Subject:
Year. Month. Date. ( )

اول کار یک Dataset داریم باید عملیات زیر را انجام بدهیم:

Pre Processing: بر اساس شاخص‌های مرکزی پراکندگی کارهای انجام می‌شود و شامل ۴ مرحله زیر می‌باشد.

۱ داده‌های ناقص داریم
۲ داده‌های پرت و نویز

برای داده‌ی ناقص می‌توانیم آن‌ها را حذف کنیم
وقتی داده‌ی ما زیاد باشد یعنی حذف کردن تأثیری در کل دیتا ما ندارد آن را حذف می‌کنیم
یک راه دیگر این است که یک میانگین در نظر بگیریم و به جای داده‌ی ناقص بذاریم.
اینکه کدام راه را انتخاب کنیم: بستگی به مقدار داده‌ی ما دارد
می‌توانیم میانگین یک دسته یا کلاس را قرار دهیم ← مثلاً از روی مهمترین ویژگی‌ها
می‌خواهیم که داده‌ی ما در این کلاس قرار دارد
اینکه میانگین را قرار دهیم یا چیزی دیگر ← یک چیز تجربی است

داده‌ی پرت را حذف می‌کنیم که این هم تجربی است برحسب میزان بستگی دارد

روش‌های تشخیص داده‌ی پرت:
۱ توزیع نرمال
۲ IQR چارک
۳ روش تشخیص مبتنی بر فاصله برای جاهایی که کلاستربندی می‌کنیم خوب است.

۳ یک داده داریم و می‌خواهیم ببینیم پرت است یا نه؟
فاصله این دیتا با همین همسایه را حساب می‌کنیم. (با P همسایه)
اگر فاصله آن با همه‌ی داده‌ها بیشتر از یک مقدار threshold(d) باشد می‌گوییم یک داده پرت است. d و P به صورت تجربی تعیین می‌شوند.

۳۵ داده

۱۰ نفر P همسایه
۱۰
۱۵
۲۰
۱۳
۷
⋮

> threshold (d)

Subject:
Year. Month. Date. ( )

مراحل آماده سازی داده:

مراحل preprocessing:
1. Data Cleaning ← 1. نقص Miss 2. پرت و نویز
2. Data Intigration
3. Data Transformation
4. Data Reduction

داده کاوی: معنی نزدیک، IQR، دسته بندی، خوشه بندی، رگرسیون

دسته بندی ← هدف: حذف داده نویز و پرت / کاهش سایز داده

مثال: {۳, ۲, ۱, ۵, ۴, ۳, ۱, ۷, ۵, ۳}

اول عدد داده ها را مرتب کنیم:
{۱, ۱, ۲, ۳, ۳, ۴, ۵, ۵, ۷}

بعد از آن دسته ها را انتخاب کنیم
{۱,۱,۲} {۳,۳,۳} {۴,۵,۵,۷}

بعد برای هر دسته یک شاخص می گذاریم یعنی به جای هر دسته شاخص را می گذاریم.
شاخص: میانگین {۱,۳ و ۱,۳ و ۱,۳} {۳,۳,۳} {۵,۲۵, ۵,۲۵, ۵,۲۵, ۵,۲۵}
مد {۱,۱,۱} {۳,۳,۳} {۵,۵,۵,۵}
مقادیر کناری دسته {۱,۱,۲} {۳,۳,۳} {۴,۴,۴,۷}

مهم این است که چند تا دسته درست کنیم و طول هر دسته چقدر باشد؟

$k = 1 + 3.3 \log n$

k ← تعداد دسته ها
n ← تعداد کل نمونه ها

Min Error = | میانگین یا میانه دسته − مقادیر یک دسته |
یا مد

به جا قدر مطلق ممکن است مجذور هم گفته شود.

Subject:
Year. Month. Date. ( )

مثال: {۶, ۸, ۱, ۲, ۹, ۲, ۲, ۸, ۱, ۵}

دسته بندی به شکل

الف - {۱, ۱, ۲} {۲, ۲, ۵} {۶, ۸, ۸, ۹}

{۱, ۱, ۱} {۲, ۲, ۲} {۸, ۸, ۸, ۸}

مد {۱, ۲, ۸}

Error = {|۱-۱| + |۱-۱| + |۲-۱| + |۲-۲| + |۵-۲|

+ |۶-۸| + |۸-۸| + |۸-۸| + |۹-۸| = ۷

دسته بندی به شکل

ب = {۱, ۱, ۲, ۲, ۲} {۵, ۶} {۸, ۸, ۹}

مد 2 5 8

Error = |1-2| + |1-2| + |2-2| + |2-2| + |2-2| + |5-5|

+ |6-5| + |8-8| + |8-9| = 4

Error کمتر شد پس دسته بندی به شکل ب بهتر است.

خوشه بندی: داده ای مشابه به هم را در یک دسته بگذاریم و متغیر خروجی یا label یا کلاس نداریم.

