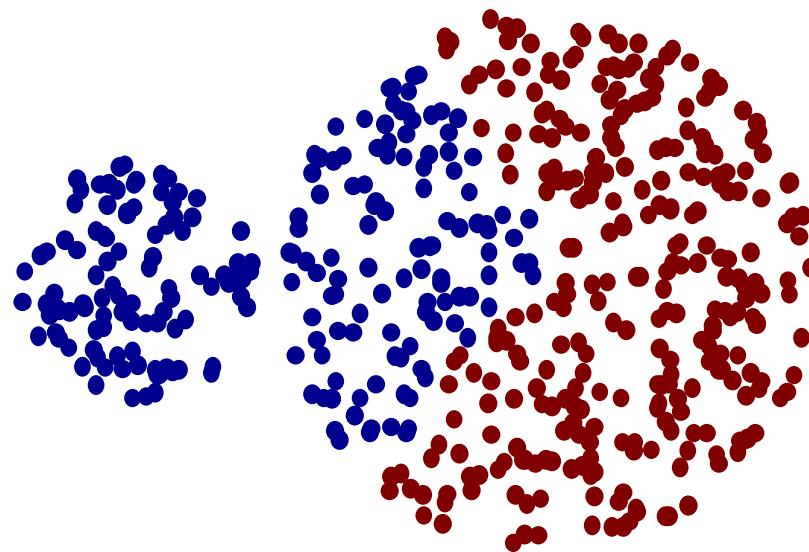
- Less susceptible to noise

کمتر مستعد نویز است

Limitations of MAX



Original Points



Two Clusters

- Tends to **break large clusters**
- Biased towards **globular clusters**

- تمایل به شکستن خوشه های بزرگ دارد
- گرایش به خوشه های کروی

Group Average

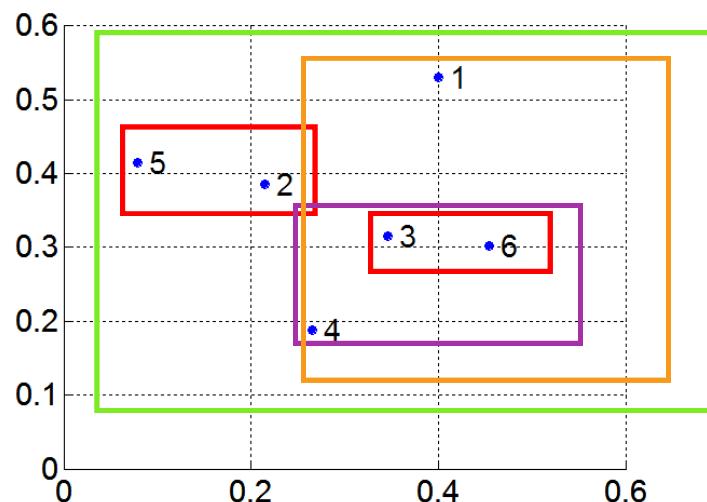
- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

مجاورت دو خوش میانگین
مجاورت زوجی بین نقاط دو خوش است.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| \times |\text{Cluster}_j|}$$

مجموع فاصله های همه نقاط در دو خوش

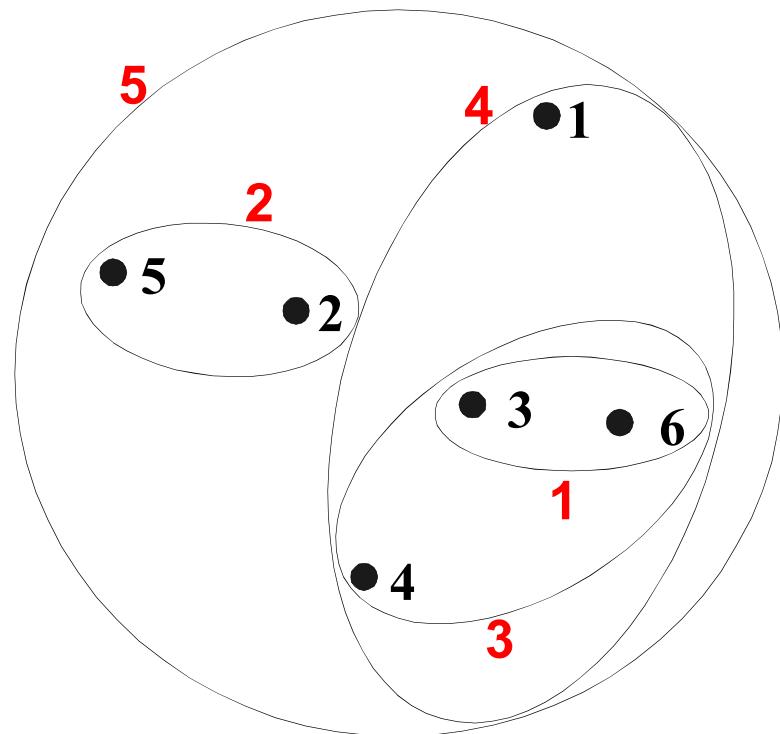
ضرب تعداد نقاط در دو خوش



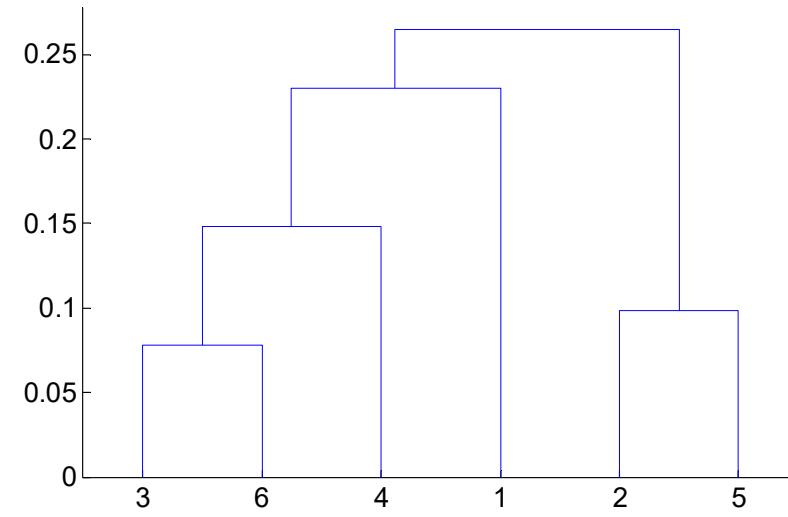
Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram

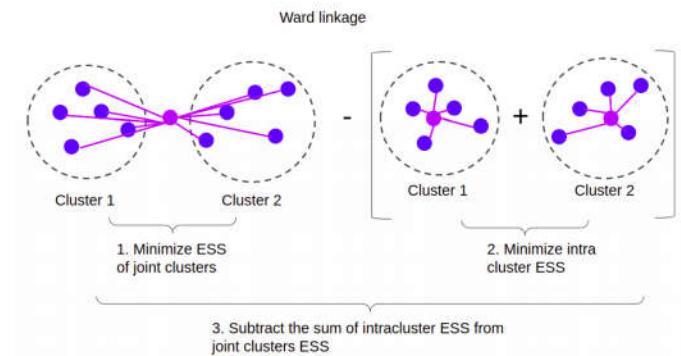
Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link
- Strengths
 - Less susceptible to noise
- Limitations
 - Biased towards globular clusters

سازش بین پیوند واحد و کامل
نقاط قوت
- کمتر مستعد نویز است
حدودیت ها
- گرایش به خوشه های کروی

Cluster Similarity: Ward's Method

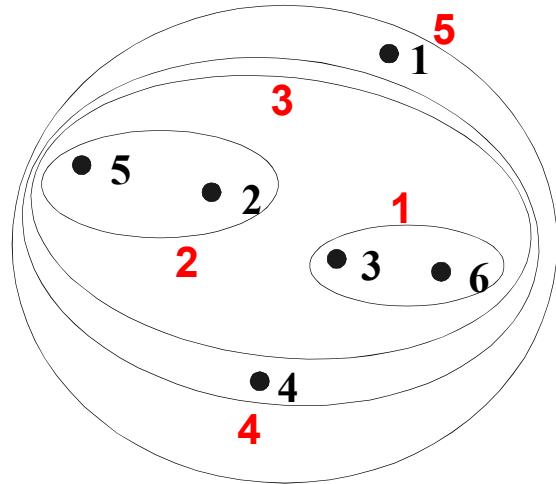
- Similarity of two clusters is based on the increase in squared error when two clusters are merged
 - Similar to group average if distance between points is distance squared
- Less susceptible to noise
- Biased towards globular clusters
- Hierarchical analogue of K-means
 - Can be used to initialize K-means



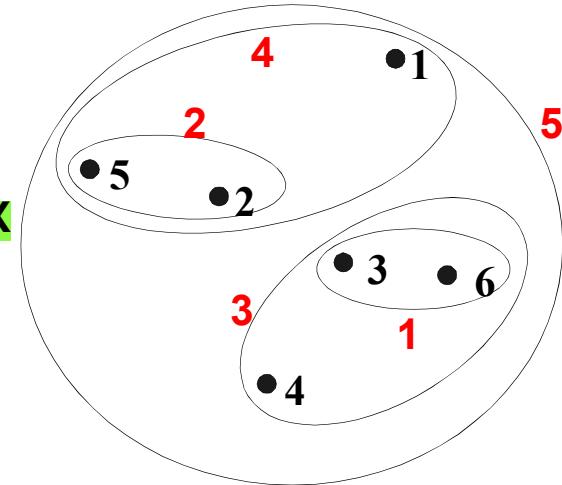
<https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/>

تشابه دو خوش بر اساس افزایش مربعات خطأ هنگام ادغام دو خوش است.
مشابه میانگین گروهی اگر فاصله بین نقاط مخذور فاصله باشد، کمتر مستعد نویز است
گرایش به خوش های کروی آنالوگ سلسله مراتبی K-means- می تواند برای مقداردهی اولیه K-means استفاده شود

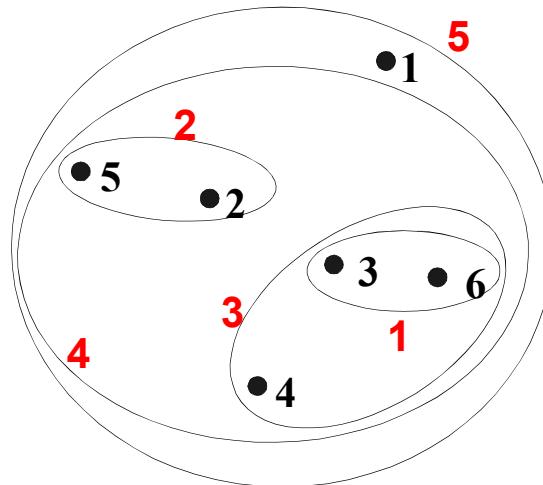
Hierarchical Clustering: Comparison



MIN

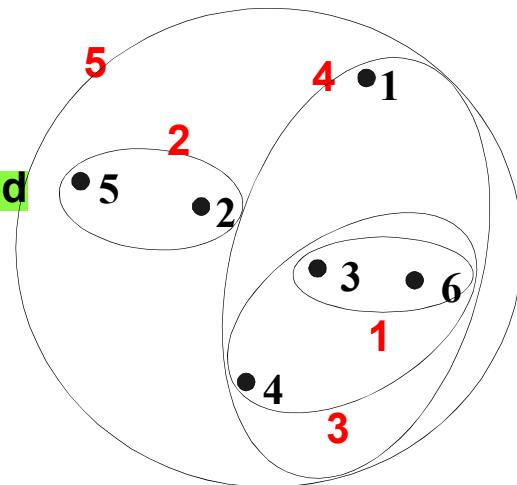


MAX



Group Average

Ward's Method



Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ space since it uses the proximity matrix.
 - N is the number of points.
- $O(N^3)$ 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 with some cleverness

Hierarchical Clustering: **Problems** and **Limitations**

- Once a decision is made to combine two clusters, it cannot be **undone**
- No global objective function is directly minimized
- Different schemes have problems with one or more of the following:
 - Sensitivity to noise
 - Difficulty handling clusters of different sizes and non-globular shapes

DENSITY BASED CLUSTERING

Density Based Clustering

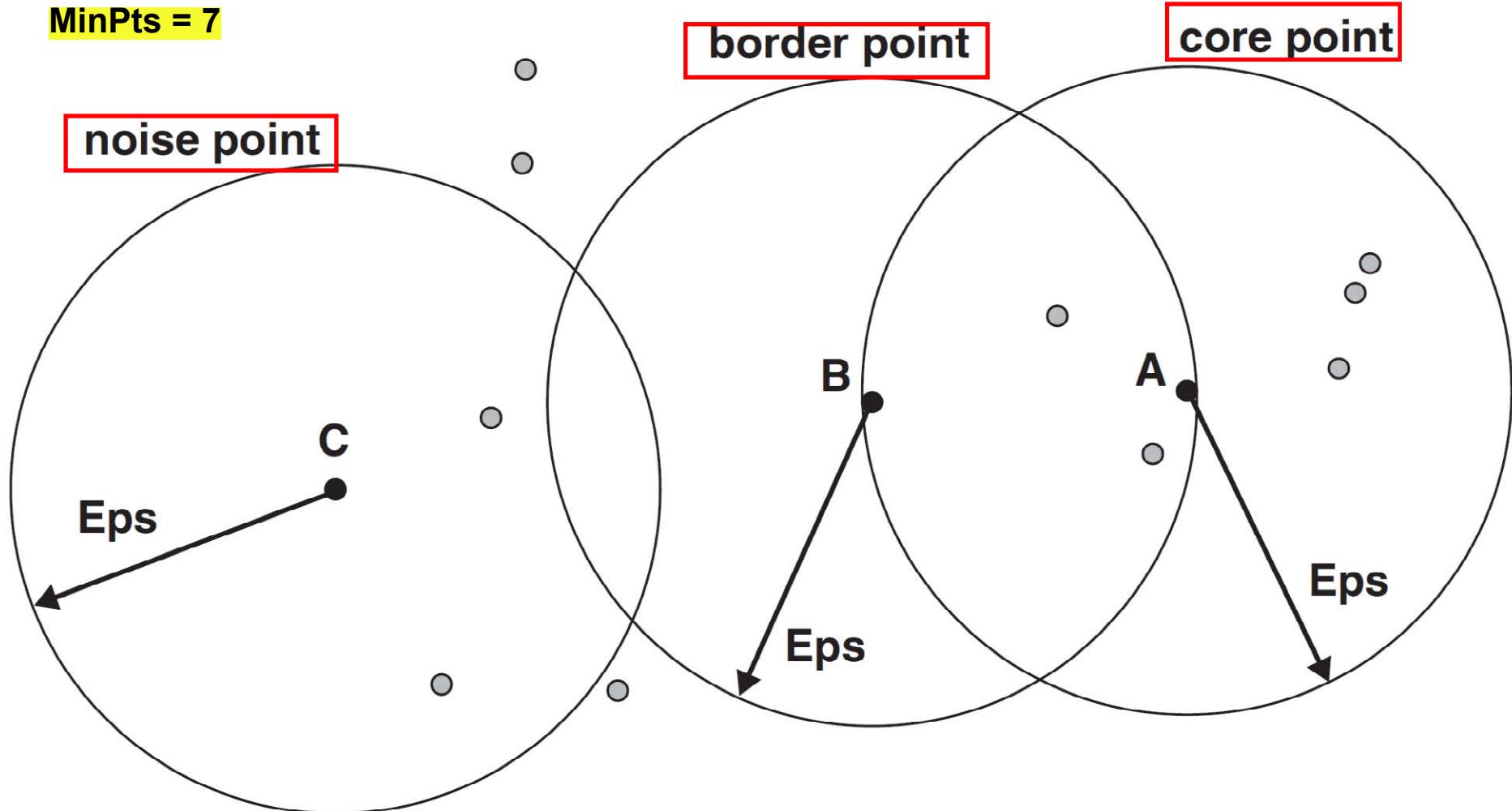
- Clusters are regions of high density that are separated from one another by regions of low density.
 - MinPts
 - Eps



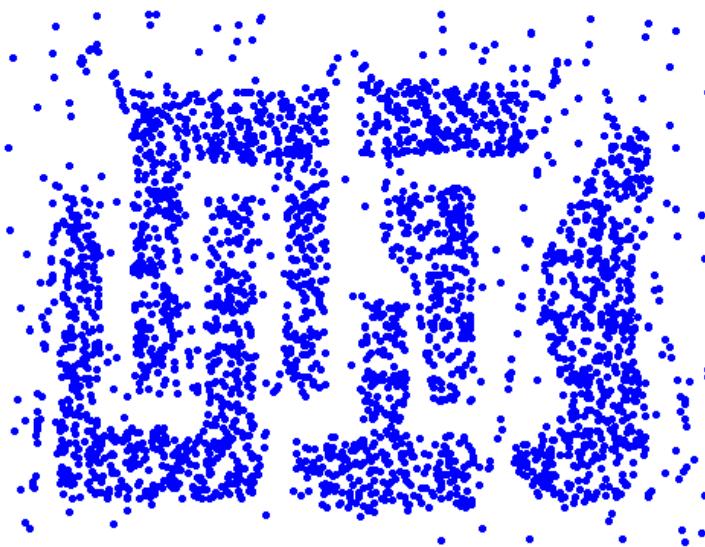
DBSCAN

- DBSCAN is a density-based algorithm.
 - Density = number of points within a specified radius (Eps)
 - A point is a core point if it has at least a specified number of points (MinPts) within Eps
 - ◆ These are points that are at the interior of a cluster
 - ◆ Counts the point itself
 - A border point is not a core point, but is in the neighborhood of a core point
 - A noise point is any point that is not a core point or a border point

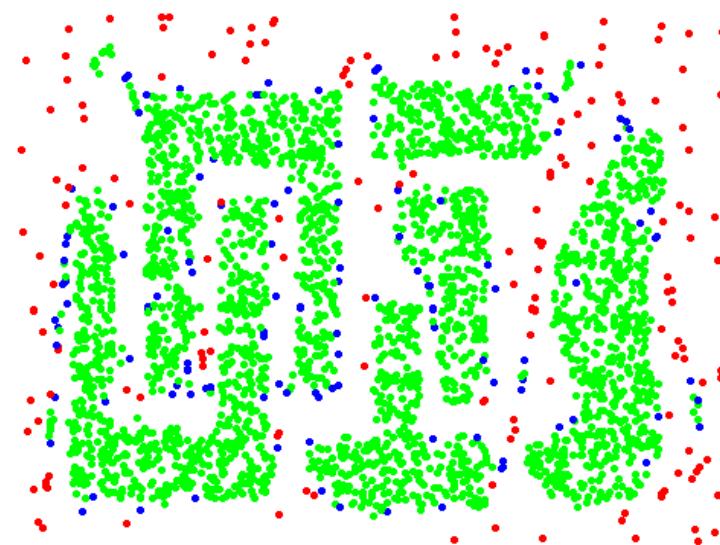
DBSCAN: Core, Border, and Noise Points



DBSCAN: Core, Border and Noise Points



Original Points



Point types: core,
border and noise

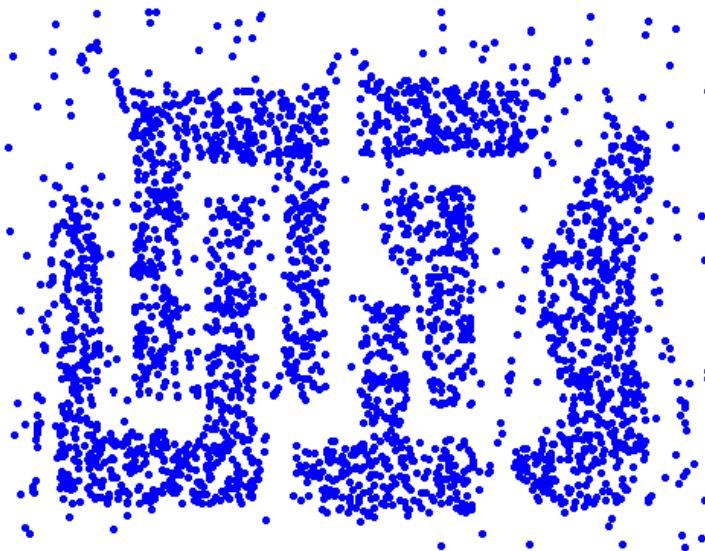
Eps = 10, MinPts = 4

DBSCAN Algorithm

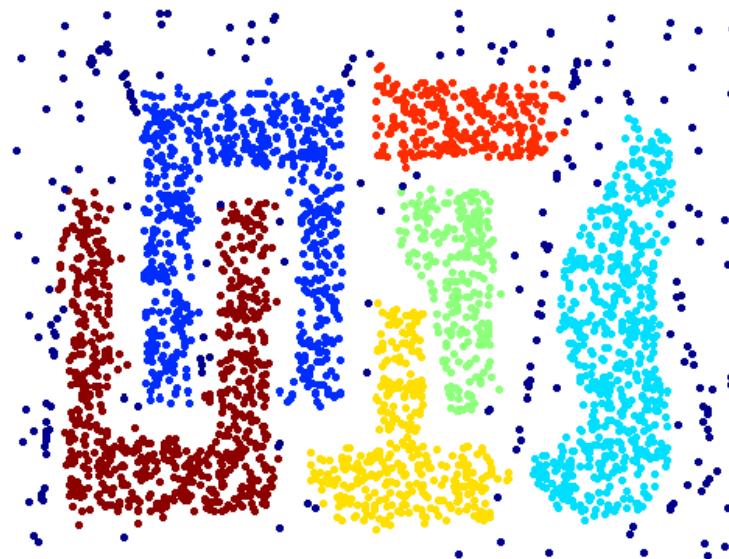
- Form clusters using core points, and assign border points to one of its neighboring clusters

- 1: Label all points as core, border, or noise points.
- 2: Eliminate noise points.
- 3: Put an edge between all core points within a distance Eps of each other.
- 4: Make each group of connected core points into a separate cluster.
- 5: Assign each border point to one of the clusters of its associated core points

When DBSCAN Works Well



Original Points

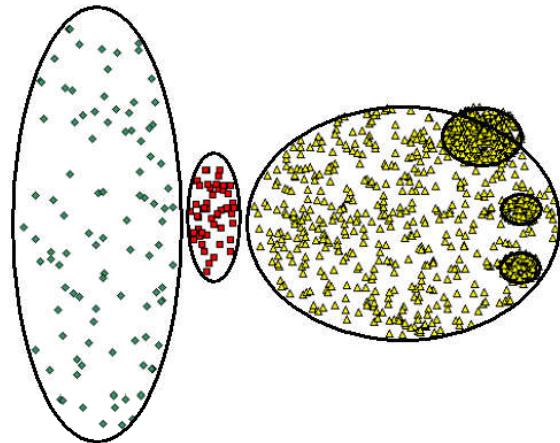


Clusters (dark blue points indicate noise)

- Can handle clusters of different shapes and sizes
- Resistant to noise

K is not required apriori

When DBSCAN Does NOT Work Well



Original Points

معایب:

DBScan با چگالی های مختلف به خوبی کار نمی کند (به زیر مراجعه کنید).
نقاط مرزی خودسرانه به خوشه های همسایه اختصاص داده می شوند.
اگر داده ها و مقیاس به خوبی درک نشده باشند، انتخاب یک آستانه فاصله معنی دار (ϵ) (Eps) می تواند دشوار باشد.
داده های بعد بالا محاسبات اقلیدسی را تنزل می دهد

Disadvantages:

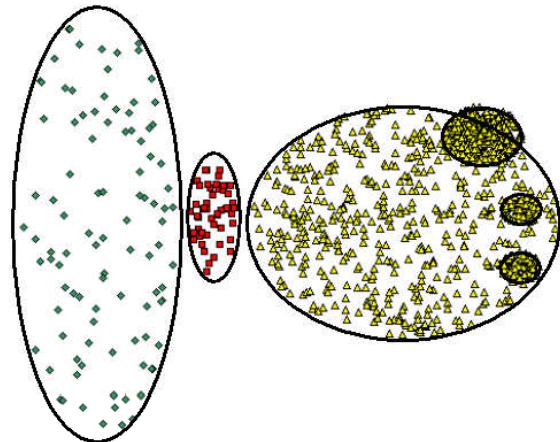
DBScan does not work well with varying densities (see below).

Border points are arbitrarily assigned to neighboring clusters.

If the data and scale are not well understood, choosing a meaningful distance threshold ϵ (Eps) can be difficult.

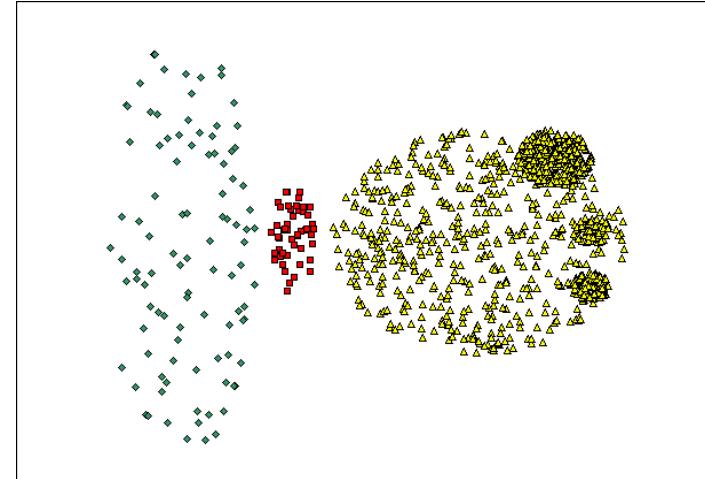
High dimension data degrades the Euclidean calculations

When DBSCAN Does NOT Work Well

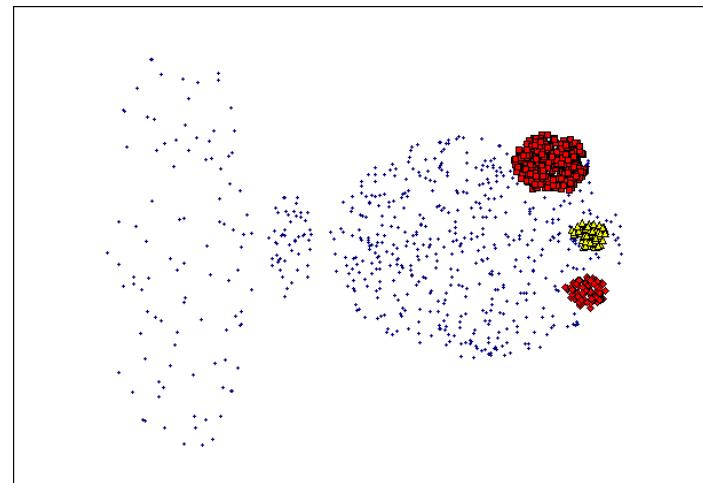


Original Points

- Varying densities
- High-dimensional data



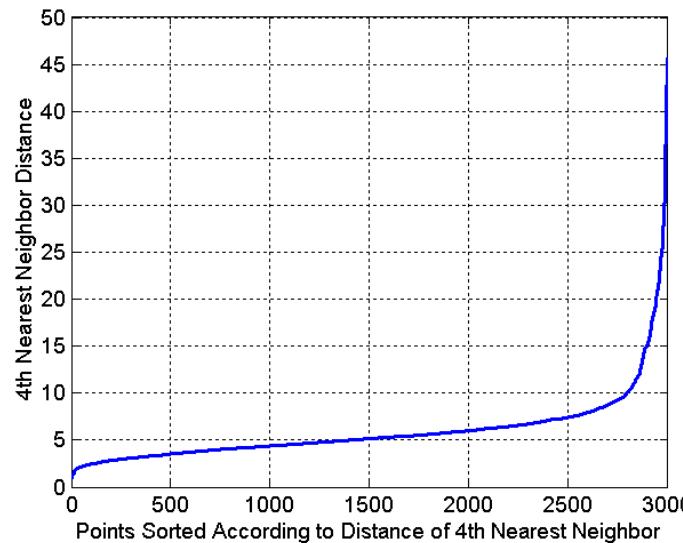
($\text{MinPts}=4$, $\text{Eps}=9.92$).



($\text{MinPts}=4$, $\text{Eps}=9.75$)

DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at close distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor

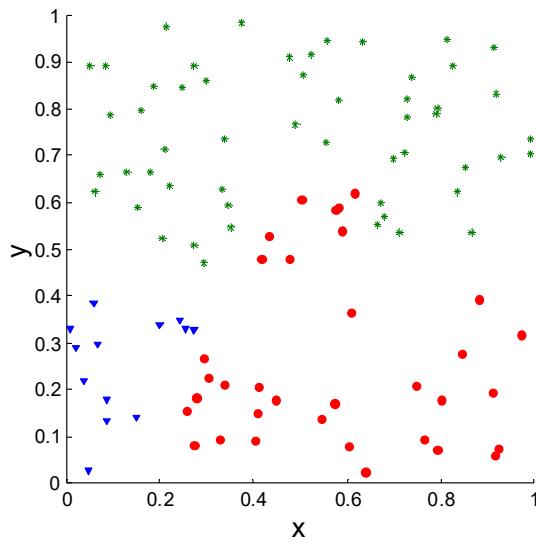
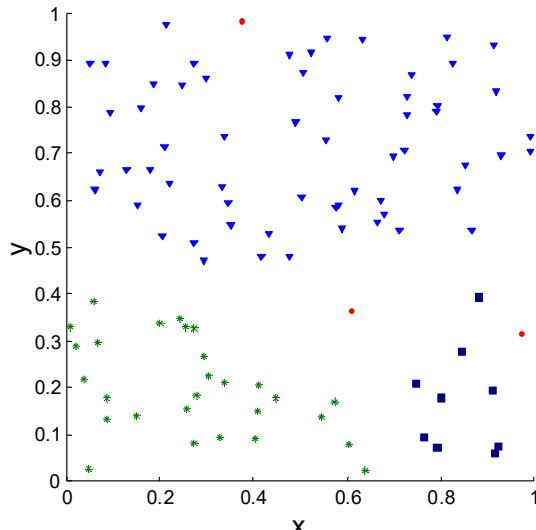
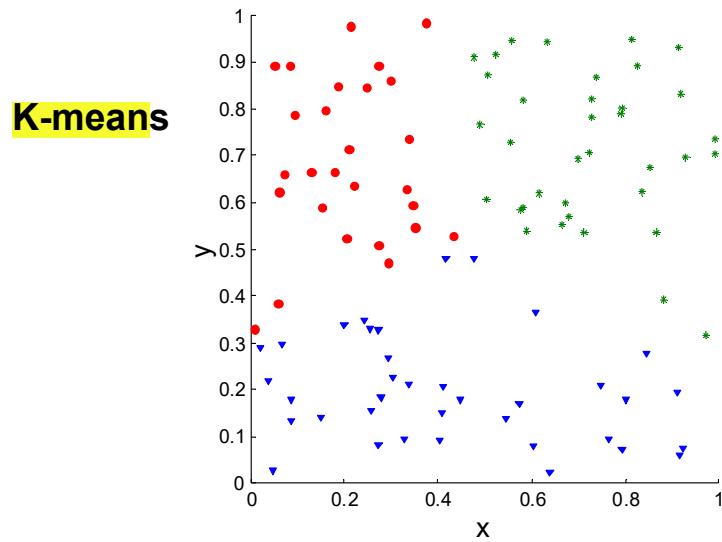
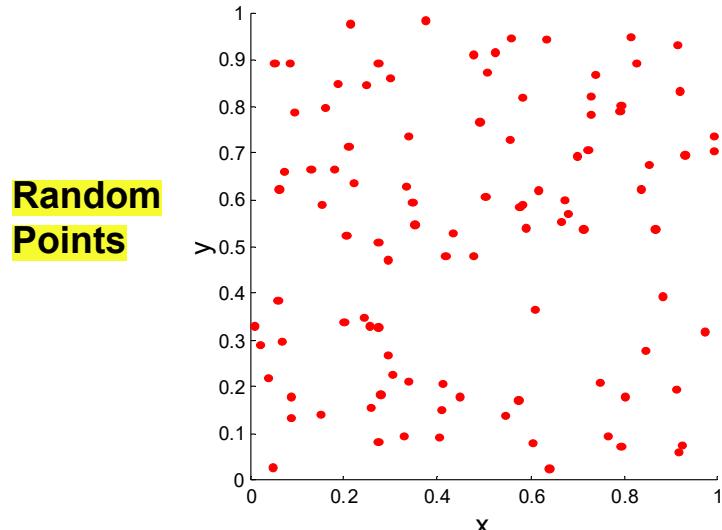


CLUSTER EVALUATION

Cluster Validity

- For supervised classification we have a variety of measures to evaluate how good our model is
 - Accuracy, precision, recall
- For cluster analysis, the analogous question is how to evaluate the “goodness” of the resulting clusters?
- But “clusters are in the eye of the beholder”!
 - In practice the clusters we find are defined by the clustering algorithm
- Then why do we want to evaluate them?
 - To avoid finding patterns in noise
 - To compare clustering algorithms
 - To compare two sets of clusters
 - To compare two clusters

Clusters found in Random Data



Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following two types.
 - Supervised: Used to measure the extent to which cluster labels match externally supplied class labels.
 - ◆ Entropy
 - ◆ Often called *external indices* because they use information external to the data
 - Unsupervised: Used to measure the goodness of a clustering structure *without* respect to external information.
 - ◆ Sum of Squared Error (SSE)
 - ◆ Often called *internal indices* because they only use information in the data
- You can use supervised or unsupervised measures to compare clusters or clusterings

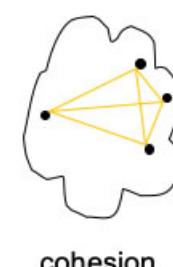
Unsupervised Measures: Cohesion and Separation

- **Cluster Cohesion:** Measures how closely related are objects in a cluster
 - Example: SSE
 - **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)
 - Separation is measured by the between cluster sum of squares
- Where $|C_i|$ is the size of cluster i

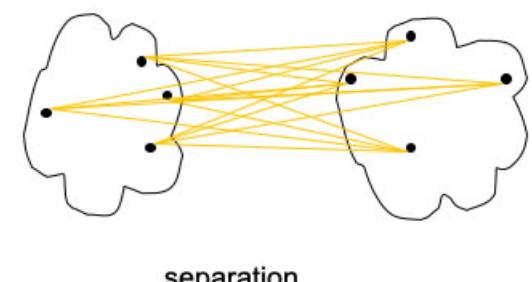
میزان یکپارچه بودن اعضای یک خوش را بررسی میکنے

$$SSE = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

$$SSB = \sum_i |C_i|(m - m_i)^2$$



cohesion



separation

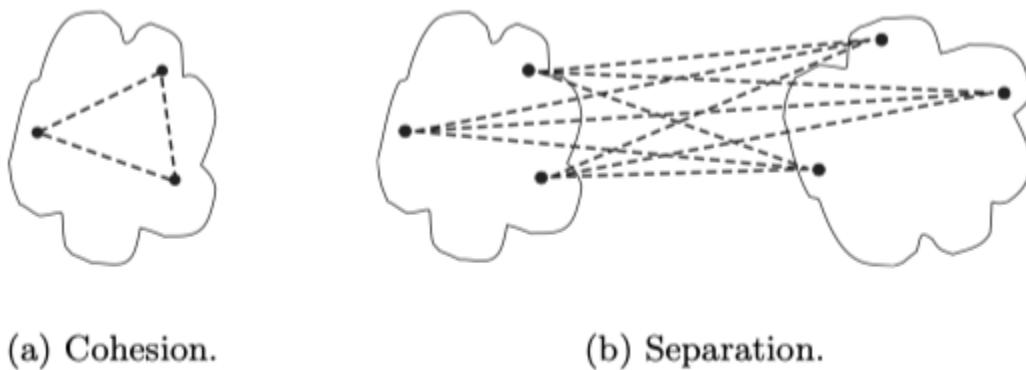


Figure 8.27. Graph-based view of cluster cohesion and separation.