کاربرد: مثلاً مشتری های شبیه به هم را در یک دسته قرار میدهیم.

مشتری جدید را در یکی از این دسته ها قرار می دهیم (شباهت بین آن و بایستی را پیدا می کنیم)

و پیش بینی کنیم مشتری جدید چه چیزهایی می خرد.

کاربرد دیگر: پردازش تصویر، تشخیص الگو، تشخیص مشتری ها

اشیا شبیه به هم را در یک دسته قرار می دهیم.

مثلاً میوه های شبیه به سیب را در یک دسته قرار میدهد

اگر داده ها برچسب بود و به هیچ کدام از دسته ها شبیه نبود می فهمیم داده پرت است

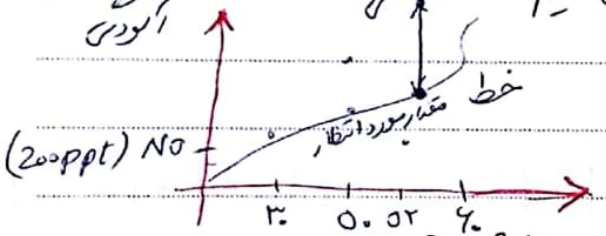
پس برای تشخیص داده های پرت کاربرد دارد

Subject:
Year. Month. Date. ( )

رگرسیون : برای اینکه مقدار یک متغیر را براساس متغیر دیگر پیش بینی کنیم.
مثلاً رابطه بین دما و آبوردگی.
رگرسیون برای دو متغیری استفاده می‌شود که این دو متغیر با هم ارتباط داشته باشند
یعنی ارتباط مثبت یا ارتباط منفی اما ارتباط صفر نباشد (صفر یعنی مستقل)
پس حتماً بین این دو متغیر ضریب همبستگی داریم. (Corrolation - Covariance)

رگرسیون
- خطی
  - غیر خطی ← یک جوری تبدیل به خطی می‌کنند.
- چندگانه ← رابطه بین یک متغیر و چند تا متغیر
  پیش بینی مقدار یک متغیر براساس چند تا متغیر

دما	آبوردگی



می‌خواهیم رابطه بین آبوردگی و دما را بررسی کنیم.
یک خط پیدا می‌کنیم. $y = \alpha x + \beta$
بعضی از داده‌ها روی خط نمی‌افتند
که به فاصله این داده‌ها تا خط Error
می‌گوییم. خطی را انتخاب می‌کنیم که Error نداشته باشد.

$y = \alpha_1 x_1 + \alpha_2 x_2 + \beta$ → رگرسیون چندگانه

$$\alpha = \bar{y} + \beta \bar{x} \qquad \beta = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Subject:
Year. Month. Date. ( )

Integeration یکپارچه سازی:

چندین دیتابیس داریم که می‌خواهیم آن‌ها را یکپارچه کنیم. مثلاً ویژگی قد در یک دیتابیس برحسب foot و در یک دیتابیس برحسب سانتی‌متر است. ← ناسازگاری

یا داده‌های تکراری داشته باشیم. پس در یکپارچه سازی هدف حذف افزونگی و ناسازگاری است ← تبدیل میشه به انبار داده.

حالا فرض کنیم یک انبار داده داریم. مثلاً داده‌های مابین ۵۰ تا ۵۵- هستند اما بعضی متدهای داده‌کاوی نیاز دارند که علاً مابین [۱و۰] باشد. ← نرمالایز

Transformation تغییر شکل داده: نرمالایز کردن یا استاندارد سازی

روش‌های استاندارد سازی:

① حرکت نقطه اعشار   مثلاً $data: 800$ تا -3.76

قدر مطلق بزرگترین data $= 800 \cong 1000 \rightarrow 10^3$ $\frac{\text{همه اعداد}}{10^3}$ → $\frac{\text{اعداد}}{\text{توان } 10}$

② $\frac{\text{عدد} - Min}{Max - Min}(NewMax - NewMin) + NewMin$

می‌خواهیم اعداد در بازه [Min و MAX] قرار بگیرند.
NewMin   NewMAX

③ $\frac{\text{میانگین} - \text{عدد}}{\text{انحراف معیار}}$

معمولاً قبل از اعمال شبکه عصبی نرمال سازی انجام میدهیم.

data Reduction ← داده‌ها خیلی زیادند. یک دیتابیس خیلی بزرگ

حالا می‌خواهیم یک سری داده‌ها را حذف کنیم.

یا مثلاً داده‌های ما ویژگی‌های خیلی زیادی دارند. مثلاً دیتابیس از دانشجویان داریم: آدرس، شماره دانشجویی، اسم، معدل و... وقتی می‌خواهیم سطح علمی دانشجویان را حساب کنیم ویژگی‌های آدرس و اسم و... تأثیر ندارند پس آن‌ها را حذف می‌کنیم.