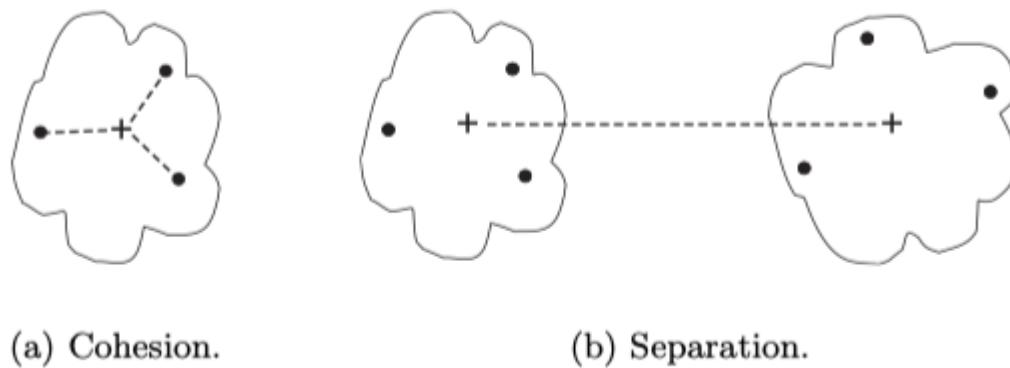


Figure 8.28. Prototype-based view of cluster cohesion and separation.

Cohesion and Separation

- **Cohesion** is measured by the within cluster sum of squares

$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

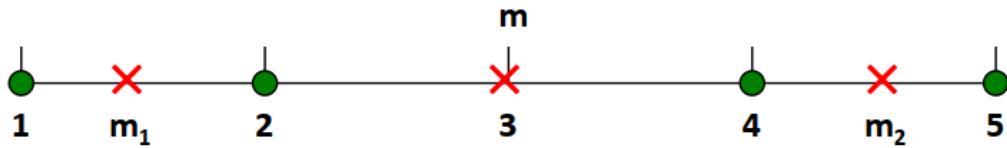
- **Separation** is measured by the between cluster sum of squares

$$BSS = \sum_i |C_i| (m - m_i)^2$$

where $|C_i|$ is the size of cluster i , m is the centroid of the whole data set

- $BSS + WSS = \text{constant}$
- WSS (Cohesion) measure is called Sum of Squared Error (**SSE**)—a commonly used measure
- A larger number of clusters tend to result in smaller SSE

Example



$$WSS = (1 - 3)^2 + (2 - 3)^2 + (4 - 3)^2 + (5 - 3)^2 = 10$$

K=1 :

$$BSS = 4 \times (3 - 3)^2 = 0$$

$$Total = 10 + 0 = 10$$

K=2 :

$$WSS = (1 - 1.5)^2 + (2 - 1.5)^2 + (4 - 4.5)^2 + (5 - 4.5)^2 = 1$$

$$BSS = 2 \times (3 - 1.5)^2 + 2 \times (4.5 - 3)^2 = 9$$

$$Total = 1 + 9 = 10$$

K=4:

$$WSS = (1 - 1)^2 + (2 - 2)^2 + (4 - 4)^2 + (5 - 5)^2 = 0$$

$$BSS = 1 \times (1 - 3)^2 + 1 \times (2 - 3)^2 + 1 \times (4 - 3)^2 + 1 \times (5 - 3)^2 = 10$$

$$Total = 0 + 10 = 10$$

Unsupervised Measures: Cohesion and Separation

- **Cluster Cohesion:** Measures how closely related are objects in a cluster
 - Example: SSE
- **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)

$$SSE = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

- Separation is measured by the between cluster sum of squares

$$SSB = \sum_i |C_i| (m - m_i)^2$$

Where $|C_i|$ is the size of cluster i

میانگین هر خوش را از
میانگین کل داده ها کم
میکنیم و به توان ۲
بررسونیم و در تعداد داده
های اون کلاستر ضرب
میکنیم

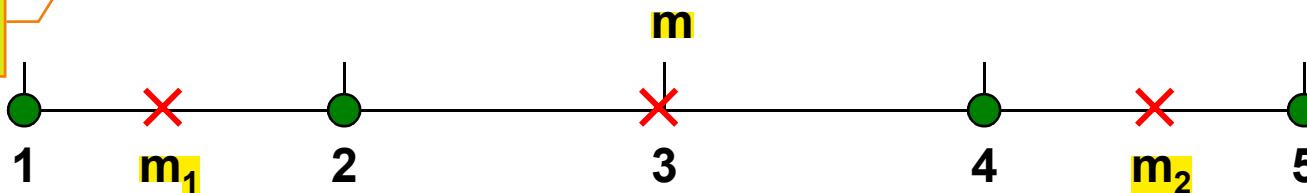
هر داده را از میانگین
کلاستری که توشه کم
میکنیم و به توان ۲
بررسونیم و این کار را به
ازای تمام داده های توی
کلاستر های مختلف انجام
میدیم

Unsupervised Measures: Cohesion and Separation

- Example: SSE

— $SSB + SSE = \text{constant}$

separation square
error between
clusters



SSE پراکندگی داده ها را حول میانگین داده های توى یک کلاستر اندازه میگیره که هرچی کمتر باشه نشون میده پراکندگی کمتره و داده ها به میانگین نزدیک تر هستند.

K=1 cluster:

$$SSE = (1 - 3)^2 + (2 - 3)^2 + (4 - 3)^2 + (5 - 3)^2 = 10$$

$$SSB = 4 \times (3 - 3)^2 = 0$$

$$Total = 10 + 0 = 10$$

فاصله ای بین خوشه ها
هم بیشتر شده پس در کل
از نتیجه ای این دو تا
معیار مفهومیم 2
بهتر است

K=2 clusters:

$$SSE = (1 - 1.5)^2 + (2 - 1.5)^2 + (4 - 4.5)^2 + (5 - 4.5)^2 = 1$$

$$SSB = 2 \times (3 - 1.5)^2 + 2 \times (4.5 - 3)^2 = 9$$

$$Total = 1 + 9 = 10$$

جمعشون ثابت میشه

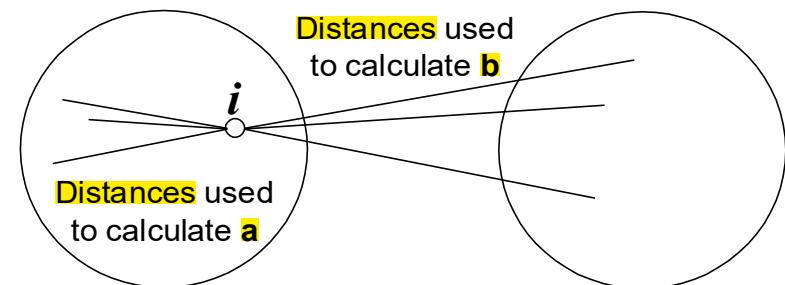
وقتی $k=2$ باشه
sse=1 میشه یعنی پراکندگی داده های هر کلاستر
نسبت به میانگین داده های توى کلاستر کمتر شده (از
10 شده 1)
پس افزایش تعداد کلاستر از یک به دو باعث بهتر شدن
معیار ارزیابی ما شده

Unsupervised Measures: Silhouette Coefficient

- Silhouette coefficient combines ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings
- For an individual point, i
 - Calculate a = average distance of i to the points in its cluster
 - Calculate b = \min (average distance of i to points in another cluster)
- The silhouette coefficient for a point is then given by

$$s = (b - a) / \max(a, b)$$

برای نرمال کردن این
معیار



- Value can vary between -1 and 1
- Typically ranges between 0 and 1.
- The closer to 1 the better.

- Can calculate the average silhouette coefficient for a cluster or a clustering

فاصله‌ی یک نقطه
از میانگین داده‌های توی
نوشه‌های دیگه چقدر تفاوت
داره با فاصله نقطه با
میانگین داده‌ها توی خوش
خودش

The Silhouette Coefficient is a measure of how similar an object is to its own cluster compared to other clusters. It is commonly used in clustering analysis to evaluate the quality of the clustering results.

The formula for Silhouette Coefficient is as follows:

$$s(i) = (b(i) - a(i)) / \max(a(i), b(i))$$

where $s(i)$ is the Silhouette Coefficient of the i -th observation, $a(i)$ is the average dissimilarity of the i -th observation to all other data points in the same cluster, and $b(i)$ is the average dissimilarity of the i -th observation to all data points in the nearest neighboring cluster that is not the same as the current cluster.

The Silhouette Coefficient ranges from -1 to 1. A score of 1 indicates that the object is very well-matched to its own cluster and poorly matched to neighboring clusters, while a score of -1 indicates the opposite. A score of 0 indicates that the object is on the border between two different clusters.

For example, let's say we have a dataset of customer transactions, and we want to cluster them into groups based on their purchasing behavior. We decide to use k-means clustering with $k=3$. After running the algorithm, we calculate the Silhouette Coefficient for each customer. Here are some hypothetical results:

Customer A: $s(A) = 0.85$

Customer B: $s(B) = -0.25$

Customer C: $s(C) = 0.56$

In this case, Customer A has a high Silhouette Coefficient, indicating that it belongs to its own cluster and is well-separated from other clusters. Customer B has a negative Silhouette Coefficient, suggesting that it may actually belong to another cluster. Customer C has a moderate Silhouette Coefficient, indicating that it is somewhat ambiguous and could belong to either of two clusters.

Overall, the Silhouette Coefficient is a useful metric for evaluating clustering algorithms and assessing the quality of clusters. However, it should be used in conjunction with other evaluation metrics to ensure that the clustering results are accurate and meaningful.

Sure, let's use a dataset of customer purchasing behavior to demonstrate how the Silhouette Coefficient can be calculated.

Suppose we have a dataset with 10 customers and two features: "Amount Spent" and "Frequency of Purchases". We want to cluster these customers into two groups based on their purchasing behavior.

Customer ID	Amount Spent	Frequency of Purchases
1	50	2
2	100	4
3	20	1
4	80	5
5	60	3
6	30	2
7	70	4
8	90	5
9	40	1
10	10	1

Customer ID	Amount Spent	Frequency of Purchases
1	50	2
2	100	4
3	20	1
4	80	5
5	60	3
6	30	2
7	70	4
8	90	5
9	40	1
10	10	1

We will use k-means clustering with k=2 to create the clusters. After running the algorithm, we can calculate the Silhouette Coefficient for each customer as follows:

Step 1: Calculate the distance between each customer and all other customers in their own cluster

For example, let's consider customer 1. The other customers in its cluster are customers 3 and 6. We can calculate the average distance between customer 1 and these two other customers as follows:

$$d(1,3) = \sqrt{(50-20)^2 + (2-1)^2} = 30.16$$

$$d(1,6) = \sqrt{(50-30)^2 + (2-2)^2} = 20$$

$$a(1) = (30.16 + 20) / 2 = 25.08$$

Similarly, we can calculate $a(i)$ for all other customers in their own cluster.

Step 2: Calculate the distance between each customer and all other customers in the nearest neighboring cluster

For example, let's consider customer 1 again. The nearest neighboring cluster is the one that contains customers 2, 4, 5, 7, and 8. We can calculate the average distance between customer 1 and these five customers as follows:

$$d(1,2) = \sqrt{(50-100)^2 + (2-4)^2} = 50.99$$

$$d(1,4) = \sqrt{(50-80)^2 + (2-5)^2} = 28.46$$

$$d(1,5) = \sqrt{(50-60)^2 + (2-3)^2} = 10.05$$

$$d(1,7) = \sqrt{(50-70)^2 + (2-4)^2} = 22.36$$

$$d(1,8) = \sqrt{(50-90)^2 + (2-5)^2} = 41.23$$

$$b(1) = \min(d(1,2), d(1,4), d(1,5), d(1,7), d(1,8)) = 10.05$$

Similarly, we can calculate $b(i)$ for all other customers.

Step 3: Calculate the Silhouette Coefficient for each customer

Using the formula, we can now calculate $s(i)$ for each customer:

$$s(1) = (10.05 - 25.08) / \max(10.05, 25.08) = -0.60$$

$$s(2) = (-1.00)$$

$$s(3) = (0.09)$$

$$s(4) = (0.52)$$

$$s(5) = (0.56)$$

$$s(6) = (-0.17)$$

$$s(7) = (0.76)$$

$$s(8) = (0.27)$$

$$s(9) = (0.33)$$

$$s(10) = (-0.62)$$

The average Silhouette Coefficient for the entire dataset is the mean of all $s(i)$ values, which in this case is -0.08.

Interpretation:

A negative Silhouette Coefficient generally indicates that a data point may be assigned to the wrong cluster or that there is no clear separation between clusters. In this example, we can see that customers 1, 2, 6, and 10 have negative Silhouette Coefficients, indicating that they may not be well-suited to their assigned clusters. Customers 3, 4, 5, 7, 8, and 9 have positive Silhouette Coefficients, suggesting that they are more likely to be correctly clustered.

Overall, this indicates that our clustering algorithm may need to be refined or that additional features may need to be included in order to obtain better clustering results.

The Silhouette Coefficient is a measure used to evaluate the quality of clustering in data mining. It assesses how well each data point fits within its assigned cluster by considering both the cohesion within the cluster and the separation from neighboring clusters. The coefficient ranges from -1 to 1, where a higher value indicates better-defined and well-separated clusters.

To calculate the Silhouette Coefficient for a data point, we need to compute two values:

Cohesion (a): The average dissimilarity of the data point to other points within the same cluster. A lower value indicates that the data point is tightly clustered with its neighbors.

Separation (b): The average dissimilarity of the data point to points in the nearest neighboring cluster. A higher value suggests that the data point is well-separated from neighboring clusters.

The Silhouette Coefficient (s) is then computed using the following formula:

$$s = (b - a) / \max(a, b)$$

It's important to note that a higher Silhouette Coefficient (closer to 1) indicates better clustering quality, where data points are well-assigned to their clusters and well-separated from neighboring clusters. Conversely, a Silhouette Coefficient close to -1 suggests that data points may be assigned to the wrong clusters, and clusters may overlap or be poorly separated.

Sure! Let's assume we have a dataset of 10 points with two features, and we want to cluster them into two clusters. Here is the example data:

Point	Feature 1	Feature 2
A	1	5
B	3	4
C	4	6
D	7	8
E	8	5
F	9	4
G	10	6
H	11	7
I	13	5
J	15	6

Now, let's say we run a clustering algorithm and it assigns points A, B, C, D, and E to Cluster 1, and points F, G, H, I, and J to Cluster 2. We can calculate the Silhouette Coefficient for each point using the following steps:

1. Calculate the distance between the point and all other points in its own cluster (intra-cluster distance). For example, for point A:

- Distance(A, B) = $\sqrt{(1-3)^2 + (5-4)^2} = 2.236$
- Distance(A, C) = $\sqrt{(1-4)^2 + (5-6)^2} = 2.828$
- Distance(A, D) = $\sqrt{(1-7)^2 + (5-8)^2} = 5.831$
- Distance(A, E) = $\sqrt{(1-8)^2 + (5-5)^2} = 7$

2. Calculate the average intra-cluster distance for the point. For example, for point A:

$$\text{- Average Intra-Cluster Distance}(A) = (2.236 + 2.828 + 5.831 + 7) / 4 = 4.224$$

3. Calculate the distance between the point and all other points in the nearest cluster (inter-cluster distance). For example, for point A:

- Distance(A, F) = $\sqrt{(1-9)^2 + (5-4)^2} = 8.246$
- Distance(A, G) = $\sqrt{(1-10)^2 + (5-6)^2} = 9.055$
- Distance(A, H) = $\sqrt{(1-11)^2 + (5-7)^2} = 10.05$
- Distance(A, I) = $\sqrt{(1-13)^2 + (5-5)^2} = 12$
- Distance(A, J) = $\sqrt{(1-15)^2 + (5-6)^2} = 14.035$

4. Calculate the average inter-cluster distance for the point. For example, for point A:

$$\text{- Average Inter-Cluster Distance}(A) = (8.246 + 9.055 + 10.05 + 12 + 14.035) / 5 = 10.0772$$

5. Calculate the Silhouette Coefficient for the point. For example, for point A:

$$\text{- Silhouette Coefficient}(A) = (10.0772 - 4.224) / \max(10.0772, 4.224) = 0.586$$

We can repeat this process for each point in the dataset to get the Silhouette Coefficient for the entire clustering.

Once we have calculated the Silhouette Coefficient for each point, we can calculate the average Silhouette Coefficient for the entire clustering. This value will give us an indication of how well the points are clustered together.

In our example, we have two clusters and 10 points. Let's assume we have calculated the Silhouette Coefficient for each point and obtained the following values:

| Point | Silhouette Coefficient |

A	0.586
B	0.562
C	0.578
D	0.536
E	0.566
F	0.584
G	0.554
H	0.526
I	0.578
J	0.541

To calculate the average Silhouette Coefficient for the entire clustering, we simply take the mean of all the individual Silhouette Coefficients. In this case, the average Silhouette Coefficient would be:

$$\text{Average Silhouette Coefficient} = (0.586 + 0.562 + 0.578 + 0.536 + 0.566 + 0.584 + 0.554 + 0.526 + 0.578 + 0.541) / 10$$

$$\text{Average Silhouette Coefficient} = 0.565$$

This value indicates that the points are relatively well-clustered together, with an average Silhouette Coefficient of 0.565. However, it is important to note that the quality of the clustering depends heavily on the choice of algorithm and hyperparameters used, as well as on the nature of the dataset itself.

SSE stands for Sum of Squared Errors, also known as the sum of squared distances, within-cluster sum of squares, or inertia. It is a measure used in clustering to evaluate the compactness or tightness of clusters. SSE calculates the sum of the squared Euclidean distances between each data point and its assigned cluster centroid. The lower the SSE value, the better the clustering result.

To calculate SSE for a clustering solution or a specific cluster, follow these steps:

Compute the centroid for each cluster. The centroid is the mean or average of all data points within a cluster.

For each data point within a cluster, calculate the squared Euclidean distance between the data point and its assigned centroid.

Sum up the squared distances for all data points within the cluster. This gives you the SSE for that cluster.

If you want to compare two clusterings or two specific clusters, calculate the SSE for each clustering or cluster individually and compare their SSE values. The clustering or cluster with the lower SSE value is generally considered to be better in terms of compactness and tightness.

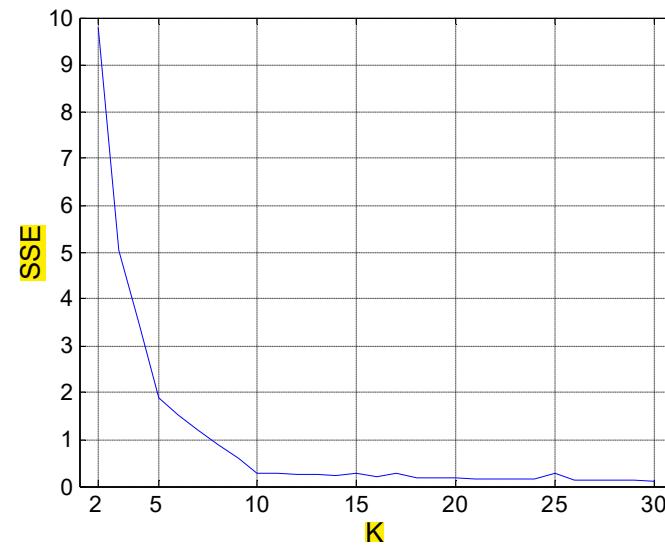
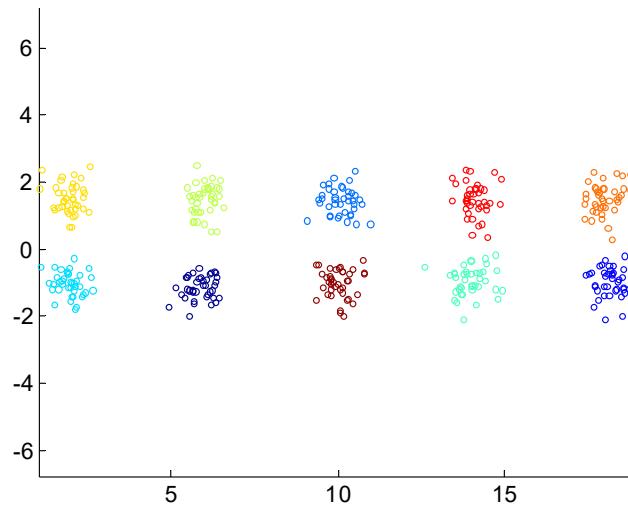
SSE is useful for comparing two clusterings or two specific clusters because it provides a quantitative measure of how well the data points within a cluster are grouped together. By comparing SSE values, you can assess which clustering or cluster results in less scatter or dispersion of data points. Lower SSE indicates that the data points within a cluster are closer to their centroid, suggesting better clustering performance and more cohesive clusters.

However, it's important to note that SSE alone does not provide a complete assessment of clustering quality. It doesn't consider factors such as the number of clusters, the inherent structure of the data, or the interpretability of the clusters. Therefore, it's recommended to use SSE in conjunction with other evaluation metrics and domain knowledge to make informed decisions about clustering results.

Determining the Correct Number of Clusters

- SSE is good for comparing two clusterings or two clusters
- SSE can also be used to estimate the number of clusters

برای اینکه ممکن است روش خوش بندی ما در مبنیم های محلی
گیر کرده باشد به دلیل مقادیر اولیه مرکز ها، این روش را چندبار
تکرار میکنیم و نتیجه ی چندبار تکرار را میگذاریم تصمیمی که باید
بگیریم



Let's consider a simple dataset with five data points in a one-dimensional space:

Data Points: 2, 4, 6, 8, 10

Now, let's assume we have two clusters defined as follows:

Cluster 1: 2, 4

Cluster 2: 6, 8, 10

To calculate the SSE for each cluster, we follow these steps:

Cluster 1:

Compute the centroid by taking the mean of the data points in Cluster 1:

$$\text{Centroid} = (2 + 4) / 2 = 3$$

Calculate the squared Euclidean distance between each data point and the centroid:

$$\text{For data point 2: } (2 - 3)^2 = 1$$

$$\text{For data point 4: } (4 - 3)^2 = 1$$

Sum up the squared distances for all data points in Cluster 1:

$$\text{SSE_Cluster1} = 1 + 1 = 2$$

Cluster 2:

Compute the centroid by taking the mean of the data points in Cluster 2:

$$\text{Centroid} = (6 + 8 + 10) / 3 = 8$$

Calculate the squared Euclidean distance between each data point and the centroid:

$$\text{For data point 6: } (6 - 8)^2 = 4$$

$$\text{For data point 8: } (8 - 8)^2 = 0$$

$$\text{For data point 10: } (10 - 8)^2 = 4$$

Sum up the squared distances for all data points in Cluster 2:

$$\text{SSE_Cluster2} = 4 + 0 + 4 = 8$$

Total SSE for the clustering:

$$\text{SSE} = \text{SSE_Cluster1} + \text{SSE_Cluster2} = 2 + 8 = 10$$

In this example, the SSE value for the clustering is 10. It indicates the sum of squared distances within the clusters, reflecting how well the data points are grouped together. Lower SSE values would indicate tighter and more compact clusters.

SE (Sum of Squared Errors) can be used as a metric to estimate the appropriate number of clusters in a dataset. The idea is to analyze how SSE changes as the number of clusters increases. The point at which adding more clusters does not significantly reduce SSE can be considered a good estimate for the optimal number of clusters.

To illustrate this approach, let's consider a dataset with two-dimensional points:

...

(2, 3), (4, 5), (5, 4), (7, 2), (8, 1), (9, 3)

...

We will calculate the SSE for different numbers of clusters (K) and observe the changes:

1. Start with K=1 (all data points belong to a single cluster).

- Calculate the centroid for the single cluster.
- Calculate the squared Euclidean distance between each point and the centroid.
- Compute the SSE for K=1.

2. Increase K to 2 and repeat the steps.

- Perform clustering with K=2, such as K-means or any other clustering algorithm.
- Calculate the centroid for each cluster.
- Calculate the squared Euclidean distance between each point and its assigned centroid.
- Compute the SSE for K=2.

3. Continue increasing K and calculating SSE until a stopping criterion is met or until reaching the maximum desired number of clusters.

4. Plot a line or curve showing the SSE values against the number of clusters.

- The x-axis represents the number of clusters (K), and the y-axis represents the SSE values.
- SSE values will generally decrease as the number of clusters increases.

5. Analyze the SSE curve and identify the "elbow point" or the point of diminishing returns.

- The elbow point is the value of K at which the SSE reduction significantly slows down.
- It indicates the optimal number of clusters in the dataset.

For example, let's say we calculate the SSE for K=1, K=2, K=3, and K=4, and obtain the following SSE values:

...

K=1: SSE = 28.2

K=2: SSE = 12.4

K=3: SSE = 7.8

K=4: SSE = 5.2

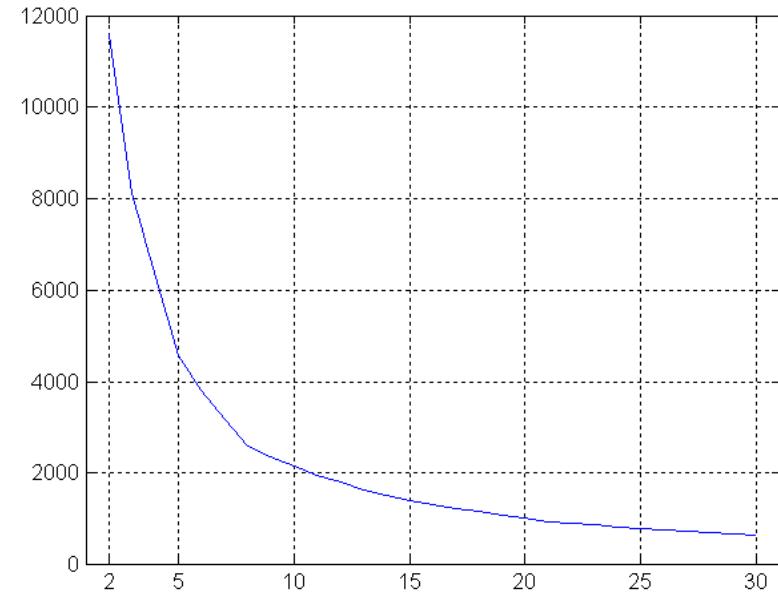
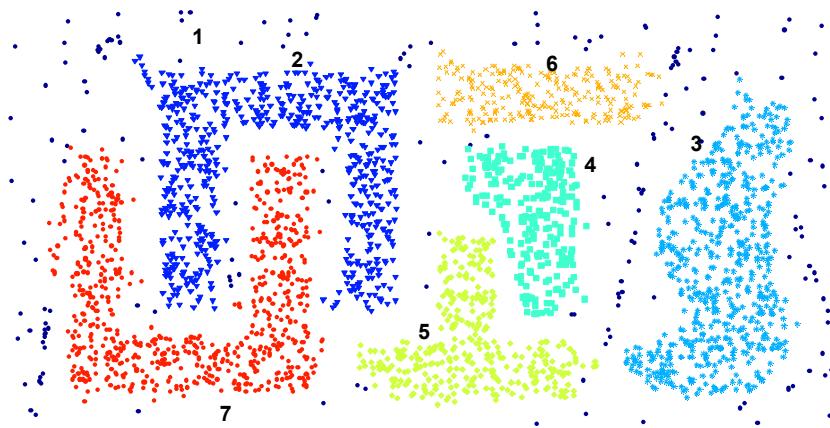
...

Plotting these SSE values against the number of clusters (K) will give us a curve. In this case, the SSE values decrease significantly as we increase K from 1 to 2, and then continue to decrease but at a slower rate from K=2 to K=4. The point of diminishing returns, or the elbow point, appears to be at K=2 or K=3. Hence, based on the SSE analysis, we can estimate that the optimal number of clusters for this dataset is 2 or 3.

It's important to note that the elbow method based on SSE is a heuristic approach and should be used in combination with other evaluation metrics and domain knowledge to determine the most suitable number of clusters for a specific dataset.

Determining the Correct Number of Clusters

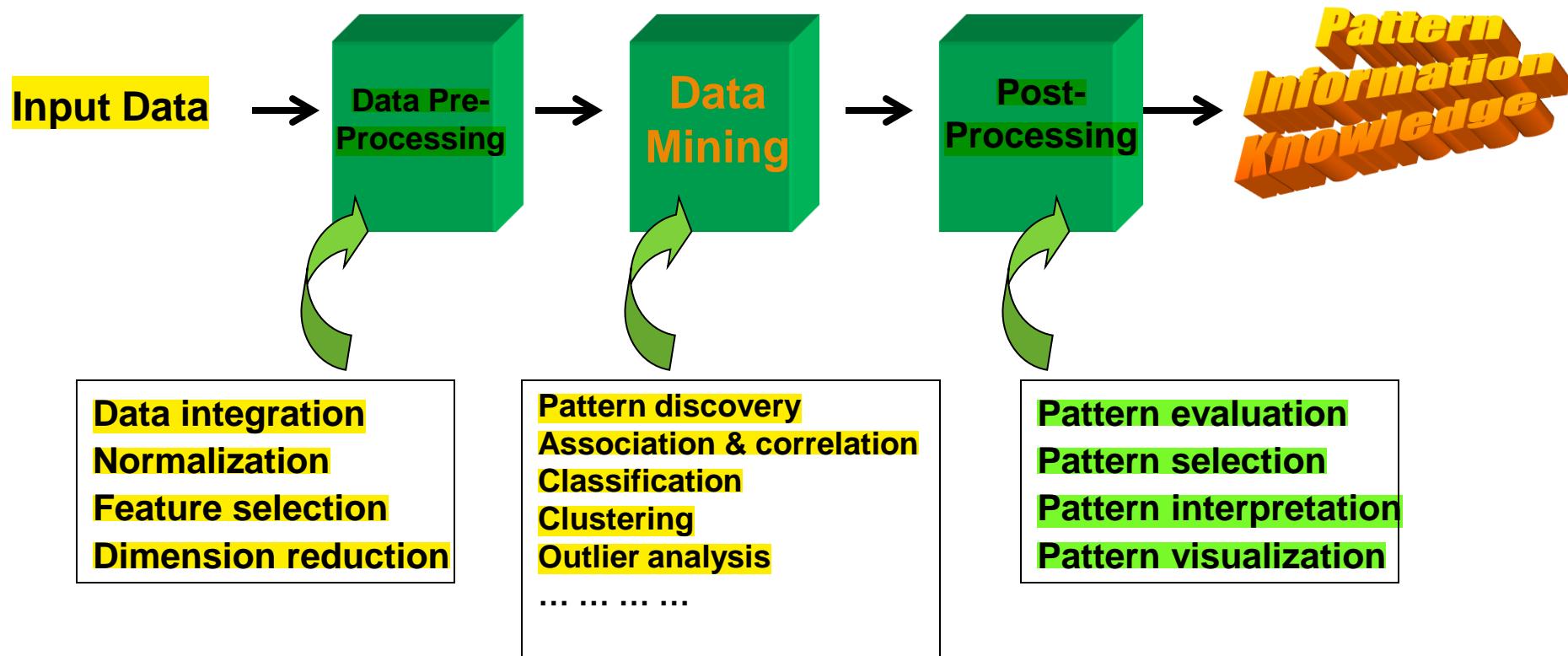
- SSE curve for a more complicated data set



SSE of clusters found using K-means

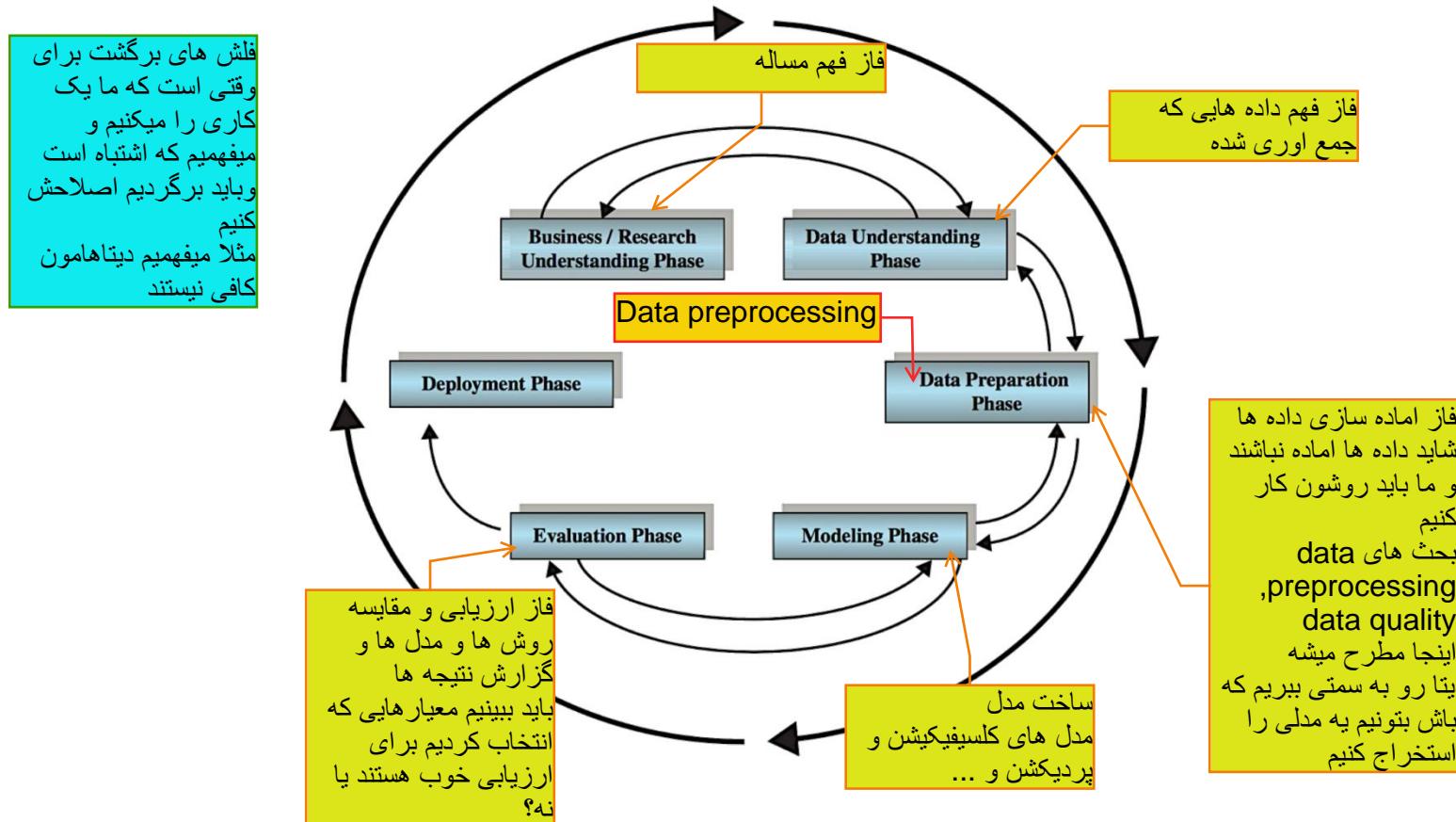
KDD Process: A Typical View from ML and Statistics

- This is a view from typical machine learning and statistics communities



Standard process for data mining

- A cross-industry standard is clearly required, that is industry-neutral, toolneutral, and application-neutral.
- Wikipedia: Polls conducted at one and the same website (KD Nuggets) in 2002, 2004, 2007 and 2014 show that CRISP-DM was the leading methodology used by industry data miners who decided to respond to the survey.
- CRISP-DM: Cross-Industry Standard Process for Data Mining.



CRISP-DM

1. Business/Research Understanding Phase

- Clearly enunciate the project objectives and requirements.
- Translate these goals into the formulation of a data mining problem.
- Prepare a preliminary strategy for achieving these objectives.

2. Data Understanding Phase

- Collect the data.
- Use exploratory data analysis to familiarize yourself with the data, and discover initial insights.
- Evaluate the quality of the data.
- Select interesting subsets that may contain actionable patterns.

3. Data Preparation Phase

- This labor-intensive phase covers all aspects of preparing the final data set, from the initial, raw, dirty data.
- Select the cases and variables appropriate for your analysis.
- Perform transformations on certain variables, if needed.
- Clean the raw data so that it is ready for the modeling tools.

4. Modeling Phase

- Select and apply appropriate modeling techniques.
- Calibrate model settings to optimize results.
- May require looping back to data preparation phase, in order to bring the form of the data into line with data mining technique.

5. Evaluation Phase

- These models must be evaluated for quality and effectiveness.
- Determine whether the model in fact achieves the objectives set for it in Phase 1.
- Finally, come to a decision regarding the use of the data mining results.

6. Deployment Phase

- Example of a simple deployment: Generate a report.
- More complex: Implement a parallel data mining process in another department.
- For businesses, the customer often carries out the deployment based on your model.

1. مرحله تفاهم تجاری/تحقیق

- اهداف و الزامات پروژه را به وضوح بیان کنید.
- این اهداف را به فرمول یک مشکل داده کاوی ترجمه کنید.
- یک استراتژی مقدماتی برای دستیابی به این اهداف تهیه کنید.

2. فاز درک داده ها

- داده ها را جمع آوری کنید.
- از تجزیه و تحلیل داده های اکتشافی برای آشنایی با داده ها و کشف بینش های اولیه استفاده کنید.
- کیفیت داده ها را ارزیابی کنید.
- زیرمجموعه های جالبی را انتخاب کنید که ممکن است حاوی الگوهای عملی باشند.

3. مرحله آماده سازی داده ها

- این مرحله پر زحمت تمام جنبه های آماده سازی مجموعه داده های نهایی را از داده های اولیه، خام و کثیف پوشش می دهد.
- موارد و متغیرهای مناسب برای تحلیل خود را انتخاب کنید.
- در صورت نیاز، تغییر روی متغیرهای خاص انجام دهید.
- داده های خام را پاک کنید تا برای ابزارهای مدل سازی آماده شود.

4. فاز مدل سازی

- تکنیک های مدل سازی مناسب را انتخاب و اعمال کنید.
 - کالیبره کردن تنظیمات مدل برای بهینه سازی نتایج.
 - ممکن است نیاز به بازگشت به مرحله آماده سازی داده باشد تا شکل داده ها را با تکنیک داده کاوی مطابقت دهد.
- این مدل ها باید از نظر کیفیت و اثربخشی ارزیابی شوند.
- تعیین کنید که آیا مدل در واقع به اهداف تعیین شده برای آن در فاز 1 دست می یابد یا خیر.
- در نهایت، در مورد استفاده از نتایج داده کاوی تصمیم بگیرید.

5. مرحله ارزیابی

- مثالی از یک استقرار ساده: یک گزارش ایجاد کنید.
- پیچیده تر: یک فرآیند داده کاوی موازی را در بخش دیگری اجرا کنید.
- برای مشاغل، مشتری اغلب بر اساس مدل شما استقرار را انجام می دهد

6. مرحله استقرار

Outline

- Introduction

گسسته سازی داده ها

- Data Discretization

تجمعیع داده ها

- Data Cleaning

تبديل داده ها

- Data Integration

کاهش داده ها
چرا باید داده هایی که
جمع کردیم را حذف
کنیم؟ چقدر شرط رو باید
حذف کنیم؟

- Data Transformation

- Data Reduction

Why Preprocess the Data?

Data in the real world is dirty

- **incomplete**: lacking attribute values, lacking certain
 - attributes of interest, or containing only aggregate data
 - e.g., occupation=" "
- **noisy**: containing errors or outliers
 - e.g., Salary="-10"
- **inconsistent**: containing discrepancies in codes or names
 - e.g., Age="42" Birthday="03/07/1997"
 - e.g., Was rating "1,2,3", now rating "A, B, C"
 - e.g., discrepancy between duplicate records

یه جاهایی طرف
اطلاعات را پر نکرده

اختلاف و تفاوت

اطلاعات غلط پرشده چون براش مهم نبوده

داده ها با
رکوردهای
قبلی نمیخواهند
فرمت تاریخ
تولدش با سن
ش نمیخوانه

چقدر داده ها را میشناسی؟ چقدر
روی دینتا کلینینگ کار کردی؟

Why is data dirty?

چه اتفاقاتی روی داده ها افتاده؟ چطوری
مدیریتشون کنیم؟

- **Incomplete** data may come from
 - “**Not applicable**” data value when collected
 - Different considerations between the time when the data was collected and when it is analyzed.
 - **Human/hardware/software problems**
- **Noisy** data (**incorrect values**) may come from
 - **Faulty data collection instruments**
 - **Human or computer error** at data entry
 - **Errors in data transmission**
- **Inconsistent** data may come from
 - **Different data sources**
 - **Functional dependency violation** (e.g., modify some linked data)
 - **Duplicate records** also need data cleaning

زمان هایی که داده هامون از چندتا
نبع داره میاد، مثلا از دوربین های
مختلف داره میاد که هر دوربین از
یک سورس داره اطلاعات جمع
میکنه

- داده های ناقص ممکن است از هنگام جمع آوری، مقدار داده «قابل اجرا نیست».
- ملاحظات مختلف بین زمان جمع آوری داده ها و زمان تجزیه و تحلیل آنها.
- مشکلات انسانی/سخت افزاری/نرم افزاری
- داده های پرس و صدا (مقادیر نادرست) ممکن است از آن ناشی شوند
- ابزار گردآوری داده معیوب
- خطای انسانی یا رایانه ای در ورود داده ها
- خطا در انتقال داده ها
- ممکن است داده های متناقض از آن ناشی شود
- منابع داده های مختلف
- نقض وابستگی عملکردی (به عنوان مثال، اصلاح برخی از داده های مرتبط)
- رکوردهای تکراری نیز به پاکسازی داده ها نیاز دارند

Why is preprocessing important?

- No quality data, no quality mining results!
 - Quality decisions must be based on quality data
 - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
 - Data warehouse needs consistent integration of quality data
 - Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse

کیفیت داده یک بحث چندبعدی است.
وقتی میگیم داده مون باکیفیت است از مبنایها و
دیدگاه های مختلفی میتوانیم درباره ی کیفیتش
صحبت کنیم
مثلًا اینکه اطلاعاتی که اومده سمت ما درسته
یا غلطه؟ آیا اطلاعات بروز است؟
سازگاری رکوردها باهمیگر
مثلًا ایا همه رکوردها برای دانشجویان است یا
وسطش یکی اومده رکوردهای کارمندان رو
هم اضافه کرده

بدون داده با کیفیت، بدون نتایج استخراج با کیفیت!
• تصمیمات کیفی باید بر اساس داده های کیفی باشد
• به عنوان مثال، داده های تکراری یا از دست رفته ممکن است باعث آمار نادرست یا حتی گمراه کننده شود.
• انبار داده نیاز به یکپارچه سازی مداوم داده های با کیفیت دارد
• استخراج، تمیز کردن و تبدیل داده ها اکثریت کار ساخت انبار داده را شامل می شود



A multidimensional measure

- Measures for data quality: A multidimensional view
 - Accuracy: correct or wrong, accurate or not
 - Completeness: not recorded, unavailable, ...
 - Consistency: some modified but some not, dangling, ...
 - Timeliness: timely update?
 - Believability: how trustable the data are correct?
 - Interpretability: how easily the data can be understood?

اندازه گیری کیفیت داده ها: نمای چند بعدی

- دقیق: درست یا غلط، دقیق یا نه
- کامل بودن: ثبت نشده، در دسترس نیست، ...
- سازگاری: برخی اصلاح شده اما برخی نه، آویزان، ...
- به موقع بودن: به روز رسانی به موقع؟
- باورپذیری: داده ها چقدر قابل اعتماد هستند؟
- تفسیرپذیری: به چه راحتی داده ها قابل درک هستند؟

Major Tasks in Data Preprocessing

● Data cleaning

- Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies

نلا یه جاهایی سن دانشجویان با عدد گفته شده به
جاهایی با تاریخ تولد گفته شده که بعد عدد سن رو
میشه از روش بدست اورد

● Data integration

- Integration of multiple databases, data cubes, or files

● Data reduction

- Dimensionality reduction/feature reduction

کاهش تعداد

Numerosity reduction

- Data compression

چه زمان هایی باید دیتا را حذف کنیم؟
مثلثاً تعداد اطربیوت ها خیلی زیاد باشه یا
فیچر های بی ارزش داشته باشه
مثلثاً یه مساله ۲۰۰ تا ستوون داره با ۱۰۰۰
تارکورد پس باید یه سری ویژگی ها را کنار
بزاریم

معیار های اندازه گیری توی دیتابیس
ها متفاوت باشه مثلاً یه جاهایی قد را
با مترا گفتن یه جاهایی با سانتی متر

● Data transformation and discretization

- Normalization

تبديل داده ها :
اطربیوت های قد و وزن داریم که مقیاس اینها به هم نمیخوره یکی
کللوگرمه یکی سانتی متره
الگوریتم فقط صفر و یک میشناسه پس باید اینها را نرمال کنیم که
مقادیر بزرگ قد تاثیر بیشتری نگیرن روی مدل ساختن

مثلث حجم زیادی از
تصاویر را داریم که باید
فسرده کنیم و بعد ذخیره
کنیم

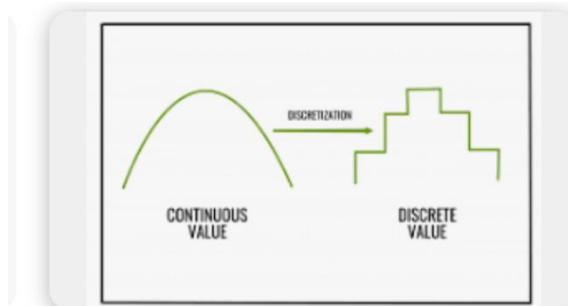
برای فشرده سازی میتوانیم از تکنیک های
لایشن لرنینگ هم استفاده کنیم و هوشمندانه عمل
کنیم مثلای کنیم که تایم سری داریم که دنباله ای از
پیش از اینکه داده ها این مدل را ذخیره کنیم
و روزها را نگه داریم، یه راه اینه که دنباله ای از
تایم سری پر دیکشن روی این داده ها بزنیم و به
ای ذخیره کردن خود داده ها این مدل را ذخیره
کنیم و از این استفاده کنیم مثلای از یک شبکه
عصیبی استفاده کنیم این مدلله یک نماینده از داده
های ماست در جاهایی که حجم داده ها خیلی
زیاد است این کار به درد میخورد



DATA DISCRETIZATION

Discretization

- Discretization: Divide the range of a continuous attribute into intervals
 - Interval labels can then be used to replace actual data values
 - Reduce data size by discretization
 - Supervised vs. unsupervised
 - Split (top-down) vs. merge (bottom-up)
 - Discretization can be performed recursively on an attribute
 - Prepare for further analysis, e.g., classification



GeeksforGeeks

گسسته سازی: محدوده یک ویژگی پیوسته را به فواصل تقسیم کنید

- سپس می توان از برچسب های فاصله برای جایگزینی مقادیر واقعی داده استفاده کرد
- کاهش اندازه داده ها با گسسته سازی
- تحت نظرارت در مقابل بدون نظرارت
- تقسیم (از بالا به پایین) در مقابل ادغام (از پایین به بالا)
- گسسته سازی را می توان به صورت بازگشتی بر روی یک ویژگی انجام داد
- برای تجزیه و تحلیل بیشتر، به عنوان مثال، طبقه بندی آمده شوید

Discretization, in the context of data mining, is the process of transforming continuous numerical variables into discrete intervals or categories. It involves dividing the range of values into distinct bins or intervals.

Discretization is used to simplify data representation, reduce complexity, handle noisy data, and facilitate the application of certain data mining techniques that require categorical or discrete input.

There are several reasons why discretization is used:

Handling categorical algorithms: Many machine learning algorithms are designed to work with categorical variables. By converting continuous variables into discrete ones, we can apply these algorithms and leverage their benefits.

Simplifying data representation: Discretization reduces the complexity of the data by transforming it into a smaller set of distinct categories. It makes the data easier to interpret and analyze.

Dealing with noisy data: Discretization can help handle noise or errors in the data by grouping similar values into the same category. It smooths out the impact of minor variations in the continuous data.

Here's an example to illustrate how discretization can be applied:

Let's consider a dataset of people's ages:

25, 30, 35, 40, 45, 50, 55, 60, 65, 70

Bin 1: 25-40

Bin 2: 40-55

Bin 3: 55-70

By discretizing the age variable, we transformed it into three categories or intervals. We can now treat it as a categorical feature and use it in various data mining tasks. For example, we can analyze the distribution of individuals across different age groups, identify patterns or trends, or apply classification algorithms that require discrete inputs.

The specific method and number of bins used for discretization depend on the characteristics of the data, the problem at hand, and the desired level of granularity. Common discretization techniques include equal-width (equal range) binning, equal-frequency (equal number of data points) binning, and clustering-based binning. It's important to carefully choose the appropriate discretization approach and evaluate its impact on the analysis or modeling task to ensure meaningful results.

Certainly! Let's consider a dataset containing information about customers in an e-commerce platform. One of the variables is "Income," which represents the annual income of each customer. In this case, we may need to discretize the income variable to handle categorical algorithms such as decision trees or association rule mining.

Suppose the original income values are as follows:

...

\$30,000, \$40,000, \$50,000, \$60,000, \$70,000, \$80,000, \$90,000, \$100,000

...

To handle categorical algorithms, we can discretize the income variable into three income groups: low, medium, and high. Here's an example of discretization:

...

Low: \$30,000 - \$50,000

Medium: \$50,000 - \$70,000

High: \$70,000 - \$100,000

...

By discretizing the income variable, we transformed it into three categories or levels. Now, instead of using the exact income values, we can treat income as a categorical feature with the three levels (low, medium, high). This enables us to utilize categorical algorithms that are designed to handle discrete or categorical variables.

For example, we can apply a decision tree algorithm to predict customer behavior based on their income level. The decision tree can have branches based on income groups, allowing us to understand how income influences customer preferences or buying patterns.

Discretization allows us to use categorical algorithms effectively, which may rely on discrete or categorical inputs to make accurate predictions or uncover patterns in the data.

Example: Product prices in an online store Suppose we have a dataset containing the prices of various products: \$10.5, \$15.2, \$20.3, \$25.1, \$30.8, \$35.5, \$40.2, \$45.6, \$50.0, \$55.2

Bin 1: \$10.5-\$25.1

Bin 2: \$25.1-\$40.2

Bin 3: \$40.2-\$55.2

For this example, we have product prices and want to divide them into three bins.

Range: The minimum price is \$10.5, and the maximum price is \$55.2, giving us a range of \$44.7.

Number of bins: We want to create three bins.

Bin width = Range / Number of bins = $44.7 / 3 = 14.9$ (approx.)

Starting from the minimum price, we assign each price to the appropriate bin based on its value. The bins are created by incrementing the lower bound of each bin by the bin width:

Bin 1: \$10.5-\$25.4

Bin 2: \$25.4-\$40.3

Bin 3: \$40.3-\$55.2

Certainly! Let's consider a dataset containing information about the prices of various products in an online marketplace. The "Price" variable represents the continuous prices of the products. To simplify the data representation and make it more interpretable, we can discretize the prices into different price ranges.

Suppose we have the following original price values:

```  
\$10.50, \$15.20, \$18.70, \$25.40, \$30.80, \$35.90, \$42.10, \$50.60  
```

To simplify the data representation, we can discretize the prices into three price ranges: low, medium, and high. Here's an example of discretization:

```  
Low: \$10 - \$20  
Medium: \$20 - \$30  
High: \$30 - \$50  
```

By discretizing the prices, we transformed the continuous variable into three discrete categories or intervals. This simplification makes it easier to understand and analyze the data.

For instance, we can now examine the distribution of products across different price ranges to gain insights into market segmentation. We can visualize the number of products or their characteristics within each price range using bar charts or other graphical representations. This simplification helps in summarizing the data and identifying patterns or trends based on price categories.

Furthermore, this discretization facilitates the comparison of products within the same price range and allows us to make meaningful comparisons and observations about their attributes or customer preferences.

Discretization, in this case, simplifies the data representation, making it more manageable and interpretable, which can aid in data analysis and decision-making processes.

Example: Daily temperatures recorded in a city Consider a dataset containing the daily temperature readings in degrees Celsius:

15.2, 16.5, 18.7, 21.3, 24.8, 28.2, 30.5, 32.1, 29.6, 26.4, 22.9, 19.8

Let's perform equal-width binning to divide the temperature range into four bins:

Bin 1: 15.2-20.0
Bin 2: 20.0-25.0
Bin 3: 25.0-30.0
Bin 4: 30.0-32.1

For this example, we have daily temperature readings and want to divide them into four bins.

Range: The minimum temperature is 15.2, and the maximum temperature is 32.1, giving us a range of 16.9.

Number of bins: We want to create four bins.

Bin width = Range / Number of bins = 16.9 / 4 = 4.225 (approx.)

Starting from the minimum temperature, we assign each temperature to the appropriate bin based on its value. The bins are created by incrementing the lower bound of each bin by the bin width:

Bin 1: 15.2-19.425
Bin 2: 19.425-23.65
Bin 3: 23.65-27.875
Bin 4: 27.875-32.1

Certainly! Let's consider a dataset that represents the temperatures recorded throughout a day. Due to measurement errors or other factors, the dataset may contain some noisy or erroneous temperature readings. In order to handle this noisy data, we can discretize the temperatures into temperature ranges or categories. Suppose we have the following original temperature values:

20.5°C, 21.2°C, 22.0°C, 30.1°C, 40.5°C, 19.8°C, 18.9°C, 50.0°C

To deal with the noisy data, we can discretize the temperatures into three temperature categories: low, moderate, and high. Here's an example of discretization:

Low: < 20°C

Moderate: 20°C - 30°C

High: > 30°C

By discretizing the temperatures, we transform the continuous variable into three discrete categories or intervals. This discretization helps in handling the noisy data by grouping similar values together.

For instance, the temperature reading of 40.5°C might be considered an outlier or noise in the dataset. By discretizing the data, we can categorize it as a high temperature, along with other extreme temperature values. This approach helps to smooth out the impact of minor variations or noisy readings, making the data more robust and resilient to outliers.

Discretization allows us to analyze and interpret the data at a higher level, focusing on temperature categories rather than individual values. This simplification helps in identifying temperature patterns or trends, understanding temperature distributions, and performing subsequent data mining tasks more effectively, such as clustering or classification.

By discretizing and handling the noisy data appropriately, we can mitigate the influence of outliers and obtain more reliable insights from the dataset.

Here are examples of equal-width binning, also known as distance-based binning, applied to different sample datasets:

Example: Age of individuals in a survey Suppose we have a dataset containing the ages of individuals participating in a survey:

20, 25, 30, 32, 35, 38, 40, 42, 45, 50, 55, 60

Let's perform equal-width binning by dividing the age range into three bins:

Bin 1: 20-35

Bin 2: 35-50

Bin 3: 50-60

Equal-width binning involves dividing the range of values into equally sized intervals. Here's how the bins were calculated in the previous examples:

Example: Age of individuals in a survey

To calculate the bins, we need to determine the range of the variable and the desired number of bins. In this case, we have the ages of individuals and want to divide them into three bins.

Range: The minimum age is 20, and the maximum age is 60, giving us a range of 40.

Number of bins: We want to create three bins.

To calculate the bin width, we divide the range by the number of bins:

Bin width = Range / Number of bins = 40 / 3 = 13.33 (approx.)

Starting from the minimum age, we assign each age to the appropriate bin based on its value. The bins are created by incrementing the lower bound of each bin by the bin width:

Bin 1: 20-33.33

Bin 2: 33.33-46.66

Bin 3: 46.66-60

Simple Discretization: Binning

- Equal-width (distance) partitioning
 - Divides the range into N intervals of equal size: uniform grid
 - if A and B are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
 - The most straightforward, but outliers may dominate presentation
 - Skewed data is not handled well

ساده‌ترین، اما پر ت ممکن است بر ارائه غالب باشد
- داده‌های کج به خوبی مدیریت نمی‌شوند

Simple Discretization: Binning

- Equal-depth (frequency) partitioning
 - Divides the range into N intervals, each containing approximately same number of samples
 - Good data scaling
 - Managing categorical attributes can be tricky

Example: Age of individuals in a survey Suppose we have a dataset containing the ages of individuals participating in a survey: 20, 25, 30, 32, 35, 38, 40, 42, 45, 50, 55, 60

To perform equal-depth binning, we need to divide the data into equal-sized bins based on the number of bins desired.

Let's say we want to create three bins:

First, we sort the ages in ascending order:

20, 25, 30, 32, 35, 38, 40, 42, 45, 50, 55, 60

Next, we calculate the bin size, which is the total number of data points divided by the number of bins. In this case, the bin size is 12 (total number of data points) divided by 3 (number of bins), resulting in a bin size of 4.

Starting from the smallest value, we assign data points to bins until each bin contains an equal number of data points. In this example:

Bin 1: Contains the first 4 data points: 20, 25, 30, 32

Bin 2: Contains the next 4 data points: 35, 38, 40, 42

Bin 3: Contains the remaining 4 data points: 45, 50, 55, 60

By performing equal-depth binning, we have divided the age variable into three bins, with each bin containing an equal number of data points. This helps ensure that the data distribution is represented equally across the bins.

Binning Methods for Data Smoothing

- 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

Binning is considered a top-down split in data discretization methods because it involves dividing the data into predefined intervals or bins starting from the highest level and progressively splitting it into smaller subgroups.

Here's a step-by-step explanation of why binning is considered a top-down split:

Initial Range: In the first step, the entire range of the data is considered as a single bin or interval. This represents the highest level or the initial partition of the data.

Splitting Criteria: Next, a splitting criterion is applied to determine how to divide the data into smaller subgroups or bins. This criterion can be based on various factors, such as equal width, equal frequency, or statistical measures.

Splitting Process: Based on the selected criterion, the initial bin is split into two or more sub-bins. This splitting process is repeated recursively until the desired number of bins or specific conditions are met.

Iterative Nature: The process continues iteratively, with each bin being further divided into sub-bins until the desired granularity is achieved or specific stopping criteria are satisfied.