حالا مثال دیگر می‌خواهیم تأثیر بینایی را بررسی کنیم. رنگ چشم: قهوه‌ای - قهوه‌ای روشن - قهوه‌ای تیره قهوه‌ای خیلی تیره ← می‌توانیم دو دسته در نظر بگیریم ← قهوه‌ای روشن و قهوه‌ای تیره

Subject:
Year. Month. Date. ( )

۱- کاهش تعداد ویژگی‌ها
۲- کاهش تعداد نمونه‌ها ←
۳- کاهش مقادیر یک ویژگی

۱- انتخاب تصادفی که می‌تواند با جایگذاری یا بدون جایگذاری باشد
۲- نظم ← مثلاً از ۱۰ تای اول یکی را انتخاب کن از ۱۰ تای دوم یکی
۳- براساس کلاس‌های مختلف انتخاب کنیم:
مثلاً می‌خواهیم از دخترها و پسرها به یک تعداد انتخاب کنیم
← جنسیت در بررسی‌ها اثر می‌گذارد.

↓

مثال تبدیل رنگ چشم به ۳ دسته

کاهش مقادیر یک ویژگی: فرض کنید سن از ۱۳۰-۵ سال باشد
می‌خواهیم دسته‌بندی کنیم. دو تا روش داریم ← کای دو χ^2 / آنتروپی
گسسته سازی کنیم ← روش‌ها گسسته سازی اند.

ID	ویژگی سن	class نوع
1	1	A
2	3	B
3	7	A
4	8	A
5	9	A
6	11	B
7	23	B
8	37	A
9	39	B
10	45	A
11	46	A
12	50	A

این دیتابیت ما ۲ تا نوع کلاس می‌تواند دختر یا پسر باشد.
خودمان شروع به دسته‌بندی می‌کنیم.
تعداد این دسته‌ها زیاد است.
بعضی از دسته‌ها را می‌توان ادغام کرد.
می‌خواهیم ببینیم چطوری می‌توان این دسته‌ها را با هم ادغام کرد.

[0,2) [2,5) [5,7.5) [7.5,8.5) [8.5,10)
[10,17) [17,30) ,---

اسم کلاس‌ها را می‌نویسیم
اسم دسته‌ها را می‌نویسیم
نگاه می‌کنیم از این کلاس در این دسته چند نفر هستند.

	A	B	
[7.5,8.5)	$A_{11}=1$	$A_{12}=0$	→ جمع = $R_1=1$
[8.5,10)	$A_{21}=1$	$A_{22}=0$	→ جمع = $R_2=1$
	جمع $C_1=2$	$C_2=0$	$N=2$

بعد این‌ها را سطری و ستونی با هم جمع می‌کنیم.

Subject:
Year. Month. Date. ( )

فرمول کای دو:

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} (A_{ij} - E_{ij})^2 / E_{ij}$$

i: دو تا دسته داریم
j: تعداد نمونه‌ها در تا. امین دسته و ل - امین کلاس
A_{ij}: فراوانی مورد انتظار

می‌خواهیم ببینیم دو تا دسته را باهم ادغام کنیم یا نه؟

N: تعداد کل نمونه‌ها

$$E_{ij} = R_i \times C_i / N$$

درجه آزادی = (تعداد کلاس‌ها − 1) ← در این مثال ← درجه آزادی = 1

یک جدول کای دو داریم که بر اساس درجه آزادی مقدار کای دو در آن نوشته شده

$$\chi^2 = (1-1)^2/1 + (0-0/1)^2/0/1 + (1-1)^2/1 + (0-0/1)^2/0/1$$

$= 0/2 < 2.706$ ← مقدار χ^2 بر اساس جدول

چون مقدار χ^2 به دست آمده کمتر از مقدار χ^2 جدول است پس باهم ادغام شدند

پس این دو دسته را ادغام می‌کنیم. به همین قدر این کار را برای دسته‌های مختلف تکرار می‌کنیم تا به این نتیجه برسیم که هیچ دو دسته‌ای باهم ادغام نمی‌شوند

(7.5 از 10)

برای این مثال ←

$E_{11} = 1$ $\quad E_{12} = 0 \Rightarrow 0/1$ (صفر نباشد)

$E_{21} = 1$ $\quad E_{22} = 0 \Rightarrow 0/1$

گسسته سازی صفتی بر آنتروپی: مجموعه D را به دو مجموعه D_1 و D_2 می‌شکنیم

حال می‌خواهیم ببینیم این تقسیم بندی خوب است یا نه؟

D_1 | D_2

SP: نقطه انفصال

$$Info_A(D) = \frac{|D_1|}{|D|} \times Entropy(D_1) + \frac{|D_2|}{|D|} \times Entropy(D_2)$$

Subject:
Year. Month. Date. ( )

احتمال کلاس K-ام در D

$$Entropy = -\sum_{i=1}^{K} P_i \log_2 P_i$$

مشتری ۱۰ ساله سنی از ۱ تا 59 است
ما آن را به دو دسته تقسیم می‌کنیم
از ۱ تا 30 و از 31 تا 59
30 نقطه انفصال می‌خواهیم ببینیم آیا 30 نقطه خوبی است؟