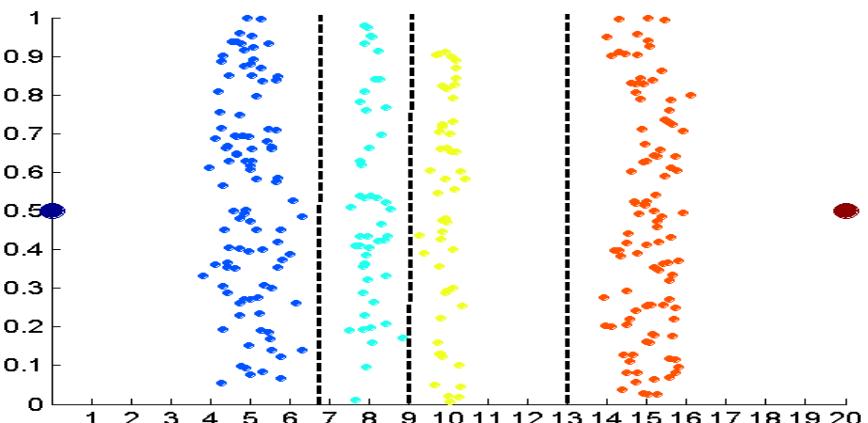
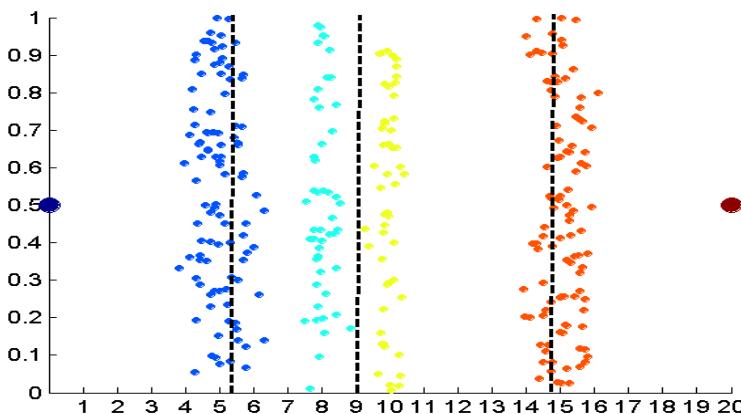
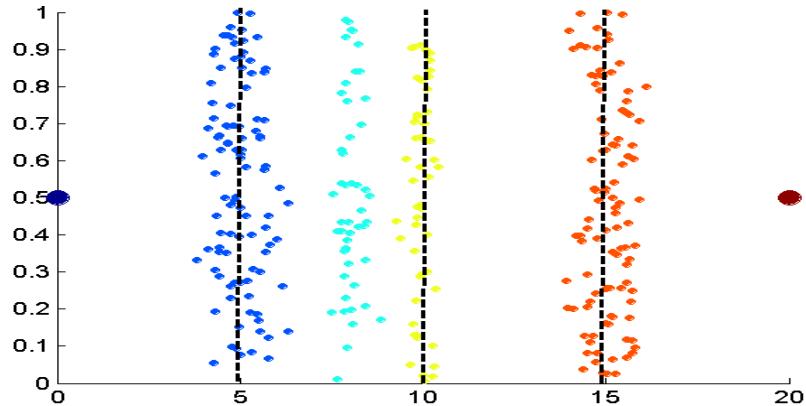
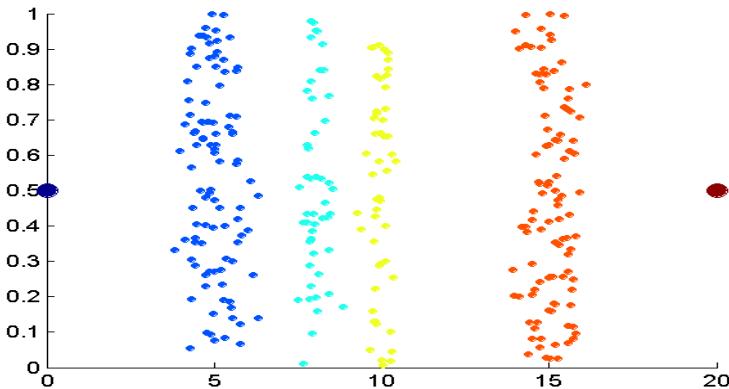
The top-down nature of binning arises from the fact that the initial range of the data is divided into broader bins or intervals, and then these bins are further divided into smaller sub-bins in subsequent iterations. It follows a hierarchical structure, resembling a tree-like representation, where each split creates branches or child bins.

Data Discretization Methods

- Typical methods: All the methods can be applied recursively
 - Binning
 - ◆ Top-down split, unsupervised
 - Histogram analysis
 - ◆ Top-down split, unsupervised
 - Clustering analysis (unsupervised, top-down split or bottom-up merge)
 - Decision-tree analysis (supervised, top-down split)
 - ...

Discretization Without Using Class Labels (Binning vs. Clustering)

Data



Equal frequency (binning)

K-means clustering leads to better results

Discretization by Classification & Correlation Analysis

- Classification (e.g., decision tree analysis)
 - Supervised: Given class labels, e.g., cancerous vs. benign
 - Using *entropy* to determine split point (discretization point)
 - Top-down, recursive split
 - Details to be covered in Chapter 7

Classification can be used as a discretization method in data mining by training a classification model to predict the class labels of data instances and then using the predicted labels as discrete categories. Here's an example to illustrate this process:

Example: Credit Card Fraud Detection

Suppose we have a dataset of credit card transactions with continuous features such as transaction amount, time, and various numerical attributes. The goal is to discretize the transaction amount feature using a classification model.

Dataset Preparation:

Split the dataset into two parts: a training set and a testing set.

Ensure the dataset contains both the continuous feature (transaction amount) and the corresponding class labels (fraudulent or non-fraudulent).

Training a Classification Model:

Use the training set to train a classification model, such as a decision tree, random forest, or logistic regression.

Use the remaining features in the dataset as input features and the class labels (fraudulent or non-fraudulent) as the target variable.

Train the model to learn the patterns and relationships between the features and the class labels.

Predicting Class Labels:

Use the trained classification model to predict the class labels for the instances in the testing set.

The model will assign each transaction a predicted class label, either fraudulent or non-fraudulent, based on the learned patterns.

Discretization using Classification:

Take the predicted class labels obtained from the classification model and treat them as discrete categories for the transaction amount feature.

For example, if the classification model predicts a transaction as fraudulent, assign it to the "fraudulent" category. Otherwise, assign it to the "non-fraudulent" category.

Evaluation and Analysis:

Evaluate the performance of the classification model using appropriate metrics such as accuracy, precision, recall, or F1 score.

Analyze the discretized transaction amount feature in conjunction with other attributes to gain insights or perform further analysis, such as identifying patterns or trends associated with fraudulent transactions.

By using classification as a discretization method, we leverage the predictive power of the classification model to assign discrete categories to continuous variables. This approach allows us to utilize the learned patterns and relationships captured by the model to create meaningful and useful discretized categories.

Why data cleaning?

Importance

“Data cleaning is
one of the three biggest problems in data warehousing”

Ralph Kimball

“Data cleaning is
the number one problem in data warehousing”

DCI survey

Data Cleaning

- Data in the Real World Is Dirty: Lots of potentially incorrect data, e.g., instrument faulty, human or computer error, transmission error
 - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
 - ◆ e.g., *Occupation*=“ ” (missing data)
 - noisy: containing noise, errors, or outliers
 - ◆ e.g., *Salary*=“-10” (an error)
 - inconsistent: containing discrepancies in codes or names, e.g.,
 - ◆ *Age*=“42”, *Birthday*=“03/07/2010”
 - ◆ Was rating “1, 2, 3”, now rating “A, B, C”
 - ◆ discrepancy between duplicate records
 - Intentional (e.g., disguised missing data)
 - ◆ Jan. 1 as everyone’s birthday?

Data cleaning tasks

- Fill in missing values
- Identify outliers and smooth out noisy data
- Correct inconsistent data
- Resolve redundancy caused by data integration

Incomplete (Missing) Data may be due

- equipment malfunction
- inconsistent with other recorded data and thus deleted
- data not entered due to misunderstanding
- certain data may not be considered important at the time of entry
- not register history or changes of the data

Missing data may need to be inferred!!

ادیده گرفتن تاپل: معمولاً زمانی انجام میشود که برچسب کلاس وجود نداشته باشد (هنگام طبقه‌بندی) - زمانی که درصد مقادیر از دست رفته در هر مشخصه بهطور قابل‌توجهی تغییر میکند مؤثر نیست.
مقدار از دست رفته را به صورت دستی پر کنید: خسته کننده + غیر قابل اجراء!
آن را به طور خودکار با
یک ثابت جهانی: به عنوان مثال، "ناشناخته"، یک کلاس جدید؟!
میانگین صفت
میانگین صفت برای همه نمونه های متعلق به یک کلاس: هوشمندتر
محتمل ترین مقدار: میتوان بر استنتاج مانند فرمول بیزی یا درخت تصمیم

خرابی تجهیزات
ناسازگار با سایر داده های ثبت شده و در نتیجه حذف شده است
داده ها به دلیل سوء تفاهم وارد نشده اند
برخی از داده ها ممکن است در زمان ورود مهم تلقی نشوند
 عدم ثبت سابقه یا تغییرات داده ها ممکن است نیاز به استنباط داده های از دست رفته باشد!!

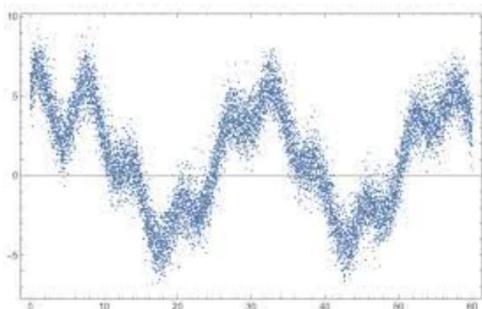
How to Handle Missing Data?

- **Ignore the tuple**: usually done when **class label** is missing (when doing classification)—**not effective** when the % of missing values per attribute **varies considerably**
- **Fill in the missing value manually**: **tedious + infeasible?**
- Fill in it **automatically** with
 - a **global constant** : e.g., “**unknown**”, **a new class?**!
 - the **attribute mean**
 - the **attribute mean for all samples** belonging to the **same class**: **smarter**
 - the most probable value: **inference-based** such as **Bayesian formula** or **decision tree**

Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may be due to
 - faulty data collection instruments
 - data entry problems
 - data transmission problems
 - technology limitation
 - inconsistency in naming convention

نویز: خطای تصادفی یا واریانس در یک متغیر اندازه گیری شده مقدار مشخصه نادرست ممکن است به دلیل ابزارهای گردآوری اطلاعات معیوب مشکلات ورود اطلاعات مشکلات انتقال داده محدودیت تکنولوژی ناهماهنگی در قرارداد نامگذاری



<https://www.javatpoint.com/what-is-noise-in-data-mining>

How to Handle Noisy Data?

- Binning
 - first sort data and partition into (equal-frequency) bins
 - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.

Binning Methods for Data Smoothing

- 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

Smoothing by bin boundaries is a technique used in histogram construction to reduce the noise caused by small fluctuations in data. In this technique, the values of neighboring bins are combined into a single bin to create smoother histograms.

Here's an example to illustrate the concept:

Suppose we have the following data set containing 10 values:

{2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

We want to construct a histogram with 4 bins of equal width. The bin width would be $(11-2)/4=1.75$.

Without smoothing by bin boundaries, the histogram would look like this:

Bin	Frequency
[2, 3.75)	1
[3.75, 5.5)	2
[5.5, 7.25)	2
[7.25, 9)	3
[9, 11]	2

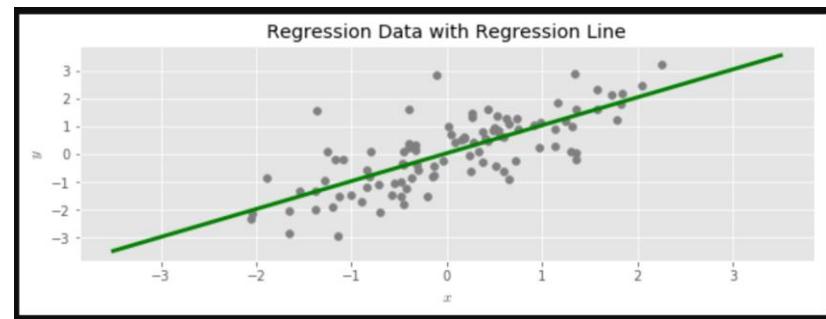
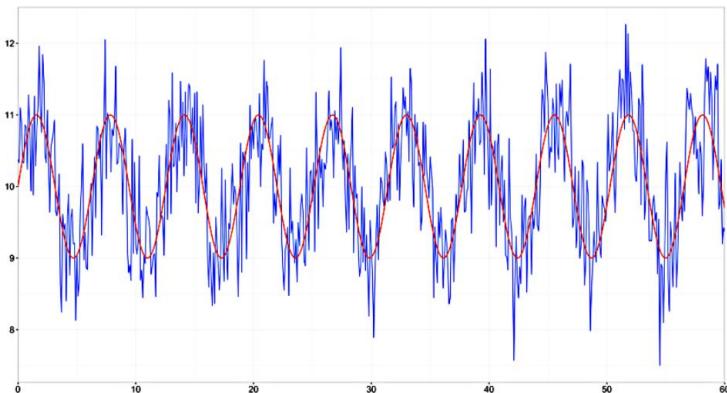
However, if we apply smoothing by bin boundaries, we can combine the first two bins and last two bins to create a smoother histogram. We take the lower boundary of the first bin and the upper boundary of the second bin as the new boundaries for the first bin, and the lower boundary of the fourth bin and the upper boundary of the fifth bin as the new boundaries for the last bin. The resulting histogram would look like this:

Bin	Frequency
[2, 5.5)	3
[5.5, 7.25)	2
(7.25, 9.5]	5

As you can see, the smoothed histogram has fewer bins and is less noisy than the unsmoothed histogram.

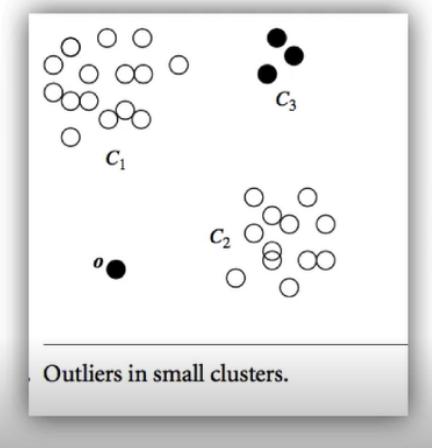
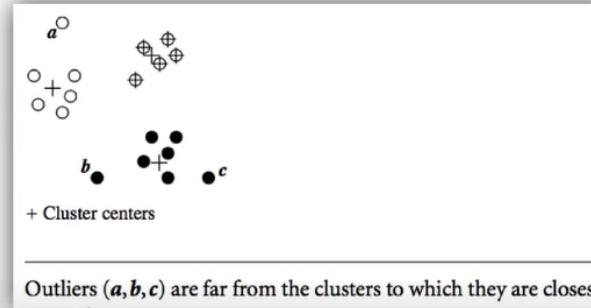
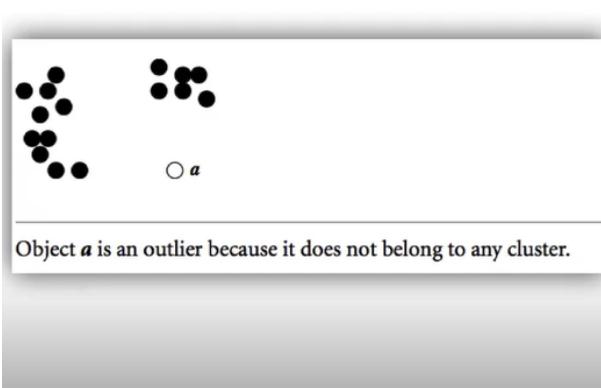
How to Handle Noisy Data?

- Regression
 - smooth by fitting the data into regression functions



How to Handle Noisy Data?

- Clustering
 - detect and remove outliers



How to Handle Noisy Data?

- Binning
 - first sort data and partition into (equal-frequency) bins
 - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Regression
 - smooth by fitting the data into regression functions
- Clustering
 - detect and remove outliers
- Combined computer and human inspection
 - detect suspicious values and check by human (e.g., deal with possible outliers)

یکپارچه سازی داده ها:

- داده هارا از چندین منبع در یک فروشگاه منسجم ترکیب می کند
- یکپارچه سازی طرحواره: به عنوان مثال، $\#-A.cust\text{-}id$ $B.cust$
- ادغام ابرداده از منابع مختلف
- مشکل شناسایی موجودیت:
- شناسایی موجودیت های دنیای واقعی از منابع داده های متعدد،
به عنوان مثال، بیل کلینتون = ویلیام کلینتون
- تشخیص و حل تضاد ارزش داده ها
- برای یک موجودیت دنیای واقعی، مقادیر مشخصه از منابع مختلف متفاوت است
- دلایل احتمالی: نمایش های مختلف، مقیاس های مختلف، به عنوان مثال، متریک در مقابل واحد های بریتانیا

داده های اضافی اغلب هنگام ادغام چندین پایگاه داده رخ می دهد

- شناسایی شی: همان ویژگی یا شیء ممکن است در پایگاه داده رخ می دارد
- داده های قابل مشتق: یک ویژگی ممکن است یک ویژگی "مشتق شده" در جدول دیگر باشد، به عنوان مثال، درآمد سالانه ویژگی های اضافی ممکن است با تجزیه و تحلیل همبستگی و تحلیل کوواریانس قابل تشخیص باشند
- یکپارچه سازی دقیق دادهها از منابع متعدد ممکن است به کاهش/جلوگیری از افزونگیها و ناسازگاریها و بهبود سرعت و کیفیت استخراج کمک کند.

DATA INTEGRATION

Data Integration

- **Data integration:**
 - Combines data from **multiple sources** into a **coherent store**
- **Schema integration:** e.g., $A.cust-id \equiv B.cust\#$
 - Integrate **metadata** from **different sources**
- **Entity identification problem:**
 - Identify real world entities from **multiple data sources**,
e.g., Bill Clinton = William Clinton
- Detecting and resolving **data value conflicts**
 - For the **same real world entity**, attribute values from different sources are different
 - Possible reasons: **different representations**, **different scales**, e.g., metric vs. British units

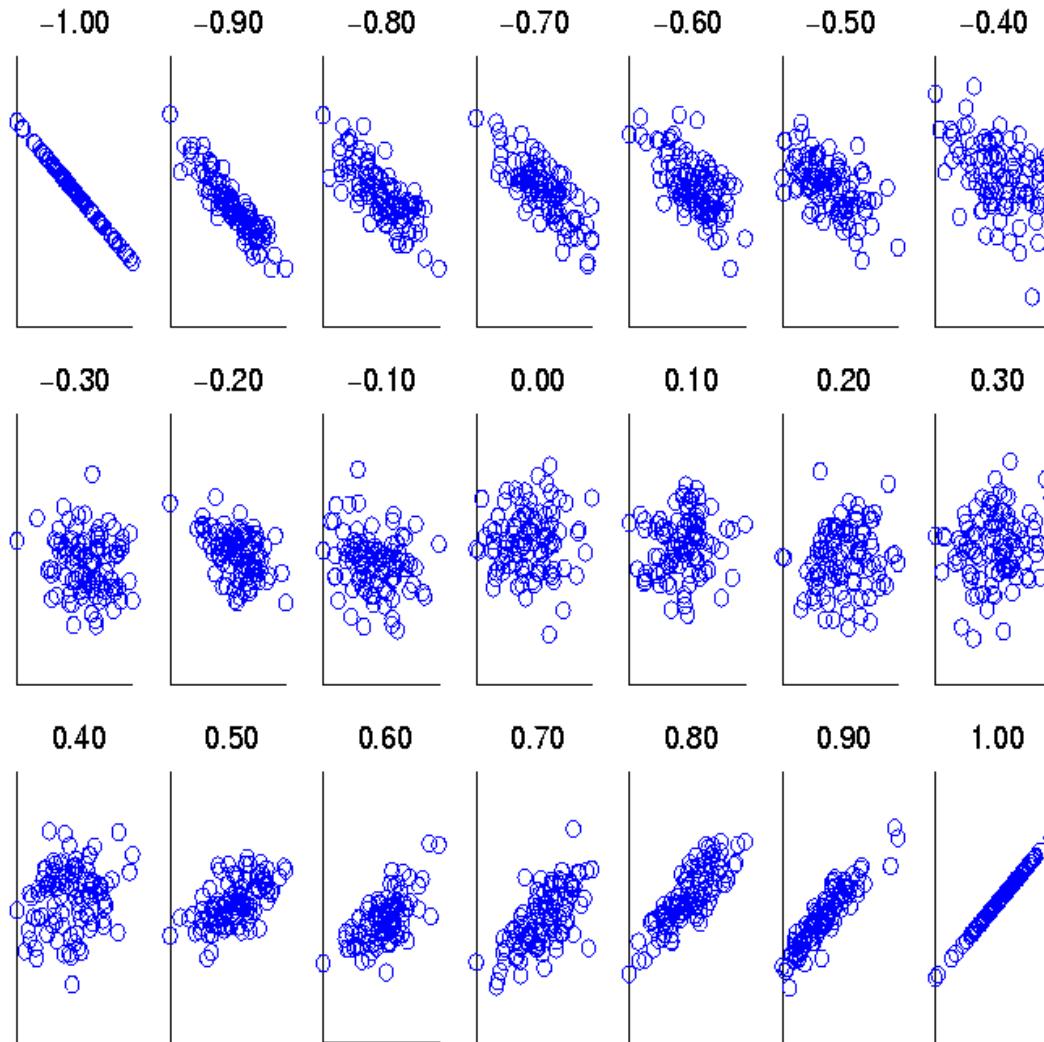
Handling Redundancy

- Redundant data occur often when integration of multiple databases
 - *Object identification*: The same attribute or object may have different names in different databases
 - *Derivable data*: One attribute may be a “derived” attribute in another table, e.g., annual revenue

Redundant attributes may be able to be detected by correlation analysis and covariance analysis

Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Visually Evaluating Correlation



داده های اضافی اغلب هنگام ادغام چندین پایگاه داده رخ می دهد
- شناسایی شی: یک ویژگی یا شیء ممکن است در پایگاه داده های مختلف نام های متفاوتی داشته باشد
- داده های قابل مشتق: یک ویژگی ممکن است یک ویژگی "مشتق شده" در جدول دیگر باشد، به عنوان مثال، در آمد سالانه ویژگی های اضافی ممکن است با تجزیه و تحلیل همبستگی و تحلیل کوواریانس قابل تشخیص باشند
- یکپارچه سازی دقیق دادهها از منابع متعدد ممکن است به کاهش جلوگیری از افزونگیها و ناسازگاریها و بهبود سرعت و کیفیت استخراج کمک کند.

Scatter plots
showing the
similarity from
-1 to 1.

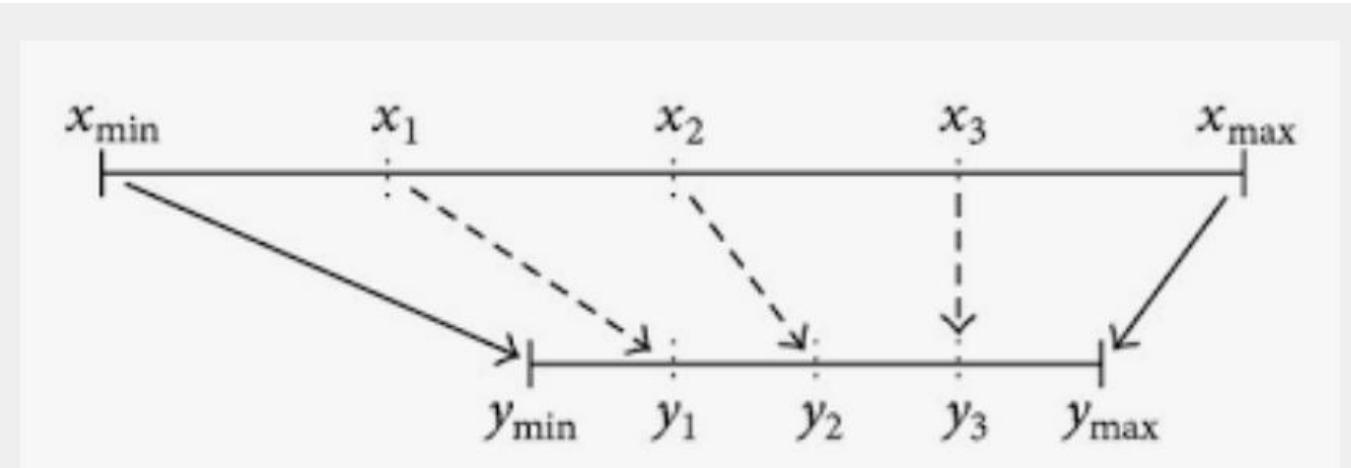


DATA TRANSFORMATION

Data Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values s.t. each old value can be identified with one of the new values
- Methods
 - Smoothing: Remove noise from data
 - Normalization: Scaled to fall within a smaller, specified range
 - ◆ min-max normalization
 - ◆ z-score normalization
 - ◆ normalization by decimal scaling
 - Attribute/feature construction
 - ◆ New attributes constructed from the given ones
 - Aggregation: Summarization, data cube construction
 - Discretization: Concept hierarchy climbing

Normalization



- تابعی که کل مجموعه مقادیر یک ویژگی معین را به مجموعه جدیدی از مقادیر جایگزین S.t نگاشت می کند. هر مقدار قدیمی را می توان با یکی از مقادیر جدید شناسایی کرد
- روش ها
- صاف کردن: حذف نویز از داده ها
 - عادی سازی: مقیاس بندی شده تا در محدوده کوچکتر و مشخصی قرار گیرد
 - نرمال سازی حداقل حداقل
 - عادی سازی امتیاز Z
 - نرمال سازی با مقیاس دهنده
 - ساخت ویژگی/ویژگی ویژگی های جدید ساخته شده از ویژگی های داده شده
 - تجمع: خلاصه سازی، ساخت مکعب داده
 - گسسته سازی: بالا رفتن از سلسله مراتب مفهومی

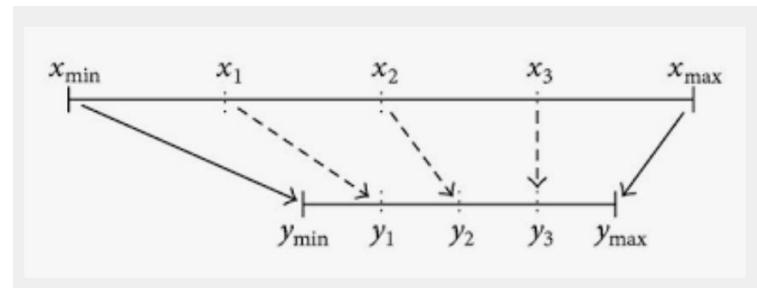
Normalization

- **Min-max normalization:** to $[new_min_A, new_max_A]$

$$v' = \frac{v - min_A}{max_A - min_A} (new_max_A - new_min_A) + new_min_A$$

- Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to

$$\frac{73,600 - 12,000}{98,000 - 12,000} (1.0 - 0) + 0 = 0.716$$



Min-max normalization, also known as feature scaling, is a common technique used in data mining to rescale numerical values within a specific range. This technique helps to prevent one variable from dominating another when calculating distances or similarities between observations.

The formula for min-max normalization is:

$$x_{\text{norm}} = (x - \min(x)) / (\max(x) - \min(x))$$

Where x is the original value, x_{norm} is the normalized value, $\min(x)$ is the minimum value of x in the dataset, and $\max(x)$ is the maximum value of x in the dataset.

To calculate min-max normalization, you can follow these steps:

1. Identify the range of the original dataset by finding the minimum and maximum values for the variable of interest.
2. Subtract the minimum value from each observation to obtain the distance from the minimum.
3. Divide each observation's distance from the minimum by the total range of the dataset to obtain a proportion between 0 and 1.
4. Multiply the resulting proportions by the desired range of the normalized dataset (e.g., 0-1 or -1 to 1) to obtain the normalized values.

For example, let's say we have the following dataset:

[10, 20, 30, 40, 50]

1. The minimum value is 10, and the maximum value is 50, so the range is 40.
2. Subtracting 10 from each observation yields:

[0, 10, 20, 30, 40]

3. Dividing each observation by 40 yields:

[0, 0.25, 0.5, 0.75, 1]

4. Multiplying each observation by the desired range of 0-1 yields:

[0, 0.25, 0.5, 0.75, 1]

Thus, the normalized dataset is:

[0, 0.25, 0.5, 0.75, 1]

This technique can be used for any numerical variable and is especially useful when working with machine learning algorithms that require inputs to be on the same scale.

Sure, here's another example:

Let's say we have a dataset of temperatures in degrees Fahrenheit for a week:

[72, 68, 75, 79, 83, 82, 78]

1. The minimum temperature is 68, and the maximum temperature is 83, so the range is 15.
2. Subtracting 68 from each observation yields:

[4, 0, 7, 11, 15, 14, 10]

3. Dividing each observation by 15 yields:

[0.27, 0, 0.47, 0.73, 1, 0.93, 0.67]

4. Multiplying each observation by the desired range of 0-100 yields:

[27, 0, 47, 73, 100, 93, 67]

Thus, the normalized dataset is:

[27, 0, 47, 73, 100, 93, 67]

This means that the lowest temperature in the original dataset (68°F) is now mapped to 0 in the normalized dataset, and the highest temperature (83°F) is mapped to 100. All other temperatures are scaled proportionally between 0 and 100.

Normalization

- **Z-score normalization** (μ : mean, σ : standard deviation):
 - Ex. Let $\mu = 54,000$, $\sigma = 16,000$. Then

$$v' = \frac{v - \mu_A}{\sigma_A}$$

$$\frac{73,600 - 54,000}{16,000} = 1.225$$

Z-score normalization, also known as standardization, is a data preprocessing technique used in data mining to scale and center numerical features of a dataset.

The formula for Z-score normalization is:

$$z = (x - \text{mean}) / \text{std_dev}$$

where:

- x is the original value of a data point
- mean is the mean value of all data points in the feature
- std_dev is the standard deviation of all data points in the feature
- z is the standardized value of the data point

Here's an example of how to perform Z-score normalization on a dataset containing the weights of apples in grams:

Apple #	Weight (g)
1	150
2	180
3	200
4	160
5	140

Step 1: Calculate the mean and standard deviation of the weights.

$$\text{mean} = (150 + 180 + 200 + 160 + 140) / 5 = 166 \text{ g}$$

$$\text{std_dev} = \sqrt{((150 - 166)^2 + (180 - 166)^2 + (200 - 166)^2 + (160 - 166)^2 + (140 - 166)^2) / 5} = 22.91 \text{ g}$$

Step 2: Apply the formula to each weight to obtain its standardized value.

For Apple 1:

$$z = (150 - 166) / 22.91 = -0.696$$

For Apple 2:

$$z = (180 - 166) / 22.91 = 0.609$$

For Apple 3:

$$z = (200 - 166) / 22.91 = 1.485$$

For Apple 4:

$$z = (160 - 166) / 22.91 = -0.261$$

For Apple 5:

$$z = (140 - 166) / 22.91 = -1.132$$

The resulting standardized dataset would look like this:

Apple #	Weight (g)	Standardized Weight (z-score)
1	150	-0.696
2	180	0.609
3	200	1.485
4	160	-0.261
5	140	-1.132

By applying Z-score normalization, we have scaled and centered the weights of the apples, making them easier to compare and analyze.

Data Reduction Strategies

- **Data reduction:** Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.

کاهش داده ها: یک نمایش کاهش یافته از مجموعه داده به دست آورید که حجم بسیار کمتری دارد اما نتایج تحلیلی یکسان (یا تقریباً یکسان) را تولید می کند.

چرا کاهش داده ها؟ - یک پایگاه داده / انبار داده ممکن است تراپایت داده را ذخیره کند. تجزیه و تحلیل داده های پیچیده ممکن است زمان بسیار زیادی طول بکشد تا در مجموعه داده کامل اجرا شود.

Data Reduction Strategies

- Dimensionality reduction, e.g., remove unimportant attributes
 - ◆ Wavelet transforms
 - ◆ Principal Components Analysis (PCA)
 - ◆ Feature subset selection, feature creation
- Numerosity reduction (some simply call it: Data Reduction)
 - ◆ Regression and Log-Linear Models
 - ◆ Histograms, clustering, sampling
 - ◆ Data cube aggregation
- Data compression

Data Reduction 1: Dimensionality Reduction

نفرین ابعاد

- وقتی ابعاد افزایش می یابد، داده ها به طور فزاینده ای پراکنده می شوند

- چگالی و فاصله بین نقاط، که برای خوش بندی، تجزیه و تحلیل پر ت حیاتی است، کمتر معنادار می شود.

- ترکیب احتمالی، زیر فضاهای به صورت تصاعدی رشد خواهند کرد

● Curse of dimensionality

- When dimensionality increases, data becomes increasingly sparse
- Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
- The possible combinations of subspaces will grow exponentially

The "Curse of dimensionality" is a term used to describe a set of problems that arise when analyzing data with a large number of features or dimensions. As the number of dimensions increases, the amount of data required to represent the space grows exponentially, which can lead to significant computational challenges and overfitting.

For instance, let's say we have a dataset of images with each image having 1000x1000 pixels. If we were to represent each pixel as a separate feature, this would give us a dataset with one million dimensions. The problem with such high dimensional datasets is that they tend to be very sparse, meaning that most of the data points are very far away from any other point in the space.

As an example, consider the task of clustering customers based on their purchase history. If we have just three features - the total amount spent, the number of items purchased, and the time of day - then we can easily visualize the data in three dimensions. However, if we were to add more features, such as the type of item purchased, the brand, the color, etc., then the data would quickly become much more difficult to work with.

This is because the more dimensions we have, the more difficult it becomes to find meaningful patterns in the data. In practice, researchers often try to reduce the number of dimensions by selecting only the most important features or by using techniques like principal component analysis (PCA) to compress the data into a smaller space.

Data Reduction 1: Dimensionality Reduction

کاهش ابعاد

- از نفرین ابعاد بپرهیزید

- کمک به حذف ویژگی های نامربوط و کاهش نویز

- کاهش زمان و فضای مورد نیاز در داده کاوی

- امکان تجسم آسان تر

- Dimensionality reduction

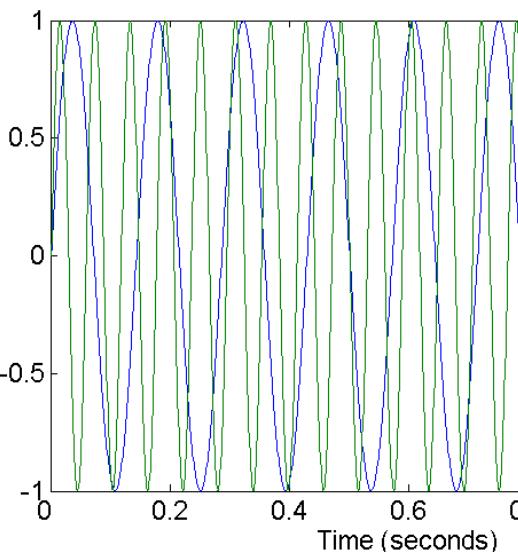
- Avoid the curse of dimensionality
- Help eliminate irrelevant features and reduce noise
- Reduce time and space required in data mining
- Allow easier visualization

Data Reduction 1: Dimensionality Reduction

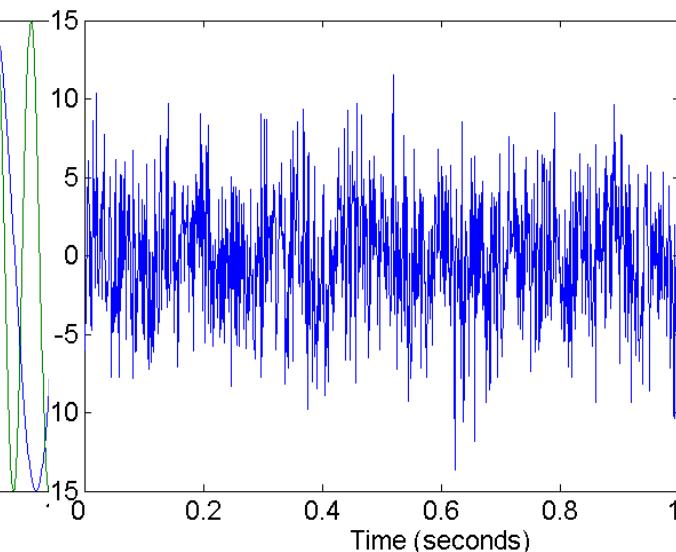
- Dimensionality reduction techniques
 - Wavelet transforms
 - Principal Component Analysis
 - Supervised and nonlinear techniques (e.g., feature selection)

Mapping Data to a New Space

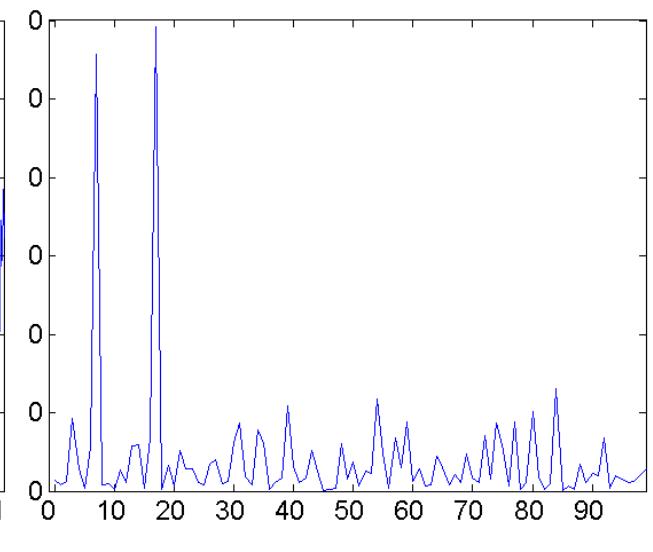
- Fourier transform
- Wavelet transform



Two Sine Waves



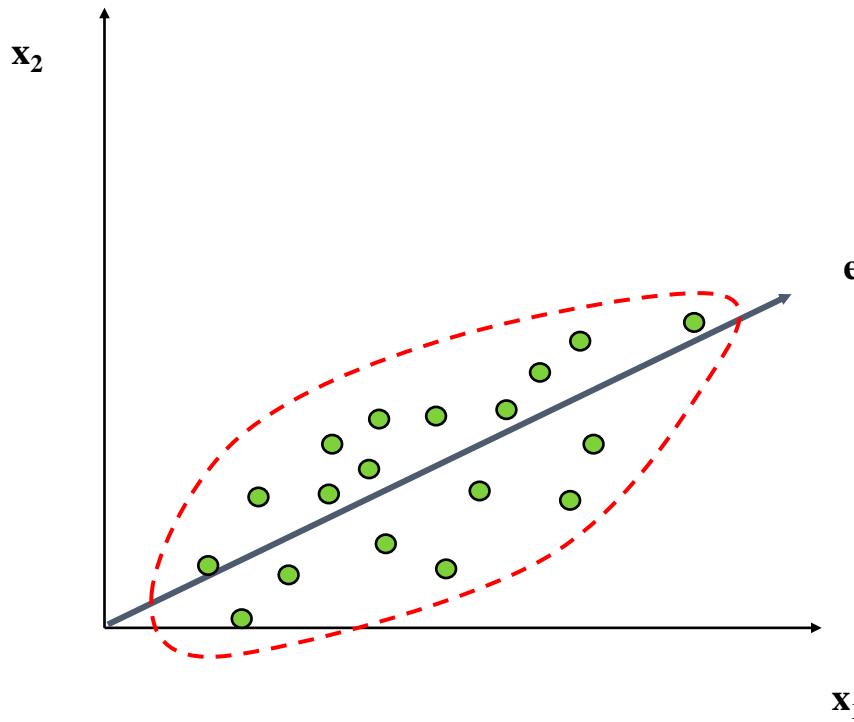
Two Sine Waves + Noise



Frequency

Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction. We find the eigenvectors of the covariance matrix, and these eigenvectors define the new space



Principal Component Analysis (Steps)

- Given N data vectors from n -dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
 - Normalize input data: Each attribute falls within the same range
 - Compute k orthonormal (unit) vectors, i.e., *principal components*
 - Each input data (vector) is a linear combination of the k principal component vectors
 - The principal components are sorted in order of decreasing “significance” or strength
 - Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only

PCA (Principal Component Analysis) is a dimensionality reduction technique that is commonly used in data analysis to compress high-dimensional datasets into a lower number of dimensions while preserving as much of the original variance as possible.

Here are the steps involved in performing PCA:

1. Standardize the data: Before performing PCA, it is important to standardize the data so that each variable has zero mean and unit variance. This is necessary because PCA is sensitive to differences in scale between variables.
2. Calculate the covariance matrix: Once the data has been standardized, we can calculate the covariance matrix. The covariance matrix describes the relationships between the different variables in the dataset.
3. Calculate the eigenvectors and eigenvalues of the covariance matrix: The next step is to calculate the eigenvectors and eigenvalues of the covariance matrix. Eigenvectors are the directions in which the data varies the most, and eigenvalues represent the amount of variance explained by each eigenvector.
4. Select the principal components: The next step is to select the principal components. These are the eigenvectors with the highest eigenvalues, i.e., the directions in which the data varies the most.
5. Project the data onto the new space: Finally, we can project the data into the new space defined by the principal components. This involves multiplying the standardized data by the transpose of the matrix of the selected eigenvectors.

By projecting the data onto the new space defined by the principal components, we can reduce the number of dimensions while still retaining most of the information in the original dataset. This can be useful for visualizing high-dimensional data or for reducing the computational complexity of machine learning algorithms.

Attribute Subset Selection

- Another way to reduce dimensionality of data
- Redundant attributes
 - Duplicate much or all of the information contained in one or more other attributes
 - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
 - Contain no information that is useful for the data mining task at hand
 - E.g., students' ID is often irrelevant to the task of predicting students' GPA

Heuristic Search in Attribute Selection

- There are 2^d possible attribute combinations of d attributes
- Typical heuristic attribute selection methods:
 - Best single attribute under the attribute independence assumption: choose by significance tests
 - Best step-wise feature selection:
 - ◆ The best single-attribute is picked first
 - ◆ Then next best attribute condition to the first
 - Step-wise attribute elimination:
 - ◆ Repeatedly eliminate the worst attribute
 - Best combined attribute selection and elimination
 - Optimal branch and bound:
 - ◆ Use attribute elimination and backtracking

روش های انتخاب ویژگی اکتشافی معمولی:- بهترین ویژگی منفرد تحت فرض استقلال ویژگی: با آزمون های معناداری انتخاب کنید- بهترین انتخاب گام به گام ویژگی: بهترین تک ویژگی ابتدا انتخاب می شودسپس بهترین شرط بعدی را به حالت اول نسبت دهید- حذف صفت گام به گام: بهترین ویژگی را مکرر حذف کنید- بهترین ترکیب و انتخاب ویژگی و حذف- شاخه و کران بهینه: از حذف صفت و عقب نشینی استفاده کنید

Attribute Creation (Feature Generation)

- Create new attributes (features) that can capture the important information in a data set more effectively than the original ones
- Three general methodologies
 - Attribute extraction
 - ◆ Domain-specific
 - Mapping data to new space (see: data reduction)
 - ◆ E.g., Fourier transformation, wavelet transformation, manifold approaches (not covered)
 - Attribute construction
 - ◆ Combining features (some patterns in Chapter 7)
 - ◆ Data discretization

ایجاد ویژگیهای جدید (ویژگیها) که میتوانند اطلاعات مهم در یک مجموعه داده را به طور مؤثرتری نسبت به موارد اصلی ضبط کنند.
سه روش کلی استخراج صفت دامنه خاص - نگاشت داده ها به فضای جدید (نگاه کنید به: کاهش داده ها) به عنوان مثال، تبدیل فوریه، تبدیل موجک، رویکردهای چندگانه (پوشش داده نشده) - ساخت صفت ترکیب ویژگی ها گسسته سازی داده ها

Summary

- **Data quality:** accuracy, completeness, consistency, timeliness, believability, interpretability
- **Data cleaning:** e.g. missing/noisy values, outliers
- **Data integration** from multiple sources:
 - Entity identification problem
 - Remove redundancies
 - Detect inconsistencies
- **Data reduction**
 - Dimensionality reduction
 - Numerosity reduction
 - Data compression
- **Data transformation and data discretization**
 - Normalization
 - Concept hierarchy generation

What Is Frequent Pattern Analysis?

الگوی مکرر: یک الگو (مجموعه ای از آیتم ها، دنباله ها، زیرساخت ها، و غیره) که اغلب در یک مجموعه داده رخ می دهد.

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently **in a data set**
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent itemsets** and **association rule mining**
- **Motivation**: Finding **inherent regularities** in data
 - What products were often **purchased together**?— Beer and diapers?!
 - What are the **subsequent purchases** after **buying a PC**?
 - What **kinds of DNA** are **sensitive** to this new drug?
 - Can we **automatically classify web documents**?
- **Applications**
 - **Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log** (click stream) analysis, and **DNA sequence analysis**.



انگیزه: یافتن قاعده مندی های ذاتی در داده ها- چه محصولاتی اغلب با هم خریداری می شد؟- آبجو و پوشک؟!- خریدهای بعد از خرید کامپیوتر چیست؟- چه نوع DNA به این داروی جدید حساس است؟- آیا می توانیم اسناد وب را به طور خودکار طبقه بندی کنیم؟ برنامه های کاربردی- تجزیه و تحلیل داده های سبد، بازاریابی متقابل، طراحی کاتالوگ، تجزیه و تحلیل کمپین فروش، تجزیه و تحلیل لاغ و ب (جريان کلیک) و تجزیه و تحلیل توالی DNA

ما یک متنی داریم و میخاییم ببینیم
چه کلمات پر تکراری توی متن
وجود داره؟ چه دنباله ای از کلمات
پشت سر هم خیلی تکرار شده؟

چه کالاهایی باهم تکرار شدند؟ یا توی
یک گرافی میخاییم نودها با ویژگی
های خاصی رو شناسایی کنیم
مثلًا ترتیب و تکرار ژن های مختلف
در افراد باعث ویژگی های متفاوت شون
میشه

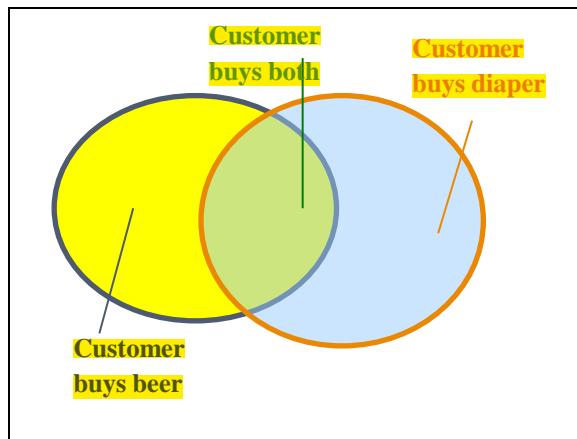
Why Is Freq. Pattern Mining Important?

- Freq. pattern: An intrinsic and **important** property of datasets
- Foundation for **many essential** data mining tasks
 - Association, correlation, and causality analysis
 - Sequential, structural (e.g., sub-graph) patterns
 - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
 - Classification: discriminative, frequent pattern analysis
 - Cluster analysis: frequent pattern-based clustering
 - Data warehousing: iceberg cube and cube-gradient
 - Semantic data compression
 - Broad applications

فرکانس الگو: ویژگی ذاتی و مهم مجموعه داده ها
پایه و اساس بسیاری از وظایف داده کاوی ضروری
- تحلیل ارتباط، همبستگی و علیت
- الگوهای متوالی و ساختاری (به عنوان مثال، نمودار فرعی).
- طبقه بندی: تجزیه و تحلیل الگوی متمایز و مکرر
- تجزیه و تحلیل خوش ای: خوش بندی مکرر مبتنی بر الگو
- انبار داده: مکعب کوه پیچ و مکعب- گردیان
- فشرده سازی داده های معنایی: فاسیکل ها
- کاربردهای گسترده

Basic Concepts: Frequent Patterns

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



itemset: A set of one or more items

k-itemset $X = \{x_1, \dots, x_k\}$

یعنی مجموعه‌ی ما
کتا عضو داره

(absolute) support, or, support count of X:

Frequency or occurrence of an itemset X

تعداد تکرار یه
مجموعه در کل لیست
ایتم هامون

(relative) support, s,

is the fraction of transactions that contains X (i.e.,
the probability that a transaction contains X)

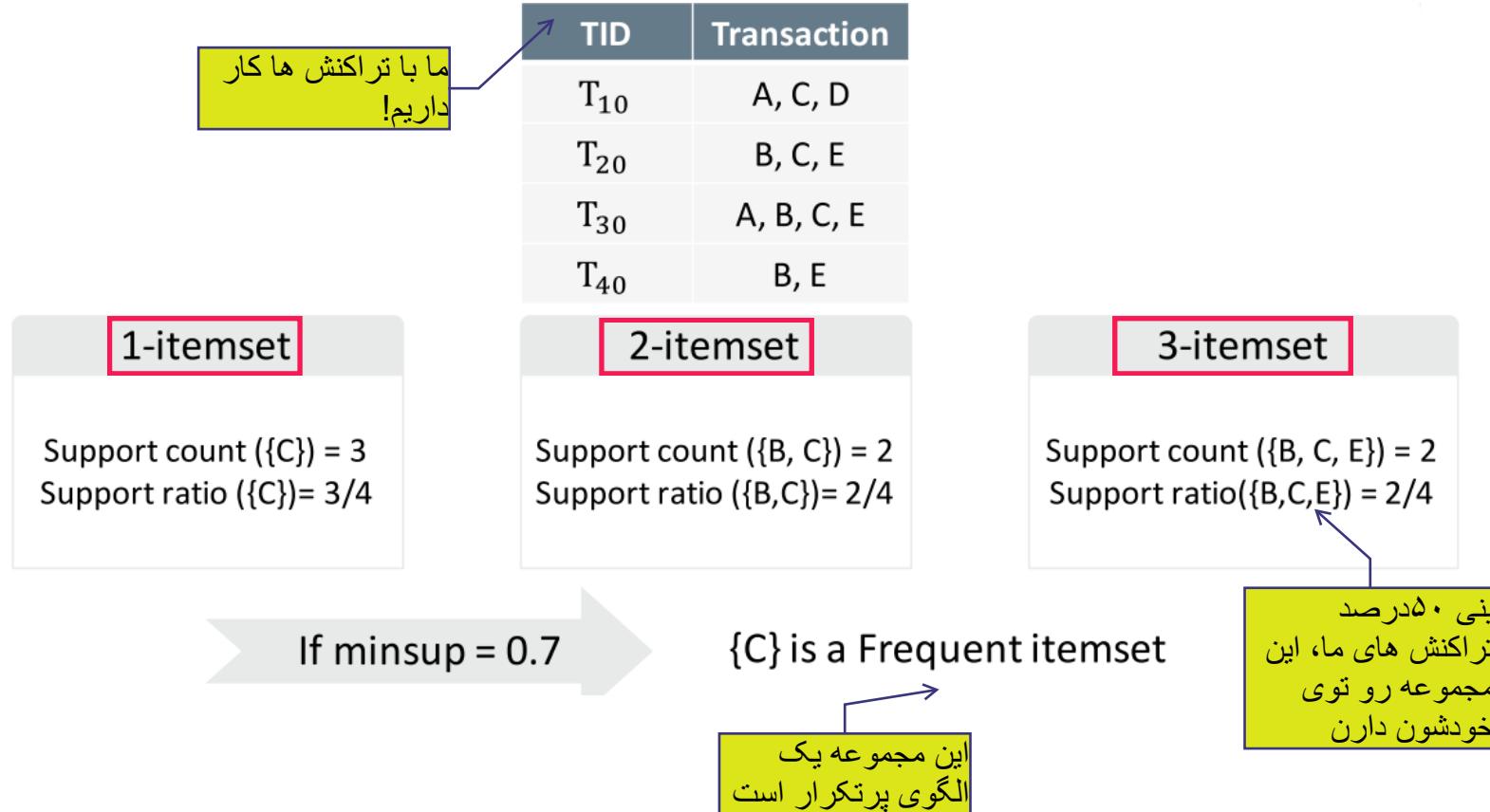
مینیمم ساپورت چقدر؟
مثلا میگیم الگوهای تکراری رو پیدا
کن که حداقل دوبار رخ دادن

An itemset X is **frequent**

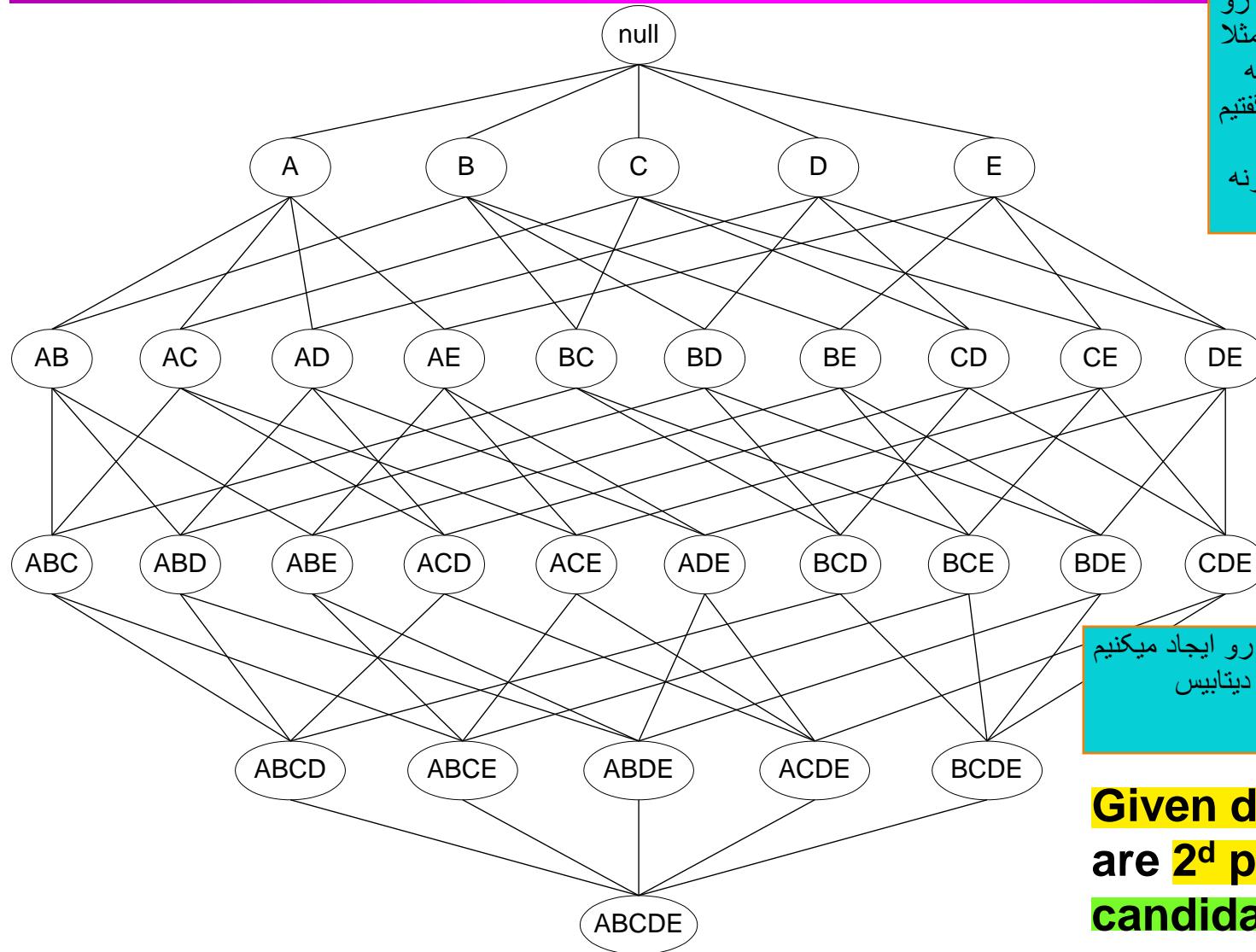
if X's **support** is no less than a **minsup** threshold

مثلا خرید شیر و ماست به طور
۰ ادرصد موقع در خرید ها باهم
خریده شدن که بهش ساپورت
نسبی میگیم

Basic Concepts: Frequent Patterns



Frequent Itemset Generation



ما اگه بخایم الگوهای پر تکرار رو برای ۴ تا ایتم ببینیم باید همچین درختی رو رسی کنیم چون نگفته مثلا الگوهای پر تکرار یه دونه یا دو تایی یا ۳ تایی و گفتیم الگوهای پر تکراری که میتونه اتفاق بیفته، که میتونه از ۱ تایی باشه تا ۴ تایی

رابطه‌ی نعداد الگوهای پر تکرار با تعداد ایتم‌ها نمایی است!

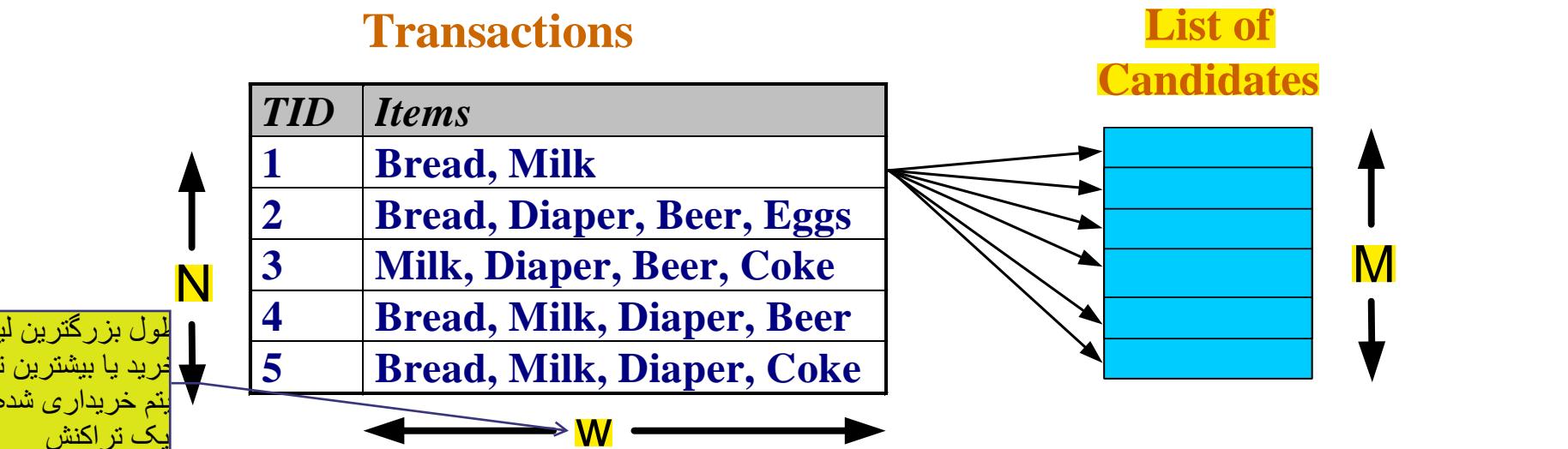
ول همه‌ی الگوهای پر تکرار رو ایجاد میکنیم بعد تعداد تکرارشون رو توی دیتابیس میشماریم

Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:

- Each itemset in the lattice is a **candidate** frequent itemset
- Count the support of each candidate by scanning the database



- Match each transaction against every candidate
- Complexity $\sim O(NMw)$ => Expensive since $M = 2^d$!!!

نمایی میشه

Illustrating Apriori Principle

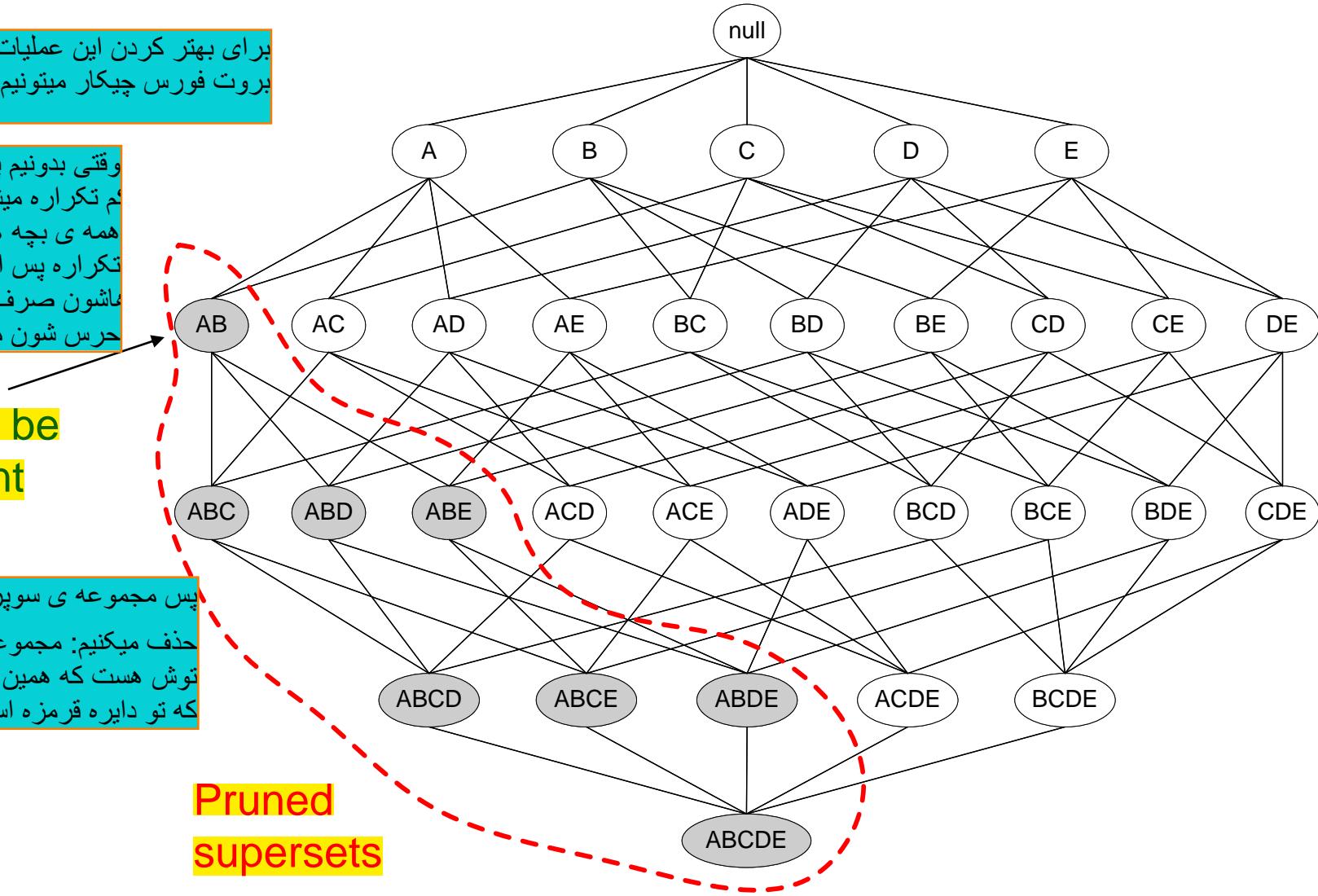
برای بهتر کردن این عملیات سرچ و بروت فورس چیکار میتوانیم بکنیم؟

وقتی بدونیم پدر یه مجموعه کم تکراره میشه نتیجه گرفت همه ی بچه هاش هم، کم تکراره پس از بررسی بچه ماشون صرف نظر میکنیم و حرس شون میکنیم

Found to be
Infrequent

پس مجموعه ی سوپرست AB را حذف میکنیم: مجموعه ای که توش هست که همین نودهایی میشه که تو دایره فرمزه است

Pruned
supersets



The Apriori principle is a key concept in frequent pattern analysis, which is used to identify the most commonly occurring patterns in a dataset. The principle states that if an itemset (a group of items) is frequent, then all of its subsets must also be frequent.

In other words, if a set of items occurs frequently in a dataset, then any subset of that set must also occur frequently. This principle allows us to reduce the search space when looking for frequent itemsets in large datasets, by eliminating candidate itemsets that cannot be frequent based on their subsets.

The Apriori algorithm uses this principle to generate candidate itemsets of increasing size, by joining smaller frequent itemsets together. It then scans the dataset to count the frequency of each candidate itemset, and eliminates those that do not meet the minimum support threshold.

By using the Apriori principle to prune the search space, the Apriori algorithm is able to efficiently identify the most frequent itemsets in a dataset, even when the number of possible itemsets is very large.

Sure, here are the steps of the Apriori algorithm in frequent pattern analysis:

1. Set the minimum support threshold: The minimum support threshold is the minimum number of occurrences that an itemset must have to be considered "frequent." This value is typically set by the user based on domain knowledge or trial and error.
2. Generate frequent 1-itemsets: Scan the entire dataset to count the number of occurrences of each item. Keep only those items that occur at least as many times as the minimum support threshold. These frequent 1-itemsets will serve as the starting point for generating larger itemsets.
3. Generate candidate k-itemsets: Join frequent (k-1)-itemsets together to form candidate k-itemsets. For example, if {A, B} and {B, C} are frequent 2-itemsets, then we can join them together to form a candidate 3-itemset {A, B, C}. However, not all candidate k-itemsets will necessarily be frequent.
4. Prune infrequent candidate k-itemsets: Scan the entire dataset again to count the number of occurrences of each candidate k-itemset. Keep only those itemsets that occur at least as many times as the minimum support threshold. These frequent k-itemsets will be used to generate candidate (k+1)-itemsets in the next iteration.
5. Terminate when no more frequent itemsets can be generated: Repeat steps 3 and 4 to generate candidate (k+1)-itemsets and prune infrequent itemsets until no more frequent itemsets can be generated. At this point, we have identified all of the frequent itemsets in the dataset.
6. Optionally, generate association rules: Once the frequent itemsets have been identified, we can generate association rules that describe relationships between different itemsets. These rules can be used to make predictions or gain insights into the underlying data.