خب ← حذف ستون‌ها (حذف بعضی از ویژگی‌ها) خوب است و پرکاربرد
کاهش حجم نمونه صحیح بلکه به کار میره اما کاربردی تر از همه حذف ویژگی‌هاست
اما روش دسته بندی صحیح نیست ← کاربردی نیست (کاهش مقادیر یک ویژگی)

حذف بعضی از ویژگی‌ها ← واضح است که بعضی از ویژگی‌ها در تعیین خروجی یا پیش‌بینی‌ها
اثری ندارند.
می‌خواهیم بر اساس ویژگی X و
ویژگی Y نوع کلاس را مشخص کنیم.
می‌خواهیم ببینیم برای تعیین کلاس
آیا هر دو ویژگی را نیاز داریم یا یک
ویژگی می‌توانیم نوع کلاس را تشخیص دهیم.

ویژگی X	ویژگی Y	نوع کلاس
3	7	A
2	9	B
6	6	A
5	5	A
8	7	B
4	9	A

$$\text{فرمول} = \frac{|\text{میانگین A} - \text{میانگین B}|}{\sqrt{\frac{\text{واریانس A}}{N_1} + \frac{\text{واریانس B}}{N_2}}}$$

N_1 و N_2 تعداد هرکدام

برای ردیف x این فرمول را حساب
و برای ردیف y این فرمول را حساب
می‌کنیم.

$X_A = \{3, 6, 5, 4\}$ ← x‌هایی که در کلاس A هستند $Mean(X_A) = 4.5$ $Var(X_A) = 1.25$

$X_B = \{2, 8\}$ ← x‌های کلاس B هستند $Mean(X_B) = 5$ $Var(X_B) = 9$

$$\text{فرمول} = \frac{4.5 - 5}{\sqrt{\frac{1.25}{4} + \frac{9}{2}}} = 0.2279 < 0.5$$

Subject:
Year. Mouth. Date. ( )

این مقدار که بدست آمد آن را با یک مقدار ترشولد مقایسه می‌کنیم که این مقدار آستانه را خودمان مشخص می‌کنیم که اینجا گفتیم ۰/۵ باشد.
حالا که این مقدار کمتر از ۰/۵ باشد آن دیزگی را حذف می‌کنیم. یعنی مقدار x در تعیین نوع کلاس نقشی ندارد و با دیزگی y نوع کلاس تعیین می‌شود.

$$Y_A = \{7, 6, 5, 9\} \Rightarrow Mean(Y_A) = 6.75 \quad Var(Y_B) = 2.20$$

$$Y_B = \{9, 7\} \Rightarrow Mean(Y_B) = 8 \quad Var(Y_B) = 1$$

$$\frac{6.75 - 8}{\sqrt{\frac{2.20}{4} + \frac{1}{2}}} = 1.2198 > 0.5$$

$$= ۱,۰۲۲۷$$

حذف بعضی از ستون‌ها ← بررسی تمام دیزگی‌ها به صورت مجزا
روش آنتروپی ←

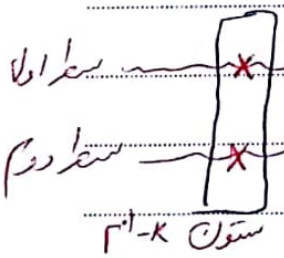
روش آنتروپی: می‌خواهیم ببینیم دو تا سطر از دیتاست چقدر با هم تفاوت دارند.

$$S_{ij} = e^{\alpha D_{ij}} \qquad \alpha = 0.5$$

α به صورت تجربی تعیین می‌شود اما ۰/۵ بهترین حالت است.

برای داده‌های عددی

$$D_{ij} = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk}) / (max_k - min_k)^2}$$



مقدار این ستون − مقدار این ستون / max ستون − min ستون

برای داده‌های که عددی نیستند

$$S_{ij} = \left(\sum_{k=1}^{n} \left(|x_{ik} = x_{jk}| \right) / n \right)$$

Subject:

Year. Month. Date. ( )

$$Enthropy = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[S_{ij} \times \log S_{ij} + (1 - S_{ij}) \times \log (1 - S_{ij}) \right]$$

با استفاده از روش‌های AI هوش مصنوعی این تابع آنتروپی را تابع هدف قرار می‌دهیم هرچه آنتروپی کمتر باشد بهتر است.

در داده کاوی با داده و Data سروکار داریم، روی آن ها کار انجام میدهیم
و یک نتیجه (Result) میگیریم.

Subject:
Year: Month: Day: ( ) page: ( )

Data set مجموعه داده

چندین مدل برای مجموعه داده داریم:

۱- رکورد Record: داده ها را به صورت رکورد ذخیره می کنیم.
رکورد را می توان به ۳ نوع تقسیم کرد:
الف) ماتریس: به صورت ماتریسی از داده ها وجود دارد.
ب) اسناد متنی: به عنوان مثال یک متن داریم و یک سری کلمات که تعداد تکرار آن ها زیاد است را استخراج می کنیم و می گوییم هر کلمه چند بار در این متن ها تکرار شده است.