Sure, let's consider a simple example dataset consisting of transactions at a grocery store:

| Transaction | Items |

	Items
1	A, B
2	A, C
3	B, C
4	A, B, C
5	B, D
6	A, C, D

We will use the Apriori algorithm to identify frequent itemsets in this dataset, with a minimum support threshold of 2.

Step 1: Set the minimum support threshold

Our minimum support threshold is set to 2, meaning that an itemset must occur in at least 2 transactions to be considered frequent.

Step 2: Generate frequent 1-itemsets

We scan the entire dataset and count the number of occurrences of each item. Items that appear in at least 2 transactions are considered frequent.

Frequent 1-itemsets:

{A}, {B}, {C}, {D}

Step 3: Generate candidate 2-itemsets

We join together the frequent 1-itemsets to generate candidate 2-itemsets.

Candidate 2-itemsets:

{A, B}, {A, C}, {B, C}, {B, D}, {C, D}

Step 4: Prune infrequent candidate 2-itemsets

We count the number of occurrences of each candidate 2-itemset in the dataset, and keep only those that occur in at least 2 transactions.

Frequent 2-itemsets:

{A, B}, {A, C}, {B, C}

Step 3: Generate candidate 3-itemsets

We join together the frequent 2-itemsets to generate candidate 3-itemsets.

Candidate 3-itemsets:

{A, B, C}

Step 4: Prune infrequent candidate 3-itemsets

We count the number of occurrences of each candidate 3-itemset in the dataset, and keep only those that occur in at least 2 transactions.

Frequent 3-itemsets:

{A, B, C}

Step 5: Terminate when no more frequent itemsets can be generated

Since there are no more candidate itemsets to generate, we have identified all of the frequent itemsets in the dataset.

Step 6: Generate association rules (optional)

We can use the frequent itemsets to generate association rules that describe relationships between different itemsets. For example, we could generate the rule {A, B} \rightarrow {C}, which means that if a customer buys items A and B, they are likely to also buy item C.

Overall, the Apriori algorithm has efficiently identified the most frequent itemsets in our dataset, without having to consider all possible combinations of items.

By using the Apriori principle to prune the search space, the algorithm is able to quickly identify the most important patterns in the data.

Sure, let's consider another example dataset consisting of transactions at a clothing store:

Transaction	Items
1	Shirt, Hat, Belt
2	Shirt, Pants, Shoes
3	Pants, Shoes, Socks
4	Shirt, Pants, Belt, Shoes
5	Hat, Belt, Scarf
6	Shirt, Hat, Shoes

We will use the Apriori algorithm to identify frequent itemsets in this dataset, with a minimum support threshold of 2.

Step 1: Set the minimum support threshold

Our minimum support threshold is set to 2, meaning that an itemset must occur in at least 2 transactions to be considered frequent.

Step 2: Generate frequent 1-itemsets

We scan the entire dataset and count the number of occurrences of each item. Items that appear in at least 2 transactions are considered frequent.

Frequent 1-itemsets:

{Shirt}, {Pants}, {Shoes}, {Belt}, {Hat}

Step 3: Generate candidate 2-itemsets

We join together the frequent 1-itemsets to generate candidate 2-itemsets.

Candidate 2-itemsets:

{Shirt, Pants}, {Shirt, Shoes}, {Shirt, Belt}, {Shirt, Hat}, {Pants, Shoes}, {Pants, Belt}, {Shoes, Belt}, {Hat, Belt}, {Hat, Shoes}

Step 4: Prune infrequent candidate 2-itemsets

We count the number of occurrences of each candidate 2-itemset in the dataset, and keep only those that occur in at least 2 transactions.

Frequent 2-itemsets:

{Shirt, Pants}, {Shirt, Shoes}, {Shirt, Belt}, {Pants, Shoes}, {Pants, Belt}, {Shoes, Belt}, {Hat, Belt}, {Hat, Shoes}

Step 3: Generate candidate 3-itemsets

We join together the frequent 2-itemsets to generate candidate 3-itemsets.

Candidate 3-itemsets:

{Shirt, Pants, Shoes}, {Shirt, Pants, Belt}, {Shirt, Shoes, Belt}, {Pants, Shoes, Belt}, {Hat, Belt, Shoes}

Step 4: Prune infrequent candidate 3-itemsets

We count the number of occurrences of each candidate 3-itemset in the dataset, and keep only those that occur in at least 2 transactions.

Frequent 3-itemsets:

{Shirt, Pants, Shoes}, {Shirt, Pants, Belt}, {Shirt, Shoes, Belt}, {Pants, Shoes, Belt}

Step 5: Terminate when no more frequent itemsets can be generated

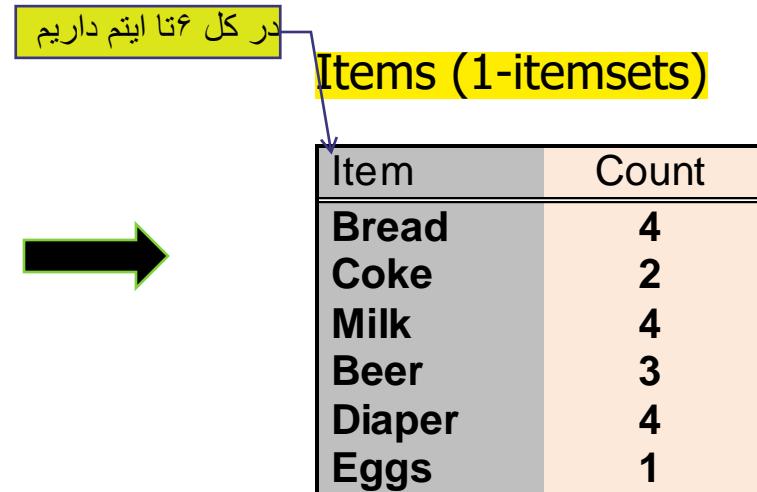
Since there are no more candidate itemsets to generate, we have identified all of the frequent itemsets in the dataset.

Step 6: Generate association rules (optional)

We can use the frequent itemsets to generate association rules that describe relationships between different itemsets. For example, we could generate the rule {Shirt, Pants} \rightarrow {Shoes}, which means that if a customer buys both a shirt and pants, they are likely to also buy shoes. Overall, the Apriori algorithm has efficiently identified the most frequent itemsets in our dataset, without having to consider all possible combinations of items. By using the Apriori principle to prune the search space, the algorithm is able to quickly identify the most important patterns in the data.

Illustrating Apriori Principle

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



Minimum Support = 3

چه ایتم هایی حذف
میشوند؟ اونایی که
تعداد تکرارشون
کمتر از ۳ باشه: تخم
مرغ و نوشابه

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3 \\ 6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

ساخت ایتم های یک تایی: انتخاب یک از ۶ : ۶
ساخت ایتم های دوتایی : انتخاب ۲ از ۶ = ۱۵
به همین ترتیب ...

$$C(n, r) = \frac{n!}{r!(n - r)!}$$

انتخاب ۱ از ۶

انتخاب ۲ از ۴

انتخاب ۳ از ۴

Illustrating Apriori Principle

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



Items (1-itemsets)

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

Illustrating Apriori Principle

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

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)



Itemset
{Bread, Milk}
{Bread, Beer }
{Bread,Diaper}
{Beer, Milk}
{Diaper, Milk}
{Beer,Diaper}

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

Illustrating Apriori Principle

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

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)



Itemset	Count
{Bread,Milk}	3
{Beer, Bread}	2
{Bread,Diaper}	3
{Beer,Milk}	2
{Diaper,Milk}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

Illustrating Apriori Principle

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

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)



Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$



Triplets (3-itemsets)

Itemset	Count
{ Beer, Diaper, Milk}	2
{ Beer,Bread, Diaper}	2
{Bread, Diaper, Milk}	2
{Beer, Bread, Milk}	1

Illustrating Apriori Principle

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

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)



Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

$$\textcolor{red}{6 + 6 + 1 = 13}$$



Triplets (3-itemsets)

Itemset	Count
{ Beer, Diaper, Milk }	2
{ Beer,Bread, Diaper }	2
{Bread, Diaper, Milk}	2
{Beer, Bread, Milk}	1

Apriori Algorithm

- F_k : frequent k-itemsets
 - L_k : candidate k-itemsets
- Algorithm
 - Let $k=1$
 - Generate $F_1 = \{\text{frequent 1-itemsets}\}$
 - Repeat until F_k is empty
 1. **Candidate Generation:** Generate L_{k+1} from F_k
 2. **Candidate Pruning:** Prune candidate itemsets in L_{k+1} containing subsets of length k that are infrequent
 3. **Support Counting:** Count the support of each candidate in L_{k+1} by scanning the DB
 4. **Candidate Elimination:** Eliminate candidates in L_{k+1} that are infrequent, leaving only those that are frequent => F_{k+1}

Candidate Generation: 1-Brute-force method

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

Items

Item
Beer
Bread
Cola
Diapers
Eggs
Milk



Candidate Generation

Itemset
{Beer, Bread, Cola}
{Beer, Bread, Diapers}
{Beer, Bread, Eggs}
{Beer, Bread, Milk}
{Beer, Cola, Diapers}
{Beer, Cola, Eggs}
{Beer, Cola, Milk}
{Beer, Diapers, Eggs}
{Beer, Diapers, Milk}
{Beer, Eggs, Milk}
{Bread, Cola, Diapers}
{Bread, Cola, Eggs}
{Bread, Cola, Milk}
{Bread, Diapers, Eggs}
{Bread, Diapers, Milk}
{Bread, Eggs, Milk}
{Cola, Diapers, Eggs}
{Cola, Diapers, Milk}
{Cola, Eggs, Milk}
{Diapers, Eggs, Milk}

نتخاب ۳ از ۶ که میشه ۲۰ تا حالت
برای بروت فورس
ینی ۲۰ تا کاندید ۳ ایتمی داریم در
این روش که از ساخت حالت های
مختلف تک ایتمی ها درست میشن

با استفاده از frequent 2-itemset
ایمی ترکیب های اشتباه
کردیم و حذف شون
کردیم که فقط یه ترکیب ۳-تایی
موند

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

اید ساپورتش رو بشماریم ببینیم رعایت میکنه
شرط رو یا نه اگر رعایت نکرد باید حذف کنیم

Frequent 2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

الآن همه ی subset های این frequent itemset باشند
bread,dipare
bread,milk
dipare , milk
که طبق جدول بالا هستن

Figure 5.6. A brute-force method for generating candidate 3-itemsets.

Candidate Generation: 2-Merge F_{k-1} and F₁ itemsets

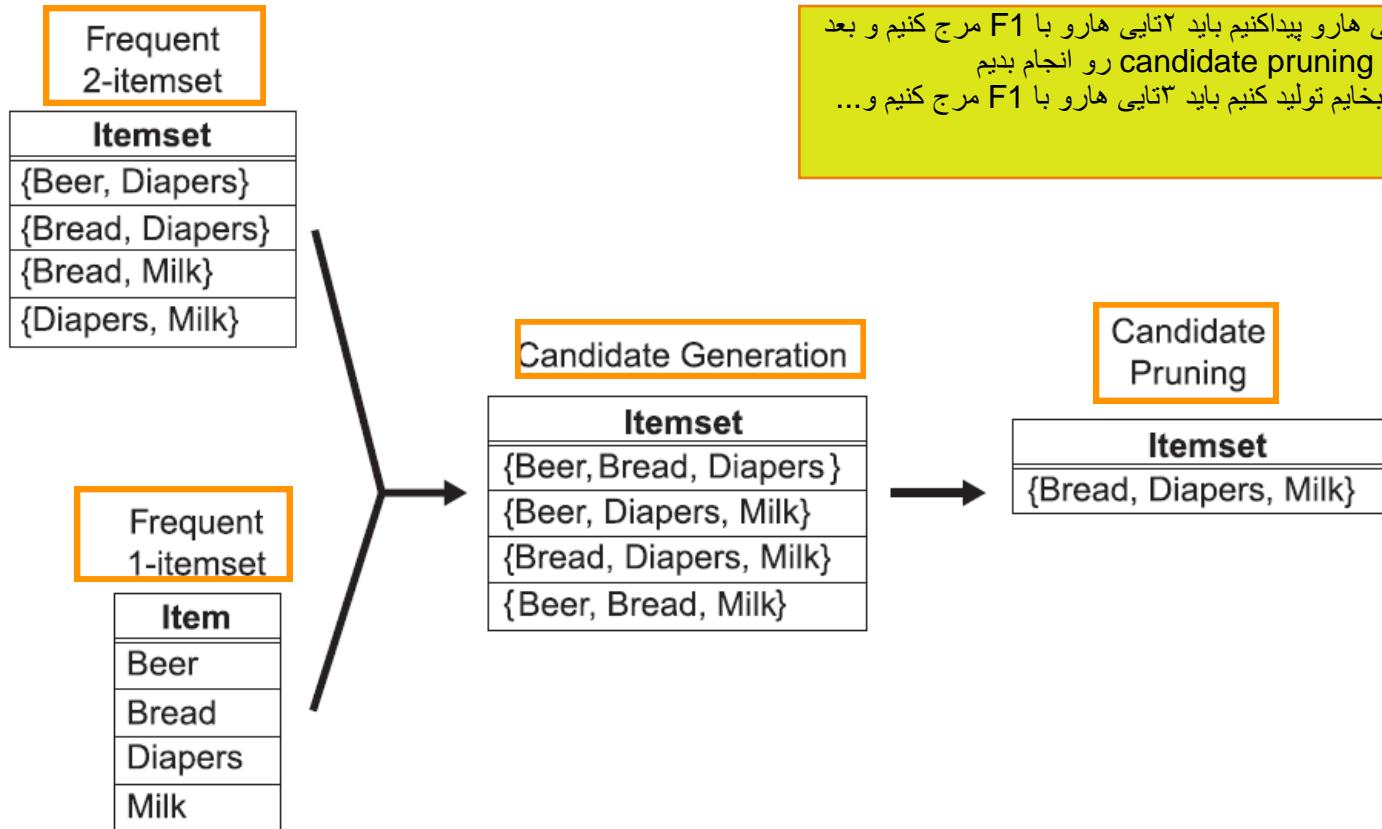


Figure 5.7. Generating and pruning candidate k -itemsets by merging a frequent $(k - 1)$ -itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

Candidate Generation: $3 \cdot F_{k-1} \times F_{k-1}$ Method

- Merge two frequent $(k-1)$ -itemsets
if their first $(k-2)$ items are identical
- $F_3 = \{\text{ABC}, \text{ABD}, \text{ABE}, \text{ACD}, \text{BCD}, \text{BDE}, \text{CDE}\}$
 - Merge(ABC, ABD) = ABCD
 - Merge(ABC, ABE) = ABCE
 - Merge(ABD, ABE) = ABDE
 - Do not merge(ABD,ACD) because they share only prefix of length 1 instead of length 2

Candidate Pruning

Let $F_3 = \{ABC, ABD, ABE, ACD, BCD, BDE, CDE\}$ be the set of frequent 3-itemsets

$L_4 = \{ABCD, ABCE, ABDE\}$ is the set of candidate 4-itemsets generated (from previous slide)

- Candidate pruning
 - Prune ABCE because ACE and BCE are infrequent
 - Prune ABDE because ADE is infrequent
- After candidate pruning: $L_4 = \{ABCD\}$

Candidate Generation:3-F_{k-1} x F_{k-1} Method

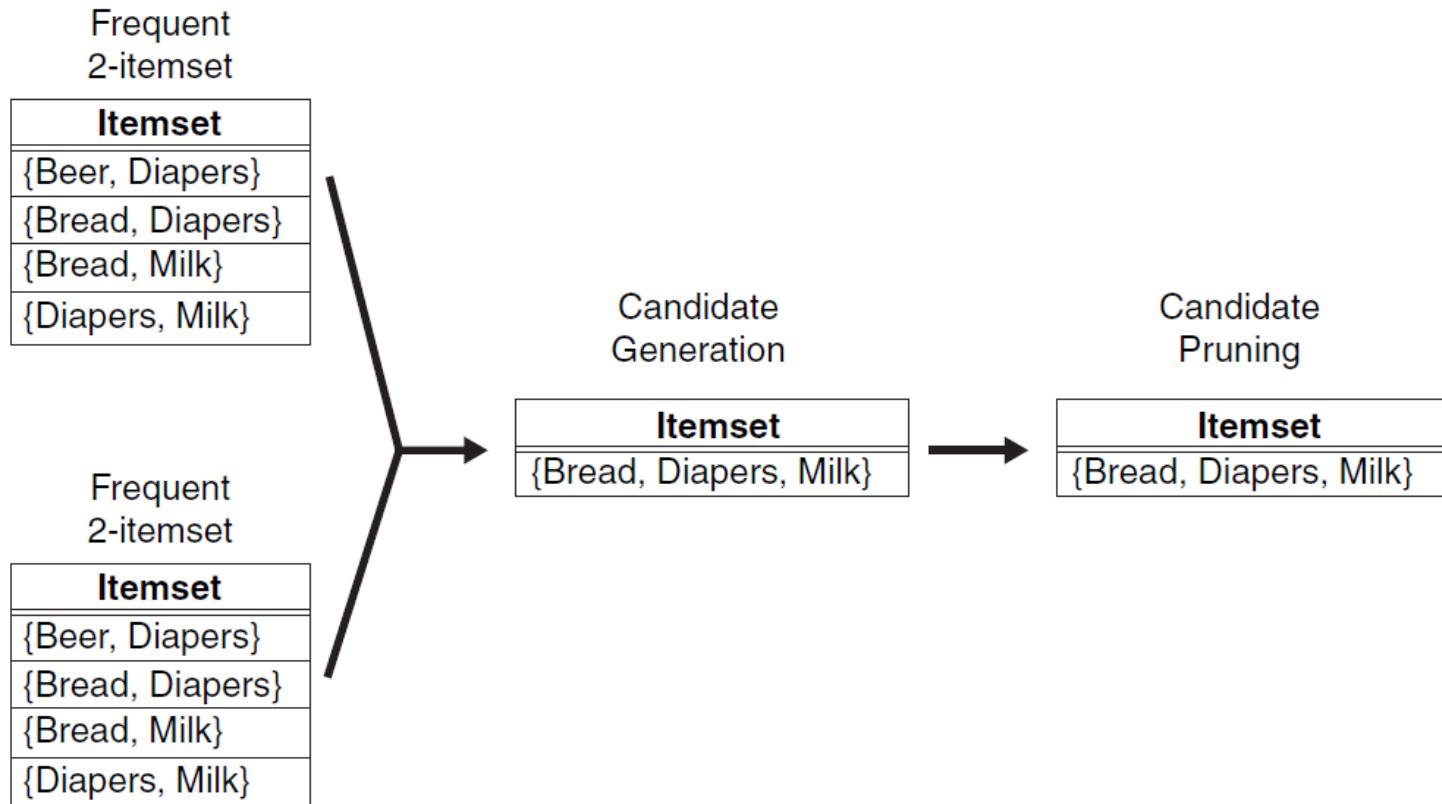


Figure 5.8. Generating and pruning candidate k -itemsets by merging pairs of frequent $(k-1)$ -itemsets.

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)



Itemset	Count
{Bread, Milk}	3
{Bread, Beer}	2
{Bread, Diaper}	3
{Milk, Beer}	2
{Milk, Diaper}	3
{Beer, Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3 \\ 6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 1 = 13$$



Triplets (3-itemsets)

Itemset	Count
{Bread, Diaper, Milk}	2

Use of $F_{k-1} \times F_{k-1}$ method for candidate generation results in only one 3-itemset. This is eliminated after the support counting step.

Support Counting of Candidate Itemsets

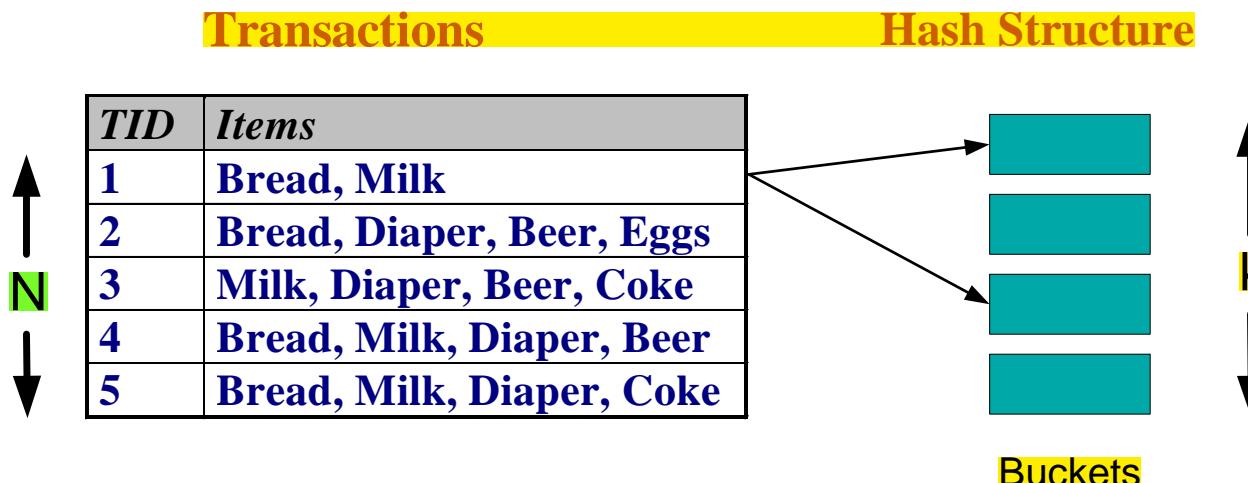
- Scan the database of transactions to determine the support of each candidate itemset
 - Must match every candidate itemset against every transaction, which is an expensive operation

<i>TID</i>	<i>Items</i>
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Itemset
{ Beer, Diaper, Milk}
{ Beer,Bread,Diaper}
{Bread, Diaper, Milk}
{ Beer, Bread, Milk}

Support Counting of Candidate Itemsets

- To reduce number of comparisons, store the candidate itemsets in a hash structure
 - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

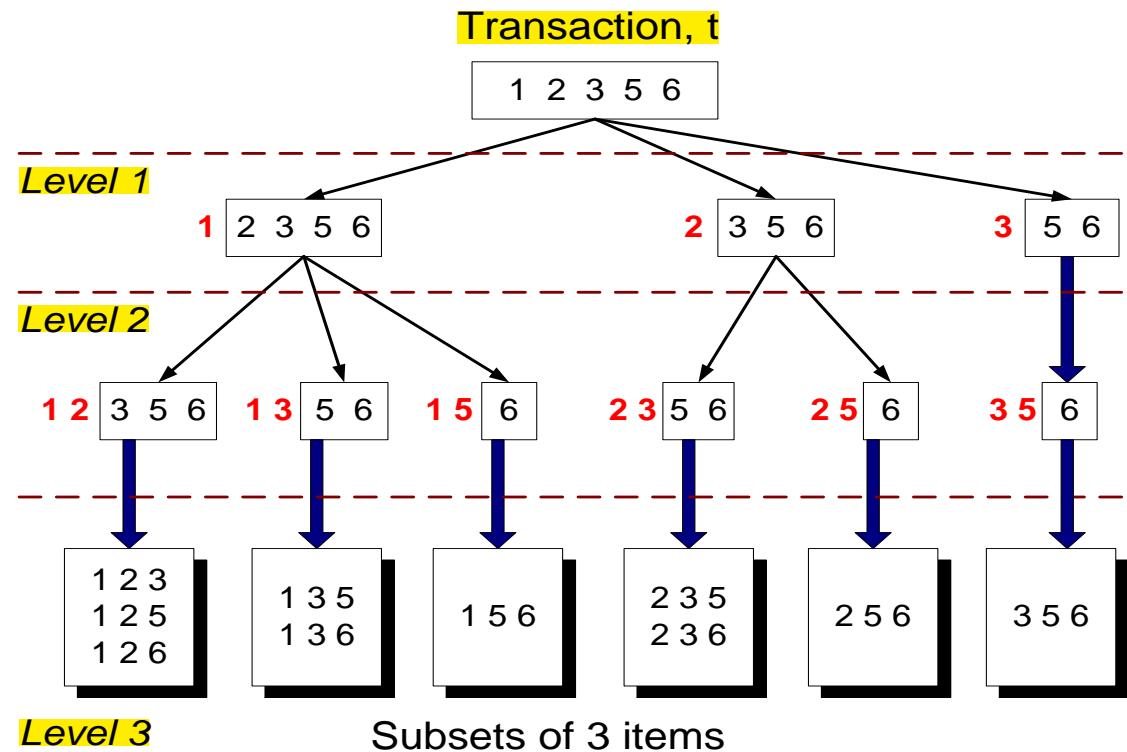


Support Counting: An Example

Suppose you have 15 candidate itemsets of length 3:

$\{1\ 4\ 5\}$, $\{1\ 2\ 4\}$, $\{4\ 5\ 7\}$, $\{1\ 2\ 5\}$, $\{4\ 5\ 8\}$, $\{1\ 5\ 9\}$, $\{1\ 3\ 6\}$, $\{2\ 3\ 4\}$, $\{5\ 6\ 7\}$, $\{3\ 4\ 5\}$,
 $\{3\ 5\ 6\}$, $\{3\ 5\ 7\}$, $\{6\ 8\ 9\}$, $\{3\ 6\ 7\}$, $\{3\ 6\ 8\}$

How many of these itemsets are supported by transaction $(1,2,3,5,6)$?



Association Rule Mining

استخراج قوانيين

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

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

Example of Association Rules

$\{\text{Diaper}\} \rightarrow \{\text{Beer}\}$,
 $\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\}$,
 $\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\}$,

Implication means co-occurrence,
not causality!

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
 - $\text{support} \geq \text{minsup threshold}$
 - $\text{confidence} \geq \text{minconf threshold}$

سطح اطمینان این
قانون چقدر؟ چقدر
این قانون مطمئنه؟



Definition: Association Rule

- **Association Rule**

- An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
- Example:
 $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

<i>TID</i>	<i>Items</i>
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Rule Evaluation Metrics

- Support (s)
 - ◆ Fraction of transactions that contain both X and Y
- Confidence (c)
 - ◆ Measures how often items in Y appear in transactions that contain X

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

Example:

$$\{ \text{Milk, Diaper} \} \Rightarrow \{ \text{Beer} \}$$

X

Y

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4$$

X, Y

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

X

Mining Association Rules(Example)

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

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
 {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Support Vs Confidence

- I. Support and confidence are both high. II. Support and confidence are both low.

I
A, B
A, B, C
A, B, D
A, B
A, B, C, D



$A \rightarrow B$
sup = 1
conf = 1

II
A
B
A, C
B, C
C, D



$A \rightarrow B$
sup = 0
conf = 0

Support Vs Confidence

III. Confidence is high and support is low.

III
A, B, D
A, C, D
A, D, E
B, E, F
B, C, D, E, F
G, A



$$\begin{aligned} G \rightarrow A \\ \text{sup} &= \frac{1}{6} \\ \text{conf} &= 1 \end{aligned}$$

IV. Confidence is low and support is high.

It is impossible because:

$$\text{Sup} \leq \text{Conf}$$

$$\text{Conf}(A \rightarrow B) = \frac{P(A,B)}{P(A)} = \frac{\text{Sup}(A \rightarrow B)}{P(A)}$$

$P(A) \leq 1$

Association Rule Mining

- Brute-force approach:
 - List all possible association rules
 - Compute the support and confidence for each rule
 - Prune rules that fail the *minsup* and *minconf* thresholds
- ⇒ Computationally prohibitive!

Mining Association Rules by Frequent Itemset

- Two-step approach:
 1. Frequent Itemset Generation
 - Generate all itemsets whose support $\geq \text{minsup}$
 2. 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

Rule Generation

- Given a frequent itemset L , find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
 - If $\{A, B, C, D\}$ is a frequent itemset, candidate rules:

$ABC \rightarrow D$,

$ABD \rightarrow C$,

$ACD \rightarrow B$,

$BCD \rightarrow A$,

$A \rightarrow BCD$,

$B \rightarrow ACD$,

$C \rightarrow ABD$,

$D \rightarrow ABC$

$AB \rightarrow CD$,

$AC \rightarrow BD$,

$AD \rightarrow BC$,

$BC \rightarrow AD$,

$BD \rightarrow AC$,

$CD \rightarrow AB$,

در اینجا، $k=4$ است پس ۱۴ تا کاندید داریم برای تولید قانون
که k طول یا تعداد ایتم ها در frequent itemset است

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

L₁

Itemset	Sup.count
I1	6
I2	7
I3	6
I4	2
I5	2

TID

Items

T1	I1, I2, I5
T2	I2, I4
T3	I2, I3
T4	I1, I2, I4
T5	I1, I3
T6	I2, I3
T7	I1, I3
T8	I1, I2, I3, I5
T9	I1, I2, I3

L₂

Itemset	Sup.count
I1, I2	4
I1, I3	4
I1, I5	2
I2, I3	4
I2, I4	2
I2, I5	2

L₃

Itemset	Sup. count
I1, I2, I3	2
I1, I2, I5	2

Minsup = 2, minconf = %70

برای ۲-ایتمی ها باید
۲ - 2^2 یعنی ۲ تا کاندید رو
چک کنیم برای قوانین
association

I1, I2	I1 → I2	$\text{conf} = \frac{4}{6}$	✗
	I2 → I1	$\text{conf} = \frac{4}{7}$	✗
I1, I3	I1 → I3	$\text{conf} = \frac{4}{6}$	✗
	I3 → I1	$\text{conf} = \frac{4}{6}$	✗
I2, I5	I2 → I5	$\text{conf} = \frac{2}{7}$	✗
	I5 → I2	$\text{conf} = 1$	✓

$$\text{Conf}(A \rightarrow B) = \frac{P(A, B)}{P(A)} = \frac{\text{Sup}(A \rightarrow B)}{P(A)}$$

$P(A) \leq 1$

L_1	
Itemset	Sup.count
I1	6
I2	7
I3	6
I4	2
I5	2

L_2	
Itemset	Sup.count
I1, I2	4
I1, I3	4
I1, I5	2
I2, I3	4
I2, I4	2
I2, I5	2

I1, I2, I3	I1 → I2 I3	$\text{conf} = \frac{2}{6}$	✗
	I2 → I1 I3	$\text{conf} = \frac{2}{7}$	✗
	I3 → I1 I2	$\text{conf} = \frac{2}{6}$	✗
	I1 I2 → I3	$\text{conf} = \frac{2}{4}$	✗
	I1 I3 → I2	$\text{conf} = \frac{2}{4}$	✗
	I2 I3 → I1	$\text{conf} = \frac{2}{4}$	✗
I1, I2, I5	I5 → I1 I2	$\text{conf} = 1$	✗
	I1 I5 → I2	$\text{conf} = 1$	✓
	I2 I5 → I1	$\text{conf} = 1$	✓

تعداد رول هایی که میتوانه تولید کنه:
چون $k=3$ است پس
 $2^3 - 2 = 6$

L ₁		L ₂	
Itemset	Sup.count	Itemset	Sup.count
I1	6	I1, I2	4
I2	7	I1, I3	4
I3	6	I1, I5	2
I4	2	I2, I3	4
I5	2	I2, I4	2
		I2, I5	2

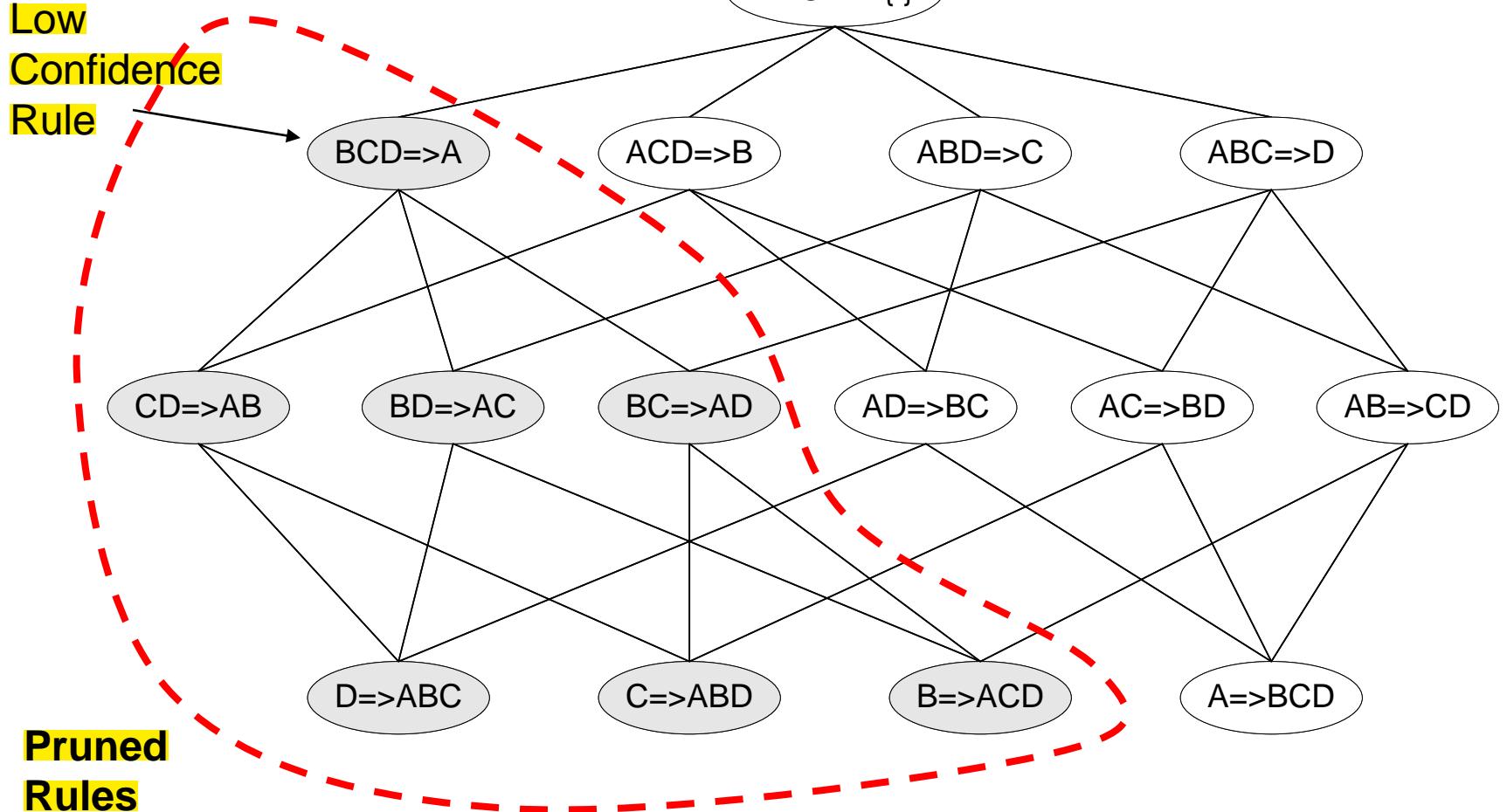
$\text{conf} = P(I_5, I_1, I_2) / P(I_5)$
 $= 2/2 = 1$

Rule Generation

- In general, confidence does not have an anti-monotone property
 - c($ABC \rightarrow D$) can be larger or smaller than c($AB \rightarrow D$)
 - ضد يکنواخت
- But confidence of rules generated from the same itemset has an anti-monotone property
 - E.g., Suppose {A,B,C,D} is a frequent 4-itemset:
$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$
 - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm

Lattice of rules



Example

ID	Basketball	Cereal consumption
...
...