کلمه / متن	تیم	مسابقات
سند ۱	۵	۱
سند ۲	۰	۲
سند ۳	۱	۰

ج) تراکنشی ← Transaction
در آن یک سری ID و Item وجود دارد.
مثلاً در یک فروشگاه ←

TID	item
1	شیر
2	شیر
3	شیر و کره

۲- گراف و شبکه: گراف مثل وب است.
می توان هر صفحه از وب را یک Node یا گره در نظر گرفت و با لینک آن ها را به هم متصل کرد.
کاربرد گراف و شبکه را می توان در وب، شبکه های اجتماعی، ساختار مولکول ها و ... مثلاً DNA
مثلاً در ساختار مولکول ها از روی Data یک سری الگو به دست می آورند و سعی می کنند که آن را فرمول بندی یا ریاضی بیان کنند.

Soroush

۳- ترتیبی یا ordered:
جاهایی که ترتیب داده ها مهم است مثل متن و ویدیو
بعضی مواقع زمان هم مهم است که به آن ها وابسته به زمان (Temporal) می گویند

۴- فضایی یا Spatial: که مجموعه نقشه هایی است که باید نقشه دارند
کاربرد آن در نقشه، تصاویر ماهواره ای است.

Data ویژگی های آن:
به ویژگی Attribute می گویند

۱- ابعاد داده: ویژگی که داده را به لحاظ ابعاد داده می شناسیم.
در هرکدام از موقعیت های پیش ویژگی را با اسامی زیر نشان می دهیم:

ویژگی در دیتا	انبار داده Data warehouse	ژن لرنینگ	آمار
Attribute	Dimention	Feature	متغیر Variable

۲- پراکندگی داده یا Sparcity یا تنک بودن دیتا: که در آن بیشتر دیتا را نداریم.

۳- رفع ابهامات Data یا Resolution:
داده ها مهم اند چون با مقیاس وابسته اند. برای رفع ابهام با مقیاس (scale) را مشخص کنیم.

۴- توزیع داده یا distribution: ← توضیح ندارد.

اگر یک Table داشته باشیم، سطرها مجموعه دیتا و ستون ها ویژگی ها هستند.
در هر Dataset به هر رکورد یک شی داده می گویند (Data object)
که به آن entity یا موجودیت هم می گویند.

Soroush

مثلاً در سیستم بیمارستان ← بیمار، دکتر، پرستار و ... موجودیت اند.
در سیستم دانشگاه ← دروس، استاد، دانشجو و ... موجودیت اند.

اسامی دیگر موجودیت ← example _ object _ tuple . instance و Sample

مثلاً ← می‌گویند یک Sample از یک Database

گفتیم که ستون‌های یک Table ویژگی یا Attribute هستند که می‌توانند مقادیر زیر را داشته باشند:

۱- Nominal اسمی: که به آن Category و Name of thing هم می‌گویند.
مثل رنگ مو، وضعیت تاهل، کد پستی، شماره دانشجویی از این نوع اند.
میانه، جمع، تفریق و میانگین برای این نوع ویژگی‌ها معنی ندارد.
اما مد برای آن‌ها قابل تعریف است.

۲- Binary دودویی: که یا مقدار صفر و یا مقدار یک می‌گیرد. و دو نوع اند:
۱- باینری متقارن Symmetric: اهمیت صفر و یک یکسان است
۲- باینری نامتقارن Asymmetric: برای ما اولویت دارند و مهم است که کدام یک باشد و کدام صفر. مثلاً اهمیت یک بیشتر از اهمیت صفر باشد.

۳- ترتیبی ordiral:

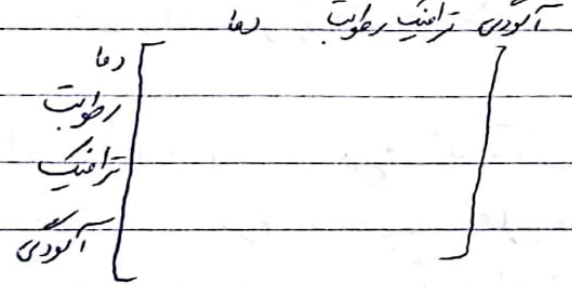
۴- Quantitative عددی:
۱- interval فاصله: فاصله بین شان یکسان باشد
در اینجا مفهومی به نام صفر نداریم (صفر ذاتی نداریم)
پس نمی‌توانیم بگوییم این چند برابر آن است.

۲- Ratio (نسبت) : صفر معنی دارد (صفر ذاتی داریم)
و نسبت ها را می توانیم نسبت به هم بگوییم. یعنی می توان مقایسه کرد.

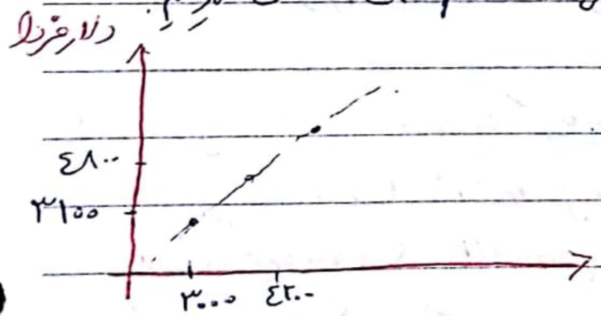
مثال از cov : (کوواریانس) دو جفت عدد.

آلودگی	ترافیک	رطوبت هوا	دمای امروز

می خواهیم ببینیم برای یک نمونه
رابطه بین دما و رطوبت یا دما و آلودگی را پیدا کنیم. (فقط برای یکی از نمونه ها)
می خواهیم ببینیم کدام یک از این ها با هم رابطه دارند؟ پس تشکیل ماتریس کوواریانس می دهیم



همبستگی رابطه بین دو متغیر را با نمودار پراکندگی - Q-plot نشان میدهیم.



الان رابطه خطی شد. ← مثلاً خوش شانس بودیم
می توانیم Q-plot دو فروشگاه را بکشیم و سود آن ها را مقایسه کنیم.

رابطه دیگری نیز برای تعیین همبستگی بین دو متغیر داریم: عددی که بدست می آید بین [۱ ,۱-] است

$$\text{Correlation} = \frac{cov(x,y)}{\sqrt{cov(x,x)\,cov(y,y)}}$$

$cov(x,x) \to \sigma^2(x)$ ، $cov(y,y) \to \sigma^2(y)$

۰ ← همبستگی ندارند. مستقل اند. ۱- ← همبستگی منفی (برعکس)
۱ ← همبستگی مثبت ← (رابطه مستقیم)

ما با پایگاه داده های خیلی بزرگ کار می کنیم.

آیا با روش های آماری می توان این اطلاعات را بدست آورد؟ خیر

زیرا روش های آماری برای داده های محدود است.

تحلیل دیتا ← روش های آماری

مرکزیت دیتا: اول باید ببینیم این دیتا مرکزیت دارد.

شاخص های مرکزی آماری

باید ببینیم دیتا پراکندگی دارد یا نه. شاخص های آماری ↔ شاخص های پراکندگی آماری

شاخص های مرکزیت داده ← ۱. میانگین $\mu = \bar{x} = \frac{\sum x_i}{n}$

اگر دیتای ما وزن داشته باشد ← $Mean = \mu = \bar{x} = \frac{\text{وزن} \times \text{مقدار}}{\text{مجموع وزن ها}}$

عیب میانگین ← داده های پرت روی آن تاثیر می گذارند.

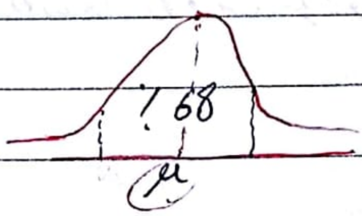
یک داده ی پرت میانگین را جابه جا می کند پس میانگین معیار خوبی برای مرکزیت داده ها نیست.

پس میانگین را اصلاح کنیم ↓

Trimmed Mean: دیتا ها را مرتب می کنیم و ۲ درصد از اول و ۲ درصد از آخر داده ها حذف می کنیم ← این شکل داده های پرت حذف می شوند.

توزیع نرمال: بیشتر افراد جامعه اطراف میانگین قرار می گیرند.

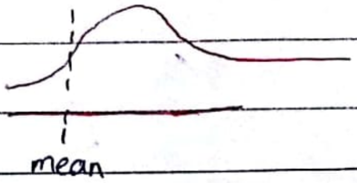
در حالت نرمال میانگین معیار خوبی است

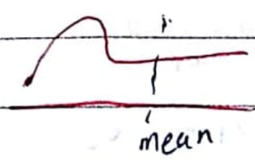


اما ممکن است نمودار چولگی داشته باشد

یعنی به یک سمت منحرف شده باشد.

در این دو حالت میانگین معیار خوبی نیست





N: تعداد کل داده ها

page: ( ) Subject: Year: Month: Day: ( )

شاخص مرکزی ← میانه: Median

داده ها را مرتب می‌چینیم، داده‌ای که در وسط قرار می‌گیرد میانه است.

فرض کنید یک سری داده به شکل زیر داریم:

افراد در این سن
۰ - ۲۰ ...
۲۰ - ۴۰ ...
۴۰ - ۶۰ ...
۶۰ - ۸۰ ...
۸۰ - ۱۰۰ ...

میانه در دسته سوم قرار می‌گیرد

$$\text{میانه} = l_1 + \frac{N/2 + \sum freq_l}{\sum freq_m}$$

l_1: کرانه پایین دسته‌ای که میانه در آن قرار دارد

$\sum freq_l$: فراوانی بازه‌های کوچکتر از بازه میانه (دسته‌های قبلی چقدر فراوانی دارند)

$freq_m$: فراوانی بازه میانه

چون داده‌ها زیاد شد، آن‌ها را دسته‌بندی می‌کنیم

سومین شاخص: مد Mode ?