C \ B	YES	NO	
YES	2000	1750	3750
NO	1000	250	1250
	3000	2000	5000

Example

ID	Basketball	Cereal consumption
...
...

$\text{Basketball} \rightarrow \text{Cereal consumption}$

$$\text{sup} = \frac{2000}{5000} = \% 40$$

$$\text{conf} = \frac{2000}{3000} = \% 66$$

$$P(\text{Cereal consumption}) = \frac{3750}{5000} = \% 75$$

C \ B	YES	NO	
YES	2000	1750	3750
NO	1000	250	1250
3000	2000	5000	

$\text{Basketball} \rightarrow \overline{\text{Cereal consumption}}$

$$\text{sup} = \frac{1000}{5000} = \% 20$$

$$\text{conf} = \frac{1000}{3000} = \% 33.3$$

$$P(\overline{\text{Cereal consumption}}) = \% 25$$

Example

Is Symptom → Disease a valid rule?

S \ D	YES	NO	
YES	80	40	120
NO	20	10	30
	100	50	150

Example

Is Symptom \rightarrow Disease a valid rule?

S \ D	YES	NO	
YES	80	40	120
NO	20	10	30
	100	50	150

$S \rightarrow D$

$$\text{sup} = \frac{80}{15000} = \% 53$$

$$\text{conf} = \frac{80}{120} = \% 66$$

P(S)

But S and D are independent!
 $P(D|S) = P(D) = 0.67$

Lift Measure

Strong Rules are not necessarily interesting.
We need more measures to evaluate rules.

$$\text{Lift}(A \rightarrow B) = \frac{P(A, B)}{P(A)P(B)} = \frac{\text{conf} (A \rightarrow B)}{P (B)} = \text{Lift}(B \rightarrow A)$$

Lift < 1

$P (B | A) < P (B)$

Negative Correlation

Lift = 1

$P (B | A) = P (B)$

Independent

Lift > 1

$P (B | A) > P (B)$

Positive Correlation

In data mining, lift is a measure of the performance of a predictive model at distinguishing between positive and negative cases. It measures the degree to which the presence of one event can predict the presence of another event. Specifically, lift is defined as the ratio of the observed frequency of a particular outcome to the expected frequency of that outcome.

Lift can be used in a variety of ways in data mining. One common use is in market basket analysis, where it is used to identify associations between items purchased together. For example, if customers who buy bread are more likely to buy butter than customers who don't buy bread, this suggests a strong association between these two items. The lift measure can be used to quantify the strength of this association, and can be used to make recommendations to customers based on their purchase history.

Another use of lift is in fraud detection, where it is used to identify unusual patterns of behavior. For example, if a customer suddenly starts making large purchases or withdrawing large sums of money from their account, this could be an indication of fraudulent activity. The lift measure can be used to flag these transactions as potentially suspicious and alert investigators to investigate further.

Overall, lift is a useful measure in data mining because it allows us to identify relationships between variables that might not be immediately apparent. By quantifying the strength of these relationships, we can better understand the underlying patterns in our data and make more informed decisions.

Here's an example: Let's assume a supermarket wants to find out what products are often purchased together. To identify this, they analyze their sales data and calculate the lift measure for item pairs. If the lift measure for milk and bread is 2.5, this means that customers who bought milk were 2.5 times more likely to buy bread compared to customers who didn't buy milk. Thus, the supermarket can use this information to create a marketing strategy such as offering discounts for customers who buy both bread and milk together.

Another example could be in the credit card industry, where a bank wants to identify fraudulent transactions. The bank can analyze their transaction data and calculate the lift measure for each transaction. If a transaction has a high lift measure, it means that it is unusual compared to the rest of the transactions and may require further investigation. This way, the bank can prevent fraudulent activity and protect its customers from financial losses.

Sure, let's consider a sample dataset consisting of 1000 transactions and 5 items: A, B, C, D, E. Here's an example:

Transaction ID | Items Purchased

---|---

1 | A, B, C
2 | A, B, D
3 | A, C, D
4 | B, C, D, E
5 | A, B

To calculate the confidence, lift, and support for itemsets, we need to first identify all the frequent itemsets in the dataset. An itemset is considered frequent if it appears in a minimum number of transactions, which is typically defined as the support threshold.

Let's assume a minimum support threshold of 20%, meaning that any itemset that appears in at least 20% of the transactions is considered frequent. Using this threshold, we can identify the following frequent itemsets:

- {A}: 60% (appears in transactions 1, 2, 3, 5)
- {B}: 80% (appears in transactions 1, 2, 4, 5)
- {C}: 40% (appears in transactions 1, 3, 4)
- {D}: 40% (appears in transactions 2, 3, 4)
- {A,B}: 40% (appears in transactions 1, 2, 5)
- {B,C}: 40% (appears in transactions 1, 4)
- {B,D}: 30% (appears in transactions 2, 4)
- {A,C}: 30% (appears in transactions 1, 3)

Now, using these frequent itemsets, we can calculate the confidence, lift, and support for each rule (i.e., each possible combination of items within each frequent itemset). Here's how to do it:

1. Confidence: The confidence of a rule A \rightarrow B is the probability that a transaction that contains A also contains B. It is calculated as $\text{support}(A,B) / \text{support}(A)$.

For example, the confidence for the rule {A} \rightarrow {B} is $40\% / 60\% = 0.67$ or 67%. This means that if a customer purchases item A, there is a 67% chance they will also purchase item B.

2. Lift: The lift of a rule A \rightarrow B is the ratio of the observed frequency of A and B occurring together to the frequency that would be expected if they were independent. It is calculated as $\text{support}(A,B) / (\text{support}(A) * \text{support}(B))$.

For example, the lift for the rule {A} \rightarrow {B} is $(40\% / 100\%) / ((60\% / 100\%) * (80\% / 100\%)) = 1.11$. A lift value greater than 1 indicates a positive correlation between items, while a lift value less than 1 indicates a negative correlation.

3. Support: The support of an itemset is the proportion of transactions that contain the itemset. It is calculated as the number of transactions containing the itemset divided by the total number of transactions.

For example, the support for itemset {A,B} is $40\% / 100\% = 0.4$, meaning that 40% of the transactions in the dataset contain both items A and B.

Using these calculations, we can generate a table of all the frequent itemsets and their corresponding confidence, lift, and support values:

Frequent Itemset | Confidence | Lift | Support

---|---|---|---

{A} | - | - | 60%
{B} | - | - | 80%
{C} | - | - | 40%
{D} | - | - | 40%
{A,B} | 67% | 1.11 | 40%
{B,C} | 50% | 1.25 | 40%
{B,D} | 37.5% | 0.94 | 30%
{A,C} | 50% | 1.25 | 30%

This table gives us valuable insight into which items are frequently purchased together, as well as how strong the relationships between them are. We can use this information to make data-driven decisions about marketing strategies, product bundling, and other business decisions.

Sure, here's another example for a different dataset with different items:

Transaction ID | Items Purchased

Transaction ID	Items Purchased
1	A, B, C
2	A, B, D
3	A, D, E
4	B, C, D, E
5	A, B, E
6	C, D, E

Let's assume a minimum support threshold of 25%, meaning that any itemset that appears in at least 25% of the transactions is considered frequent. Using this threshold, we can identify the following frequent itemsets:

- {A}: 50% (appears in transactions 1, 2, 3, 5)
- {B}: 66.67% (appears in transactions 1, 2, 4, 5)
- {C}: 33.33% (appears in transactions 1, 4, 6)
- {D}: 50% (appears in transactions 2, 4, 6)
- {E}: 50% (appears in transactions 3, 4, 6)
- {A,B}: 33.33% (appears in transactions 1, 2, 5)
- {B,D}: 33.33% (appears in transactions 2, 4)
- {B,E}: 33.33% (appears in transactions 4, 5)
- {C,D}: 33.33% (appears in transactions 4, 6)
- {C,E}: 33.33% (appears in transactions 3, 4, 6)
- {D,E}: 50% (appears in transactions 4, 6)

Using these frequent itemsets, we can calculate the confidence, lift, and support for each rule:

Frequent Itemset | Confidence | Lift | Support

Frequent Itemset	Confidence	Lift	Support
{A}	-	-	50%
{B}	-	-	66.67%
{C}	-	-	33.33%
{D}	-	-	50%
{E}	-	-	50%
{A,B}	50%	0.75	33.33%
{B,D}	50%	0.75	33.33%
{B,E}	50%	0.75	33.33%
{C,D}	100%	3	33.33%
{C,E}	100%	3	33.33%
{D,E}	100%	2	50%

From this table, we can see that items C and D are always purchased together in the dataset, with a confidence of 100% and a lift of 3. This suggests a strong association between these items, which could be used to inform marketing strategies or product placement decisions.

We can also see that the association between items A and B is weaker than the association between items C and D, with a lower confidence and lift value. This suggests that while these items are frequently purchased together, the relationship between them is not as strong as the relationship between C and D.

Overall, by analyzing the frequent itemsets and calculating the confidence, lift, and support values for each rule, we can gain valuable insights into the relationships between items in our dataset and make data-driven decisions about how to improve our business operations.

Lift Measure

C \ B	YES	NO	
YES	2000	1750	3750
NO	1000	250	1250
	3000	2000	5000

Basketball → Cereal consumption

$$\text{Lift} = \frac{\frac{2000}{5000}}{\frac{3000}{5000} \times \frac{3750}{5000}} = \frac{100}{3 \times 375} = 0.88$$

Basketball → Cereal consumption

$$\text{Lift} = \frac{\frac{1000}{5000}}{\frac{3000}{5000} \times \frac{1250}{5000}} = \frac{500}{3 \times 125} = 1.33$$

Lift Measure

Lift measure is not null-invariant

	B	\bar{B}	
C	100	1000	1100
\bar{C}	1000	null count	
	1100		

If null count = 100000

$$\text{Lift } (B,C) = \frac{P(B,C)}{P(B)P(C)} = \frac{\frac{100}{102100}}{\frac{1100}{102100} \times \frac{1100}{102100}} = 8.44 \gg 1$$

If null count = 100

$$\text{Lift } (B,C) = \frac{P(B,C)}{P(B)P(C)} = \frac{\frac{100}{2200}}{\frac{1100}{2200} \times \frac{1100}{2200}} = 0.18 \ll 1$$

All Confidence

$$\text{All-confidence}(A,B) = \frac{P(A,B)}{\max(P(A),P(B))}$$

$$0 \leq \text{All-confidence} \leq 1$$

Other Measure

symbol	measure	range	formula
ϕ	ϕ -coefficient	-1 ... 1	$\frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	-1 ... 1	$\frac{P(A,B)P(\bar{A},\bar{B}) - P(A,\bar{B})P(\bar{A},B)}{P(A,B)P(\bar{A},B) + P(A,\bar{B})P(\bar{A},B)}$
Y	Yule's Y	-1 ... 1	$\frac{\sqrt{P(A,B)P(\bar{A},\bar{B})} - \sqrt{P(A,\bar{B})P(\bar{A},B)}}{\sqrt{P(A,B)P(\bar{A},\bar{B})} + \sqrt{P(A,\bar{B})P(\bar{A},B)}}$
k	Cohen's	-1 ... 1	$\frac{P(A,B) + P(\bar{A},\bar{B}) - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$
PS	Piatetsky-Shapiro's	-0.25 ... 0.25	$P(A, B) - P(A)P(B)$
F	Certainty factor	-1 ... 1	$\max\left(\frac{P(B A) - P(B)}{1 - P(B)}, \frac{P(A B) - P(A)}{1 - P(A)}\right)$
AV	added value	-0.5 ... 1	$\max(P(B A) - P(B), P(A B) - P(A))$
K	Klosgen's Q	-0.33 ... 0.38	$\sqrt{P(A,B)\max(P(B A) - P(B), P(A B) - P(A))}$
g	Goodman-kruskal's	0 ... 1	$\frac{\sum_j \max_k P(A_j, B_k) + \sum_k \max_j P(A_j, B_k) - \max_j P(A_j) - \max_k P(B_k)}{2 - \max_j P(A_j) - \max_k P(B_k)}$ $\Sigma_i \Sigma_j P(A_i, B_j) \log \frac{P(A_i)P(B_j)}{P(A_i)P(B_j)}$
M	Mutual Information	0 ... 1	$\min(-\sum_i P(A_i) \log P(A_i), -\sum_i P(B_i) \log P(B_i) \log P(B_i))$
J	J-Measure	0 ... 1	$\max(P(A, B) \log(\frac{P(B A)}{P(B)}) + P(\bar{A}\bar{B}) \log(\frac{P(\bar{B} A)}{P(\bar{B})}))$ $P(A, B) \log(\frac{P(A B)}{P(A)}) + P(\bar{A}\bar{B}) \log(\frac{P(\bar{A} B)}{P(\bar{A})})$
G	Gini index	0 ... 1	$\max(P(A)P(\bar{B} A)^2 + P(\bar{B} A)^2 + P(\bar{A} P(\bar{B})^2 + P(\bar{B} \bar{A})^2) - P(B)^2 - P(\bar{B})^2,$ $P(B)(P(A B)^2 + P(\bar{A} B)^2) + P(\bar{B} P(A \bar{B})^2 + P(\bar{A} \bar{B})^2) - P(A)^2 - P(\bar{A})^2)$
s	support	0 ... 1	$P(A, B)$
c	confidence	0 ... 1	$\max(P(B A), P(A B))$
L	Laplace	0 ... 1	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
IS	Cosine	0 ... 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
γ	coherence(Jaccard)	0 ... 1	$\frac{P(A) + P(B) - P(A,B)}{P(A) + P(B)}$
α	all_confidence	0 ... 1	$\frac{\max(P(A), P(B))}{P(A, B)}$
o	odds ratio	0 ... ∞	$\frac{P(A,B)}{P(\bar{A},B)P(A,\bar{B})}$
V	Conviction	0.5 ... ∞	$\max(\frac{P(A)P(\bar{B})}{P(AB)}, \frac{P(B)P(\bar{A})}{P(B\bar{A})})$
λ	lift	0 ... ∞	$\frac{P(A,B)}{P(A)P(B)}$
S	Collective strength	0 ... ∞	$\frac{P(A,B) + P(\bar{A}\bar{B})}{P(A)P(B) + P(\bar{A})P(\bar{B})} \times \frac{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A,B) - P(\bar{A}\bar{B})}$
χ^2	χ^2	0 ... ∞	$\sum_i \frac{(P(A_i) - E_i)^2}{E_i}$

- 6.14 The following contingency table summarizes supermarket transaction data, where *hot dogs* refers to the transactions containing hot dogs, *hot dogs* refers to the transactions that do not contain hot dogs, *hamburgers* refers to the transactions containing hamburgers, and *hamburgers* refers to the transactions that do not contain hamburgers.

	<i>hot dogs</i>	$\overline{\text{hot dogs}}$	Σ_{row}
<i>hamburgers</i>	2000	500	2500
$\overline{\text{hamburgers}}$	1000	1500	2500
Σ_{col}	3000	2000	5000

- (a) Suppose that the association rule "*hot dogs* \Rightarrow *hamburgers*" is mined. Given a minimum support threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong?
- (b) Based on the given data, is the purchase of *hot dogs* independent of the purchase of *hamburgers*? If not, what kind of correlation relationship exists between the two?

Answer:

- (a) Suppose that the association rule "*hotdogs* \Rightarrow *hamburgers*" is mined. Given a minimum support threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong?
For the rule, support = $2000/5000 = 40\%$, and confidence = $2000/3000 = 66.7\%$. Therefore, the association rule is strong.
- (b) Based on the given data, is the purchase of *hotdogs* independent of the purchase of *hamburgers*? If not, what kind of correlation relationship exists between the two?
 $\text{corr}_{\{\text{hotdog}, \text{hamburger}\}} = P(\{\text{hot dog, hamburger}\}) / (P(\{\text{hot dog}\}) P(\{\text{hamburger}\})) = 0.4 / (0.5 \times 0.6) = 1.33 > 1$. So, the purchase of hotdogs is NOT independent of the purchase of hamburgers. There exists a POSITIVE correlation between the two.

13. Give a short example to show that items in a strong association rule may actually be negatively correlated.

Answer:

Consider the following table:

	<i>A</i>	\overline{A}	Σ_{row}
<i>B</i>	65	35	100
\overline{B}	40	10	50
Σ_{col}	105	35	150

Let the minimum support be 40%. Let the minimum confidence be 60%. $A \Rightarrow B$ is a strong rule because it satisfies minimum support and minimum confidence with a support of $65/150 = 43.3\%$ and a confidence of $65/100 = 61.9\%$. However, the correlation between *A* and *B* is $\text{corr}_{A,B} = \frac{0.433}{0.700 \times 0.667} = 0.928$, which is less than 1, meaning that the occurrence of *A* is negatively correlated with the occurrence of *B*.

6. A database has 5 transactions. Let $\text{min_sup} = 60\%$ and $\text{min_conf} = 80\%$.

<i>TID</i>	<i>items_bought</i>
T100	{M, O, N, K, E, Y}
T200	{D, O, N, K, E, Y }
T300	{M, A, K, E}
T400	{M, U, C, K, Y}
T500	{C, O, O, K, I ,E}

- (a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.
- (b) List all of the strong association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and $item_i$ denotes variables representing items (e.g., “A”, “B”, etc.):

$$\forall x \in \text{transaction}, buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3) \quad [s, c]$$

- (a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.

i. For Apriori, one finds the following frequent itemsets, and candidate itemsets (after deletion as a result of has_infrequent_subset):

$$L_1 = \{E, K, M, O, Y\}$$

$$C_2 = \{EK, EM, EO, EY, KM, KO, KY, MO, MY, OY\}$$

$$L_2 = \{EK, EO, KM, KO, KY\}$$

$$C_3 = \{EKO\}$$

$$L_3 = \{EKO\}$$

$$C_4 = \emptyset$$

$$L_4 = \emptyset$$

Finally resulting in the complete set of frequent itemsets:

$$\{E, K, M, O, Y, EK, EO, KM, KO, KY, EKO\}$$

- (b) List all of the strong association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and $item_i$ denotes variables representing items (e.g., “A”, “B”, etc.):

$$\forall x \in \text{transaction}, buys(X, item_1) \wedge buys(X, item_2) \Rightarrow buys(X, item_3) \quad [s, c]$$

The following strong association rules are found:

$$\forall X \in \text{transaction}, buys(X, E) \wedge buys(X, O) \Rightarrow buys(X, K) \quad [60\%, 100\%] \quad \forall X \in \text{transaction}, buys(X, K) \wedge buys(X, O) \Rightarrow buys(X, E) \quad [60\%, 100\%]$$

6.3 The Apriori algorithm makes use of prior knowledge of subset support properties.

- (a) Prove that all nonempty subsets of a frequent itemset must also be frequent.
- (b) Prove that the support of any nonempty subset s' of itemset s must be at least as great as the support of s .
- (c) Given frequent itemset l and subset s of l , prove that the confidence of the rule $"s' \Rightarrow (l - s)"$ cannot be more than the confidence of $"s \Rightarrow (l - s)"$, where s' is a subset of s .
- (d) A partitioning variation of Apriori subdivides the transactions of a database D into n nonoverlapping partitions. Prove that any itemset that is frequent in D must be frequent in at least one partition of D .

(a) Prove that all nonempty subsets of a frequent itemset must also be frequent.

Let s be a frequent itemset. Let min_sup be the minimum support. Let D be the task-relevant data, a set of database transactions. Let $|D|$ be the number of transactions in D . Since s is a frequent itemset $\text{support_count}(s) = \text{min_sup} \times |D|$.

Let s' be any nonempty subset of s . Then any transaction containing itemset s will also contain itemset s' . Therefore, $\text{support_count}(s') \geq \text{support_count}(s) = \text{min_sup} \times |D|$. Thus, s' is also a frequent itemset.

(b) Prove that the support of any nonempty subset s' of itemset s must be as great as the support of s .

Let D be the task-relevant data, a set of database transactions. Let $|D|$ be the number of transactions in D . By definition,

$$\text{support}(s) = \frac{\text{support_count}(s)}{|D|}.$$

Let s' be any nonempty subset of s . By definition, $\text{support}(s') = \frac{\text{support_count}(s')}{|D|}$.

From part (a) we know that $\text{support}(s') \geq \text{support}(s)$. This proves that the support of any nonempty subset s' of itemset s must be as great as the support of s .

(c) Given frequent itemset l and subset s of l , prove that the confidence of the rule " $s' \Rightarrow (l - s')$ " cannot be more than the confidence of " $s \Rightarrow (l - s)$ ", where s' is a subset of s .

Let s be a subset of l . Then $\text{confidence}(s \Rightarrow (l - s)) = \frac{\text{support}(l)}{\text{support}(s)}$.

Let s' be any nonempty subset of s . Then $\text{confidence}(s' \Rightarrow (l - s')) = \frac{\text{support}(l)}{\text{support}(s')}$.

From Part (b) we know that $\text{support}(s') \geq \text{support}(s)$, therefore, $\text{confidence}(s' \Rightarrow (l - s')) \leq \text{confidence}(s \Rightarrow (l - s))$. That is, the confidence of the rule " $s' \Rightarrow (l - s')$ " cannot be more than the confidence of the rule " $s \Rightarrow (l - s)$ ".

10.1 Briefly describe and give examples of each of the following approaches to clustering: *partitioning* methods, *hierarchical* methods, *density-based* methods, and *grid-based* methods.

Clustering is the process of grouping data into classes, or clusters, so that objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. There are several approaches to clustering methods.

Partitioning methods: Given a databases of n objects or data tuples, a partitioning methods constructs k partitions of data, where each partition represents a cluster and $k \leq n$. Given k , the number of partitions to construct, it creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. The general criterion of a good partitioning is that objects in the same cluster are "close" or related to each other, whereas objects of different clusters are "far apart". For example, the k -means method is one of the partitioning methods commonly used.

Hierarchical methods: A hierarchical method creates a hierarchical decomposition of the given set of data objects. It can be either agglomerative or divisive. The agglomerative (bottom-up) approach starts with each object forming a separate group. It successively merges the objects close to one another, until all of the groups are merged into one, or until a termination condition holds. The divisive (top-down) approach starts with all objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster or until a termination condition holds. BIRCH is an example of integration of hierarchical method with distance-based clustering.

Density-based methods: This method is based on density such as density-connected points. The main idea is to continue growing a given cluster as long as the density in its "neighborhood" exceeds some threshold. That is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. This method can be used to filter out noise and discover clusters of arbitrary shape. DBSCAN is a typical example of density-based clustering method.

3. Use an example to show why the *k-means* algorithm may not find the global optimum, that is, optimizing the within-cluster variation.

Answer:

Consider applying the *k-means* algorithm on the following points, and set $k = 2$. $A(0, 1)$, $B(0, -1)$, $C_i(100, 50)$ ($i = 1, \dots, 100$), and $D_j(100, -50)$ ($j = 1, \dots, 100$).

If we use A and C_1 as the initial cluster centers, then the clustering process will converge to two centers, $(0, 0)$ and $(100, 0)$. The within-cluster variation is

$$E = 1^2 + 1^2 + 100 \times 50^2 + 100 \times 50^2 = 1 + 1 + 100 \times 2500 + 100 \times 2500 = 500002.$$

However, if we use $(100, 50)$ and $(100, -50)$ as the cluster centers, the within-cluster variation is

$$E = (100^2 + 49^2) + (100^2 + 49^2) + 100 \times 0 + 100 \times 0 = 24802,$$

which is much smaller. This example shows *k-means* may be trapped by a local optimal, and cannot jump out to find the global optimum.

6. Both *k-means* and *k-medoids* algorithms can perform effective clustering.

- (a) Illustrate the strength and weakness of *k-means* in comparison with the *k-medoids* algorithm.

Answer:

k-means vs. *k medoids*: more efficient but quality deteriorates by noise and outliers.

■

- (b) Illustrate the strength and weakness of these schemes in comparison with a hierarchical clustering scheme (such as AGNES).

Answer:

Partition: can undo what was done (by moving objects around clusters)—quality is good in general, require the number of clusters to be known; find only spherical shaped clusters.

Hierarchical: cannot undo what was done—quality could be poor, does not require the number of clusters to be known; more efficient and parallel (divide and conquer), may find only arbitrary shaped clusters.

■

Present conditions under which density-based clustering is more suitable than partitioning-based clustering and hierarchical clustering. Give application examples to support your argument.

Answer:

Density-based clustering is more suitable if no or very limited domain knowledge is available to determine the appropriate values of the parameters, the clusters are expect of arbitrary shape including non convex shapes, and the efficiency is essential on large data sets.

For example, consider the application of recognizing residential area on a map where buildings and their types are labeled. A user may not have the domain knowledge about how a residential area would look like. Moreover, a residential area may be of arbitrary shape, and may not be in convex, such as those built along a river. In a large city, there are hundreds of thousands of buildings. Efficiency on large data sets is important.

- (a) An application that takes clustering as a major data mining function

Answer:

For example, clustering is a major data mining function in community finding in social networks.

■

- (b) An application that takes clustering as a preprocessing tool for data preparation for other data mining tasks.

Answer:

Web search engines often provide query suggestion services. When a user inputs one or a series of queries, the search engine tries to suggest some queries that may capture the user's information need. To overcome the data sparsity, that is, many queries are asked very few times, and many web pages are not clicked, clustering is often used as a preprocessing step to obtain micro-clusters of queries, which represent similar user intents, and web pages, which are about the same topics.

16. Describe each of the following clustering algorithms in terms of the following criteria: (1) shapes of clusters that can be determined; (2) input parameters that must be specified; and (3) limitations.

(a) *k*-means

1. Compact clouds (clusters of non-convex shape cannot be determined); 2. Number of clusters;
3. Sensitive to noise and outliers, works good on small data sets only.

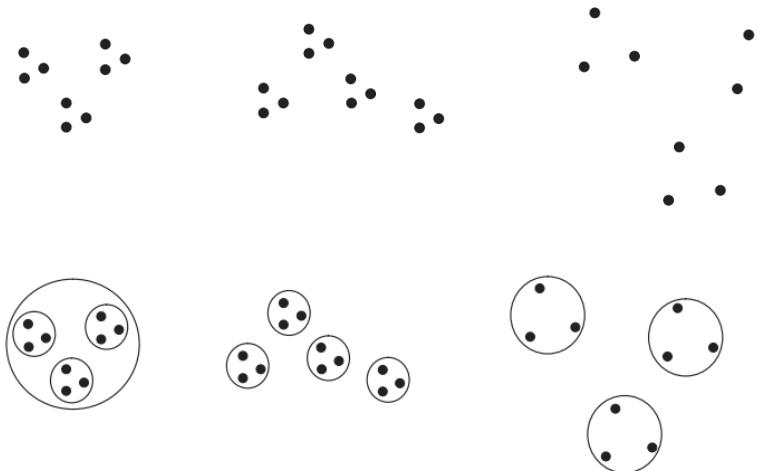
(b) *k*-medoids

1. Compact clouds (clusters of non-convex shape cannot be determined); 2. Number of clusters;
3. Small data sets (not scalable).

(f) DBSCAN

1. Arbitrary shape; 2. Maximum possible distance for a point to be considered density reachable and minimum number of points in a cluster. 3. Quadratic time in the worst case.

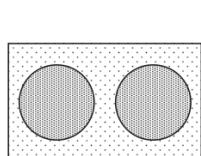
Find all well-separated clusters in the set of points shown in Figure 5.35.



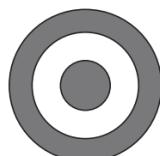
3. Many partitional clustering algorithms that automatically determine the number of clusters claim that this is an advantage. List two situations in which this is not the case.

- (a) When there is hierarchical structure in the data. Most algorithms that automatically determine the number of clusters are partitional, and thus, ignore the possibility of subclusters.
- (b) When clustering for utility. If a certain reduction in data size is needed, then it is necessary to specify how many clusters (cluster centroids) are produced.

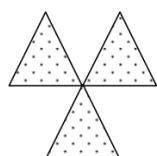
5. Identify the clusters in Figure 8.3 using the center-, contiguity-, and density-based definitions. Also indicate the number of clusters for each case and give a brief indication of your reasoning. Note that darkness or the number of dots indicates density. If it helps, assume center-based means K-means, contiguity-based means single link, and density-based means DBSCAN.