داده‌ای که تکرار آن از بقیه بیشتر باشد.

Uni Mode ← از هر داده یکی داریم * دسته‌ای که بیشترین تکرار (فراوانی) را دارد دسته مد است

Bi Mode ← از هر داده ۲ تا داریم. یعنی ۲ تا داده‌ی پر تکرار داریم. فراوانی دسته کم‌تر از مد

No Mode ← داده‌های ما مد ندارد.

(حاشیه: uni mode ← تکرار ۱ ؛ No mode ← اگر داده‌ها برابر باشند ؛ فرق No mode و uni mode ؛ طول دسته‌ای که مد در آن است)

* اگر به صورت دسته‌بندی بود مد را چه‌طوری مشخص کنیم:

$$M = l + \frac{D_{befor}}{D_{after}} \times w$$

l: کران پایین ؛ D_{after}: فراوانی دسته بعد از دسته مد

رابطه‌ی بین این ۳ تا: $mean - mode = 3 \times (mean - median)$

پراکندگی داده‌ها چگونه است ← مفهوم چارک Quartile

داده‌ها را مرتب کرده و آن‌ها را به ۴ دسته تقسیم می‌کنیم

Q_1: ۲۵٪ داده‌ها

Q_2: ۵۰ درصد داده‌ها

Q_3: ۷۵ درصد داده‌ها

Q_4: ۱۰۰٪ داده‌ها

Soroush

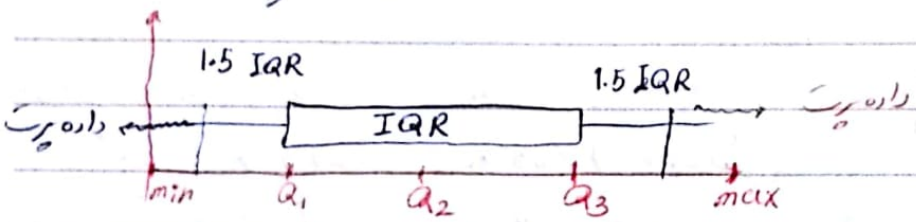
مهمترین توزیعی که از این شاخص ها استفاده می‌کند، توزیع نرمال است.

Subject:
Year Month Day ( ) page: ( )

Inter Quartile Range = IQR = $Q_3 - Q_1$

Five Num: Q_1-Q_2-Q_3-Min-Max ← ۵ تا عدد مهم

فرض کنید داده‌ها از min تا max داریم. ← Box Plot

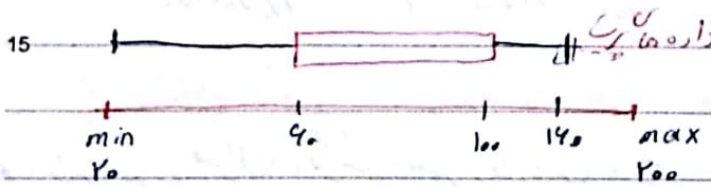


با استفاده از Boxplot چگونه داده‌های پرت را تشخیص دهیم؟

۱٫۵ برابر IQR را حساب کنید.

بعد از Q_3 به اندازه 1.5 برابر IQR را قبول کنید و بقیه را داده پرت بنامید.

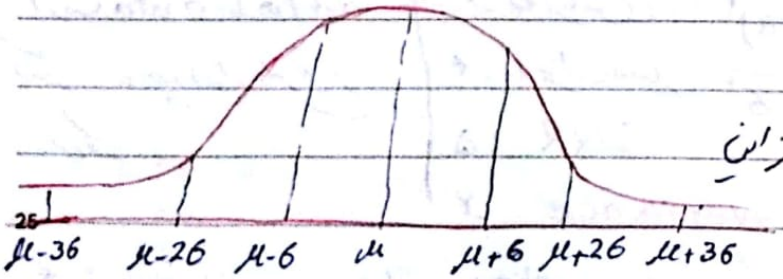
قبل از Q_1 ،



مثال:

$IQR = 100 - 60 = 40$ $\quad 1.5\ IQR = 60$

قبل از Q_1 ← $60 - 60 = 0$ ای داده کمتر از 20 نداریم پس از این طرف داده پرت نداریم. و بعد از Q_3 به اندازه 1.5 IQR جلو می‌رویم یعنی از 160 به بعد داده‌های پرت می‌باشند.



توزیع نرمال:

بنابراین داده‌ای که خارج از این بازه اند داده پرت اند.