(a)



(b)



(c)

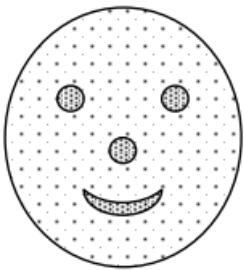


(d)

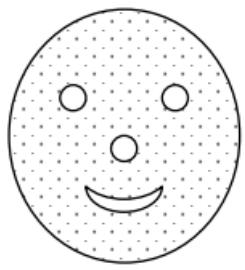
- (a) **center-based** 2 clusters. The rectangular region will be split in half. Note that the noise is included in the two clusters.
contiguity-based 1 cluster because the two circular regions will be joined by noise.
density-based 2 clusters, one for each circular region. Noise will be eliminated.
- (b) **center-based** 1 cluster that includes both rings.
contiguity-based 2 clusters, one for each rings.
density-based 2 clusters, one for each ring.
- (c) **center-based** 3 clusters, one for each triangular region. One cluster is also an acceptable answer.
contiguity-based 1 cluster. The three triangular regions will be joined together because they touch.
density-based 3 clusters, one for each triangular region. Even though the three triangles touch, the density in the region where they touch is lower than throughout the interior of the triangles.
- (d) **center-based** 2 clusters. The two groups of lines will be split in two.
contiguity-based 5 clusters. Each set of lines that intertwines becomes a cluster.
density-based 2 clusters. The two groups of lines define two regions of high density separated by a region of low density.
11. Total SSE is the sum of the SSE for each separate attribute. What does it mean if the SSE for one variable is low for all clusters? Low for just one cluster? High for all clusters? High for just one cluster? How could you use the per variable SSE information to improve your clustering?
- If the SSE of one attribute is low for all clusters, then the variable is essentially a constant and of little use in dividing the data into groups.
 - if the SSE of one attribute is relatively low for just one cluster, then this attribute helps define the cluster.
 - If the SSE of an attribute is relatively high for all clusters, then it could well mean that the attribute is noise.
 - If the SSE of an attribute is relatively high for one cluster, then it is at odds with the information provided by the attributes with low SSE that define the cluster. It could merely be the case that the clusters defined by this attribute are different from those defined by the other attributes, but in any case, it means that this attribute does not help define the cluster.
 - The idea is to eliminate attributes that have poor distinguishing power between clusters, i.e., low or high SSE for all clusters, since they are useless for clustering. Note that attributes with high SSE for all clusters are particularly troublesome if they have a relatively high SSE with respect to other attributes (perhaps because of their scale) since they introduce a lot of noise into the computation of the overall SSE.



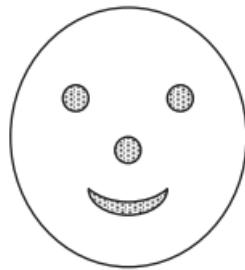
(a)



(b)



(c)



(d)

Figure 8.7. Figure for Exercise 20.

- (a) For each figure, could you use **single link** to find the patterns represented by the **nose**, **eyes**, and **mouth**? Explain.

Only for (b) and (d). For (b), the **points** in the nose, eyes, and mouth are much **closer** together than the points between these areas. For (d) there is **only space between these regions**.

- (b) For each figure, could you use **K-means** to find the patterns represented by the **nose**, **eyes**, and **mouth**? Explain.

Only for (b) and (d). For (b), **K-means** would find the nose, eyes, and mouth, but the **lower density points** would also be included. For (d), K-

means would find the nose, eyes, and mouth **straightforwardly** as long as the number of clusters was **set to 4**.

- (c) What **limitation** does clustering have in detecting all the patterns formed by the points in Figure 8.7(c)?

Clustering techniques can only **find patterns of points, not of empty spaces**.

23. Using the data in Exercise 24, compute the **silhouette coefficient** for each **point**, each of the **two clusters**, and the **overall clustering**.

Cluster 1 contains {P1, P2}, **Cluster 2** contains {P3, P4}. The dissimilarity matrix that we obtain from the similarity matrix is the following:

Table 8.3. Table of distances for Exercise 23

	P1	P2	P3	P4
P1	0	0.10	0.65	0.55
P2	0.10	0	0.70	0.60
P3	0.65	0.70	0	0.30
P4	0.55	0.60	0.30	0

Let a indicate the average distance of a point to other points in its cluster.
 Let b indicate the minimum of the average distance of a point to points in another cluster.

Point P1: $SC = 1 - a/b = 1 - 0.1/((0.65+0.55)/2) = 5/6 = 0.833$

Point P2: $SC = 1 - a/b = 1 - 0.1/((0.7+0.6)/2) = 0.846$

Point P3: $SC = 1 - a/b = 1 - 0.3/((0.65+0.7)/2) = 0.556$

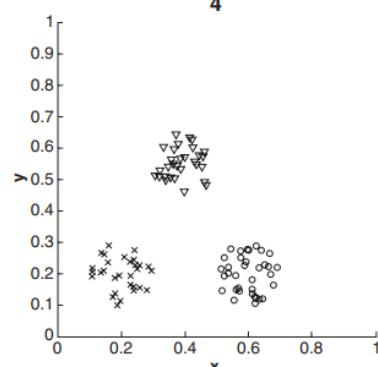
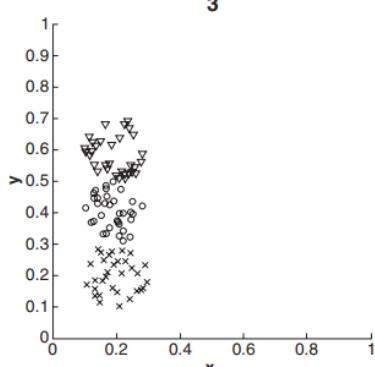
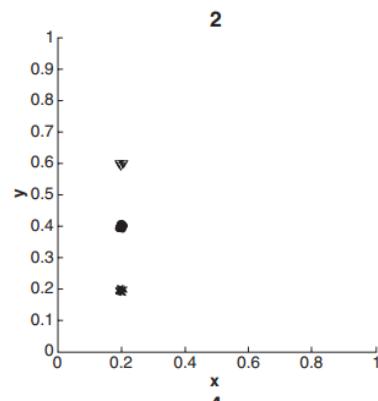
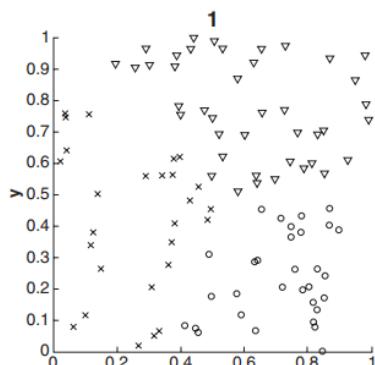
Point P4: $SC = 1 - a/b = 1 - 0.3/((0.55+0.6)/2) = 0.478$

Cluster 1 Average SC = $(0.833+0.846)/2 = 0.84$

Cluster 2 Average SC = $(0.556+0.478)/2 = 0.52$

Overall Average SC = $(0.840+0.517)/2 = 0.68$

32. In Figure 8.9, match the similarity matrices, which are sorted according to cluster labels, with the sets of points. Differences in shading and marker shape distinguish between clusters, and each set of points contains 100 points and three clusters. In the set of points labeled 2, there are three very tight, equal-sized clusters.



Answers: 1 - D, 2 - C, 3 - A, 4 - B

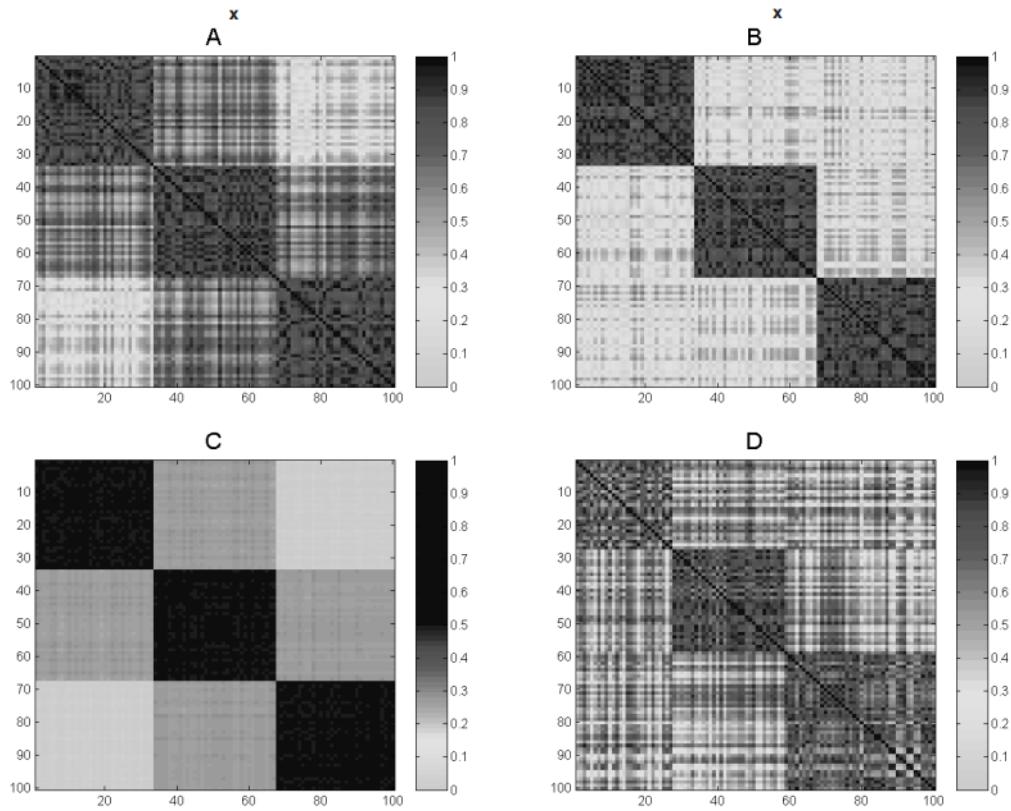


Table 6.1. Example of market basket transactions.

Customer ID	Transaction ID	Items Bought
1	0001	{a, d, e}
1	0024	{a, b, c, e}
2	0012	{a, b, d, e}
2	0031	{a, c, d, e}
3	0015	{b, c, e}
3	0022	{b, d, e}
4	0029	{c, d}
4	0040	{a, b, c}
5	0033	{a, d, e}
5	0038	{a, b, e}

$$s(\{e\}) = \frac{8}{10} = 0.8$$

$$s(\{b, d\}) = \frac{2}{10} = 0.2$$

$$s(\{b, d, e\}) = \frac{2}{10} = 0.2$$

- (b) Use the results in part (a) to compute the confidence for the association rules $\{b, d\} \rightarrow \{e\}$ and $\{e\} \rightarrow \{b, d\}$. Is confidence a symmetric measure?

Answer:

$$c(bd \rightarrow e) = \frac{0.2}{0.2} = 100\%$$

$$c(e \rightarrow bd) = \frac{0.2}{0.8} = 25\%$$

No, confidence is not a symmetric measure.

- (c) Repeat part (a) by treating each customer ID as a market basket. Each item should be treated as a binary variable (1 if an item appears in at least one transaction bought by the customer, and 0 otherwise.)

Answer:

$$s(\{e\}) = \frac{4}{5} = 0.8$$

$$s(\{b, d\}) = \frac{5}{5} = 1$$

$$s(\{b, d, e\}) = \frac{4}{5} = 0.8$$

- (d) Use the results in part (c) to compute the confidence for the association rules $\{b, d\} \rightarrow \{e\}$ and $\{e\} \rightarrow \{b, d\}$.

Answer:

$$c(bd \rightarrow e) = \frac{0.8}{1} = 80\%$$

$$c(e \rightarrow bd) = \frac{0.8}{1} = 100\%$$

3. (a) What is the confidence for the rules $\emptyset \rightarrow A$ and $A \rightarrow \emptyset$?

Answer:

$$c(\emptyset \rightarrow A) = s(\emptyset \rightarrow A),$$

$$c(A \rightarrow \emptyset) = 100\%.$$

- (b) Let c_1 , c_2 , and c_3 be the confidence values of the rules $\{p\} \rightarrow \{q\}$, $\{p\} \rightarrow \{q, r\}$, and $\{p, r\} \rightarrow \{q\}$, respectively. If we assume that c_1 , c_2 , and c_3 have different values, what are the possible relationships that may exist among c_1 , c_2 , and c_3 ? Which rule has the lowest confidence?

Answer:

$$c_1 = \frac{s(p \cup q)}{s(p)}$$

$$c_2 = \frac{s(p \cup q \cup r)}{s(p)}$$

$$c_3 = \frac{s(p \cup q \cup r)}{s(p \cup r)}$$

Considering $s(p) \geq s(p \cup q) \geq s(p \cup q \cup r)$

Thus: $c_1 \geq c_2 \& c_3 \geq c_2$.

Therefore c_2 has the lowest confidence.

- (c) Repeat the analysis in part (b) assuming that the rules have identical support. Which rule has the **highest confidence**?

Answer:

Considering $s(p \cup q) = s(p \cup q \cup r)$

but $s(p) \geq s(p \cup r)$

Thus: $c_3 \geq (c_1 = c_2)$

Either all rules have the same confidence or c_3 has the highest confidence.

11. The *Apriori* algorithm uses a generate-and-count strategy for deriving frequent itemsets. Candidate itemsets of size $k + 1$ are created by joining a pair of frequent itemsets of size k (this is known as the candidate generation step). A candidate is discarded if any one of its subsets is found to be infrequent during the **candidate pruning step**. Suppose the *Apriori* algorithm is applied to the data set shown in Table 4.22 with $\text{minsup} = 30\%$, i.e., any itemset occurring in less than 3 transactions is considered to be infrequent.

Table 4.22. Example of market basket transactions.

Transaction ID	Items Bought
1	{a, b, d, e}
2	{b, c, d}
3	{a, b, d, e}
4	{a, c, d, e}
5	{b, c, d, e}
6	{b, d, e}
7	{c, d}
8	{a, b, c}
9	{a, d, e}
10	{b, d}

- (a) Draw an itemset lattice representing the data set given in Table 4.22. Label each node in the lattice with the following letter(s):

- **N**: If the itemset is not considered to be a candidate itemset by the *Apriori* algorithm. There are two reasons for an itemset not to be considered as a candidate itemset: (1) it is not generated at all during the candidate generation step, or (2) it is generated during the candidate generation step but is subsequently removed during the **candidate pruning step** because one of its subsets is found to be infrequent.
- **F**: If the **candidate** itemset is found to be **frequent** by the *Apriori* algorithm.
- **I**: If the **candidate** itemset is found to be **infrequent** after support counting.

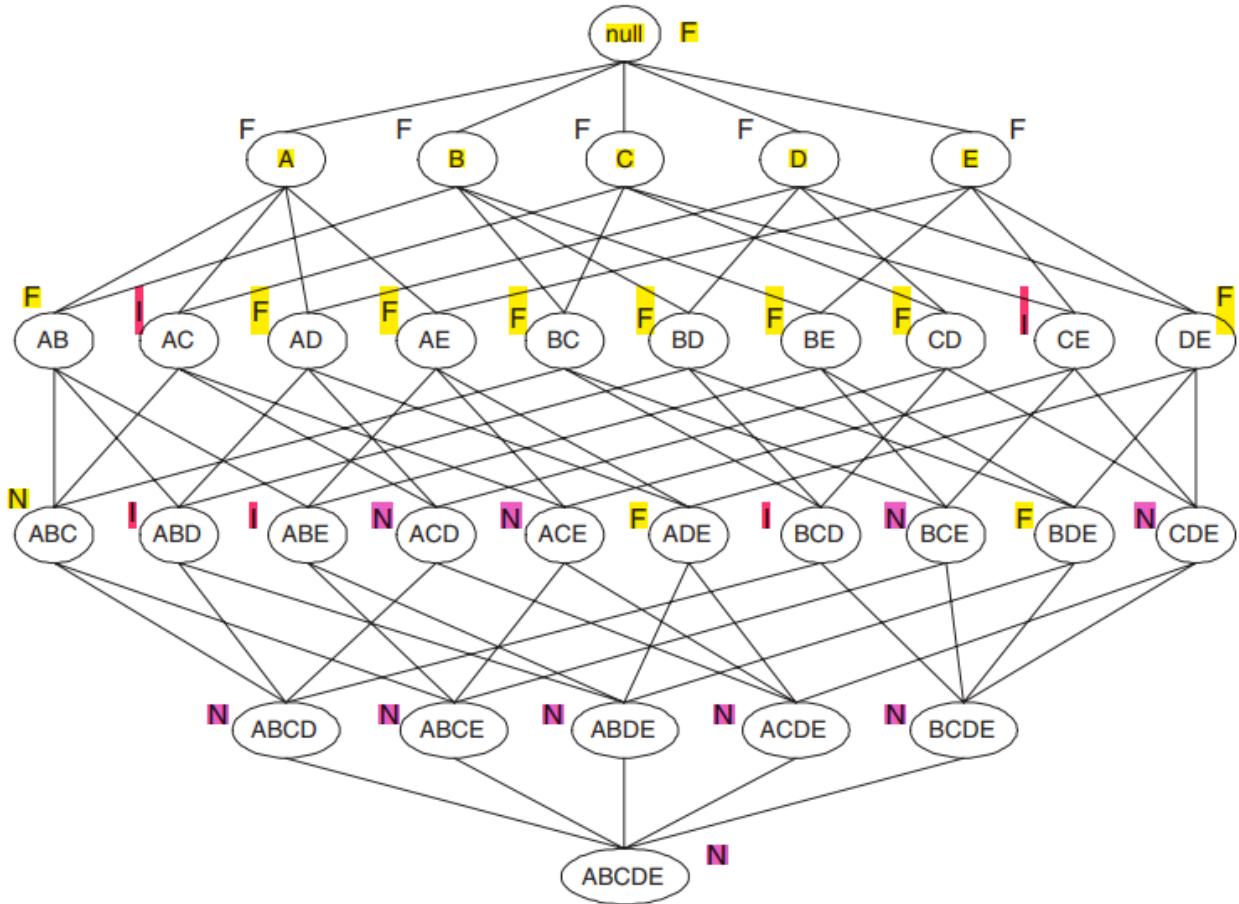


Figure 6.1. Solution.

- (b) What is the percentage of frequent itemsets (with respect to all itemsets in the lattice)?

Answer:

Percentage of frequent itemsets = $16/32 = 50.0\%$ (including the null set).

- (c) What is the pruning ratio of the *Apriori* algorithm on this data set? (Pruning ratio is defined as the percentage of itemsets not considered to be a candidate because (1) they are not generated during candidate generation or (2) they are pruned during the candidate pruning step.)

Answer:

Pruning ratio is the ratio of N to the total number of itemsets. Since the count of $N = 11$, therefore pruning ratio is $11/32 = 34.4\%$.

- (d) What is the false alarm rate (i.e, percentage of candidate itemsets that are found to be infrequent after performing support counting)?

Answer:

False alarm rate is the ratio of I to the total number of itemsets. Since the count of $I = 5$, therefore the false alarm rate is $5/32 = 15.6\%$.

Table 6.4. Example of market basket transactions.

Transaction ID	Items Bought
1	$\{a, b, d, e\}$
2	$\{b, c, d\}$
3	$\{a, b, d, e\}$
4	$\{a, c, d, e\}$
5	$\{b, c, d, e\}$
6	$\{b, d, e\}$
7	$\{c, d\}$
8	$\{a, b, c\}$
9	$\{a, d, e\}$
10	$\{b, d\}$

12. The original association rule mining formulation uses the support and confidence measures to prune uninteresting rules.

- (a) Draw a contingency table for each of the following rules using the transactions shown in Table 6.4.

Rules: $\{b\} \rightarrow \{c\}$, $\{a\} \rightarrow \{d\}$, $\{b\} \rightarrow \{d\}$, $\{e\} \rightarrow \{c\}$, $\{c\} \rightarrow \{a\}$.

Answer:

	c	\bar{c}
b	3	4
\bar{b}	2	1

	d	\bar{d}
a	4	1
\bar{a}	5	0

	d	\bar{d}
b	6	1
\bar{b}	3	0

	c	\bar{c}
e	2	4
\bar{e}	3	1

	a	\bar{a}
c	2	3
\bar{c}	3	2

- (b) Use the contingency tables in part (a) to compute and rank the rules in decreasing order according to the following measures.

- i. Support.

Answer:

Rules	Support	Rank
$b \rightarrow c$	0.3	3
$a \rightarrow d$	0.4	2
$b \rightarrow d$	0.6	1
$e \rightarrow c$	0.2	4
$c \rightarrow a$	0.2	4

- ii. Confidence.

Answer:

Rules	Confidence	Rank
$b \rightarrow c$	3/7	3
$a \rightarrow d$	4/5	2
$b \rightarrow d$	6/7	1
$e \rightarrow c$	2/6	5
$c \rightarrow a$	2/5	4

- 8.7 The following table consists of training data from an employee database. The data have been generalized. For example, “31 … 35” for *age* represents the age range of 31 to 35. For a given row entry, *count* represents the number of data tuples having the values for *department*, *status*, *age*, and *salary* given in that row.

<i>department</i>	<i>status</i>	<i>age</i>	<i>salary</i>	<i>count</i>
sales	senior	31…35	46K…50K	30
sales	junior	26…30	26K…30K	40
sales	junior	31…35	31K…35K	40
systems	junior	21…25	46K…50K	20
systems	senior	31…35	66K…70K	5
systems	junior	26…30	46K…50K	3
systems	senior	41…45	66K…70K	3
marketing	senior	36…40	46K…50K	10
marketing	junior	31…35	41K…45K	4
secretary	senior	46…50	36K…40K	4
secretary	junior	26…30	26K…30K	6

Let *status* be the class label attribute.

- (a) How would you modify the basic decision tree algorithm to take into consideration the *count* of each generalized data tuple (i.e., of each row entry)?
 - (b) Use your algorithm to construct a decision tree from the given data.
 - (c) Given a data tuple having the values “systems,” “26…30,” and “46–50K” for the attributes *department*, *age*, and *salary*, respectively, what would a naïve Bayesian classification of the *status* for the tuple be?
- (c) Given a data tuple having the values “systems”, “26…30”, and “46–50K” for the attributes *department*, *age*, and *salary*, respectively, what would a naïve Bayesian classification of the *status* for the tuple be?
 $P(X|\text{senior}) = 0; P(X|\text{junior}) = \frac{31}{113} \times \frac{46}{113} \times \frac{20}{113} = 0.018$. Thus, a naïve Bayesian classification predicts “junior”.

4. Discuss issues to consider during *data integration*.

Answer:

Data integration involves combining data from multiple sources into a coherent data store. Issues that must be considered during such integration include:

- **Schema integration:** The metadata from the different data sources must be integrated in order to match up equivalent real-world entities. This is referred to as the entity identification problem.
- **Handling redundant data:** Derived attributes may be redundant, and inconsistent attribute naming may also lead to redundancies in the resulting data set. Also, duplications at the tuple level may occur and thus need to be detected and resolved.
- **Detection and resolution of data value conflicts:** Differences in representation, scaling or encoding may cause the same real-world entity attribute values to differ in the data sources being integrated.

Answer:

The various methods for handling the problem of missing values in data tuples include:

- (a) **Ignoring the tuple:** This is usually done when the **class label is missing** (assuming the mining task involves **classification or description**). This method is not very effective unless the tuple contains **several attributes with missing values**. It is especially poor when the percentage of missing values per attribute varies considerably.
- (b) **Manually filling in the missing value:** In general, this approach is **time-consuming** and may not be a reasonable task for **large data sets** with many missing values, especially when the value to be filled in is not easily determined.
- (c) **Using a global constant to fill in the missing value:** Replace **all missing attribute values** by the same constant, such as a label like "*Unknown*," or $-\infty$. If missing values are replaced by, say, "*Unknown*," then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common — that of "*Unknown*." Hence, although this method is simple, **it is not recommended**.
- (d) **Using a measure of central tendency for the attribute**, such as the **mean** (for symmetric numeric data), the **median** (for asymmetric numeric data), or the **mode** (for nominal data): For example, suppose that the **average income of *AllElectronics* customers** is \$28,000 and that the data are symmetric. Use this value to replace any missing values for *income*.
- (e) **Using the attribute mean for numeric (quantitative) values or attribute mode for nominal values**, for all samples belonging to the **same class** as the given tuple: For example, if **classifying customers** according to ***credit_risk***, replace the missing value with the average income value for customers in the **same credit risk category** as that of the given tuple. If the data are **numeric** and **skewed**, use the **median value**.
- (f) **Using the most probable value to fill in the missing value:** This may be determined with **regression**, **inference-based tools using Bayesian formalism**, or **decision tree induction**. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for *income*.

3. Exercise 2.2 gave the following data (in increasing order) for the attribute *age*: 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

- (a) Use **smoothing by bin means** to smooth the above data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.
- (b) How might you determine **outliers** in the data?
- (c) What other methods are there for ***data smoothing***?

Answer:

- (a) Use **smoothing by bin means** to smooth the above data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.

The following steps are required to smooth the above data using smoothing by bin means with a bin depth of 3.

- **Step 1:** Sort the data. (This step is not required here as the data are already sorted.)
- **Step 2:** Partition the data into equidepth bins of depth 3.

Bin 1: 13, 15, 16	Bin 2: 16, 19, 20	Bin 3: 20, 21, 22
Bin 4: 22, 25, 25	Bin 5: 25, 25, 30	Bin 6: 33, 33, 35
Bin 7: 35, 35, 35	Bin 8: 36, 40, 45	Bin 9: 46, 52, 70

- **Step 3:** Calculate the arithmetic mean of each bin.
 - **Step 4:** Replace each of the values in each bin by the arithmetic mean calculated for the bin.
- | | | |
|------------------------------|------------------------------|------------------------------|
| Bin 1: $142/3, 142/3, 142/3$ | Bin 2: $181/3, 181/3, 181/3$ | Bin 3: $21, 21, 21$ |
| Bin 4: 24, 24, 24 | Bin 5: $262/3, 262/3, 262/3$ | Bin 6: $332/3, 332/3, 332/3$ |
| Bin 7: 35, 35, 35 | Bin 8: $401/3, 401/3, 401/3$ | Bin 9: 56, 56, 56 |

This method smooths a sorted data value by consulting to its "neighborhood". It performs **local smoothing**.

- (b) How might you determine **outliers** in the data?

Outliers in the data may be detected by **clustering**, where similar values are organized into groups, or 'clusters'. Values that fall outside of the set of clusters may be considered outliers. Alternatively, a **combination of computer and human inspection** can be used where a predetermined data distribution is implemented to allow the computer to identify possible outliers. These possible outliers can then be verified by human inspection with much less effort than would be required to verify the entire initial data set.

- (c) What other methods are there for **data smoothing**?

Other methods that can be used for data smoothing include alternate forms of **binning** such as **smoothing by bin medians** or **smoothing by bin boundaries**. Alternatively, **equiwidth bins** can be used to implement any of the forms of binning, where the interval range of values in each bin is constant. Methods other than binning include **using regression techniques to smooth the data** by fitting it to a function such as through linear or multiple regression. Also, **classification techniques** can be used to implement concept hierarchies that can smooth the data by rolling-up lower level concepts to higher-level concepts.

9. Suppose a group of 12 *sales price* records has been sorted as follows:

5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215.

Partition them into three bins by each of the following methods.

- equal-frequency (equidepth) partitioning
- equal-width partitioning
- clustering

Answer:

- (a) **equal-frequency (equidepth) partitioning**

Partition the data into equidepth bins of depth 4:

Bin 1: 1: 5, 10, 11, 13 Bin 2: 15, 35, 50, 55 Bin 3: 72, 92, 204, 215

- (b) **equal-width partitioning**

Partitioning the data into 3 equi-width bins will require the width to be $(215 - 5)/3 = 70$. We get:

Bin 1: 5, 10, 11, 13, 15, 35, 50, 55, 72 Bin 2: 92 Bin 3: 204, 215

- (c) **clustering**

Using **K-means clustering** to partition the data into three bins we get:

Bin 1: 5, 10, 11, 13, 15, 35 Bin 2: 50, 55, 72, 92 Bin 3: 204, 215

6. Use the methods below to *normalize* the following group of data:

$$200, 300, 400, 600, 1000$$

- (a) min-max normalization by setting $\min = 0$ and $\max = 1$
- (b) z-score normalization
- (c) z-score normalization using the mean absolute deviation instead of standard deviation
- (d) normalization by decimal scaling

Answer:

- (a) *min-max normalization* by setting $\min = 0$ and $\max = 1$ get the new value by computing

$$v'_i = \frac{v_i - 200}{1000 - 200} (1 - 0) + 0.$$

The normalized data are:

$$0, 0.125, 0.25, 0.5, 1$$

- (b) In *z-score normalization*, a value v_i of A is normalized to v'_i by computing

$$v'_i = \frac{v_i - \bar{A}}{\sigma_A},$$

where

$$\bar{A} = \frac{1}{5}(200 + 300 + 400 + 600 + 1000) = 500,$$

$$\sigma_A = \sqrt{\frac{1}{5}(200^2 + 300^2 + \dots + 1000^2) - \bar{A}^2} = 282.8.$$

The normalized data are:

$$-1.06, -0.707, -0.354, 0.354, 1.77$$

14. You are given a data set with 100 records and are asked to cluster the data. You use K-means to cluster the data, but for all values of K , $1 \leq K \leq 100$, the K-means algorithm returns only one non-empty cluster. You then apply an incremental version of K-means, but obtain exactly the same result. How is this possible? How would single link or DBSCAN handle such data?
- (a) The data consists completely of duplicates of one object.
 - (b) Single link (and many of the other agglomerative hierarchical schemes) would produce a hierarchical clustering, but which points appear in which cluster would depend on the ordering of the points and the exact algorithm. However, if the dendrogram were plotted showing the proximity at which each object is merged, then it would be obvious that the data consisted of duplicates. DBSCAN would find that all points were core points connected to one another and produce a single cluster.

If we have 100 duplicate records in our dataset, K-means clustering with K=2 to K=100 will not change the result of the algorithm because duplicate records have identical feature values. Therefore, regardless of the value of K, the algorithm will always cluster the duplicate records into the same cluster.

For example, suppose we have a dataset of 200 records, where 100 records are duplicates. If we run K-means clustering with K ranging from 2 to 100, the resulting clusters will look the same for all values of K, as follows:

Cluster 1 Cluster 2	
1	101
2	102
...	...
100	200

In this case, the first 100 records (which are duplicates) will always be clustered together in Cluster 1, while the remaining 100 records will be clustered together in Cluster 2, regardless of the value of K.

It's important to remember that in practice, K-means clustering is not suitable for detecting and handling duplicate records, as it operates solely on the feature values of each record. To handle duplicates, data preprocessing steps such as deduplication should be performed before applying clustering algorithms like K-means.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is another clustering algorithm that can be used to group similar records together. DBSCAN works by grouping records that are close to each other in a high-density region while ignoring records in low-density regions or noise.

When dealing with duplicate records, DBSCAN will treat them as separate individual data points and may form clusters around them based on the density of the surrounding points. However, this behavior may depend on the specific implementation of DBSCAN and the distance metric used.

Suppose we have 100 duplicate records in our dataset and we apply DBSCAN with a Euclidean distance metric and a minimum cluster size of 10. In this case, the algorithm will consider each duplicate record as a separate data point and may form clusters around them based on the density of their neighboring points.

For example, if we have a dataset of 200 records, where 100 records are duplicates, and we run DBSCAN with the parameters mentioned above, the resulting clusters might look something like this:

...
Noise Cluster 1
-----+-----
100 10
100 11
... ...
100 19
...

In this example, all 100 duplicate records are treated as individual data points, and they are not merged into a single cluster. Instead, the algorithm identifies them as noise, since they do not satisfy the minimum cluster size requirement. The remaining non-duplicate records form one or more clusters based on the density of their neighboring points.

Note that the exact behavior of DBSCAN when dealing with duplicate records may vary depending on the distance metric, the minimum cluster size, and other parameters. It's also important to preprocess the data to remove any duplicates before applying clustering algorithms to avoid unintended effects on the clustering results.