۶۸٪ داده‌ها

۹۷٪ داده‌ها

۹۹٪ داده‌ها

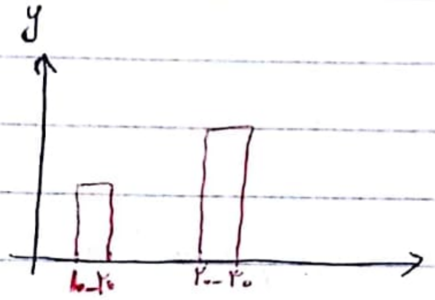
Soroush

اگر دو متغیر مستقل باشند: Cov = ۰

اگر همبستگی دو متغیر مستقیم باشد یعنی با افزایش متغیر اول، متغیر دوم نیز افزایش می‌یابد ← Cov > ۰

اگر رابطه عکس داشته باشند ← Cov < ۰

نمودارها: Histogram میله‌ای



در این تکنیک که داده‌ها را به صورت بازه، بازه در نظر می‌گیرد، گسسته سازی می‌شوند x

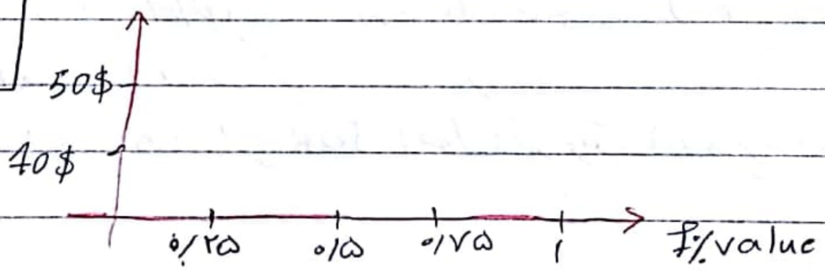
می‌توانیم بفهمیم که نمودارهایی که خیلی بلندند یا خیلی کوتاه‌اند، پرت اند.

قیمت	تعداد فروش
40$	300
50$	120

نمودار Q-Plot : Quantile Plot

محور x آن‌ها f% value می‌باشد.

یک نمودار داریم که از هر کالایی چقدر فروختیم؟



مثال بزنیم؟ قرار شد مثال بیاره مشخص نشد

* برای بررسی میزان تاثیر یک متغیر از واریانس و برای بررسی همبستگی دو متغیر از کوواریانس استفاده می‌کنیم.

شاخص‌های پراکندگی: از یک data تا data دیگر با چه اندازه‌ای تغییر می‌کند.

۱. دامنه تغییرات: Max − Min

۲. انحراف میانگین داده: $\frac{1}{n}\sum_{i=1}^{n}|x_i-\mu|$

قدر مطلق دردسرساز است پس می‌ریم سراغ معیار دیگر ← واریانس

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

۴. انحراف معیار: $\sqrt{\sigma^2}$

۵. IQR

۶. covariance ضریب همبستگی $\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_{x_i})(y_i-\mu_{y_i})$

* مثلاً از روی دمای امروز می‌خواهیم دمای فردا را تشخیص دهیم ← از واریانس استفاده می‌کنیم

اما دما را داریم و رطوبت بعد از روی این‌ها می‌خواهیم دمای فردا را تعیین کنیم ← کوواریانس

مثلاً رابطه رطوبت امروز با دمای فردا چیست؟ اگر

برای هفته بعد یک نرم افزار داشته باشیم

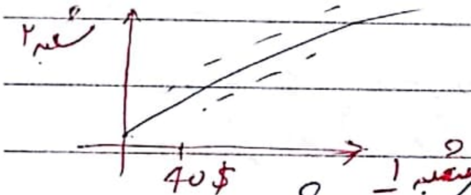
Subject:
Year: Month: Day: ( ) page: ( )

نمودار ? Quantile _ Quantile plot Q_Q Plot

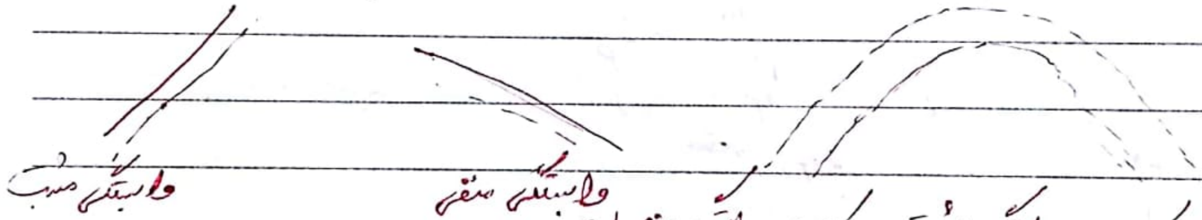
اقلامی که در فروشگاه ۱/۲ به فروش رفته را می‌گیرد
و برای هرکدام یک Plot رسم می‌کند.

در واقع نسبت به هم می‌گوید. مثلاً تعداد فروش 80 دلار در شعبه ۱ معادل می‌شود با تعداد فروش 90$ در شعبه ۲

این مبحث مشابه با رگرسیون است. در رگرسیون می‌خواهیم ارتباط بین دو تا متغیر را بیابیم



وقتی دو تا متغیر را با هم مقایسه می‌کنیم اگر شیب‌ها به شکل زیر باشد:



در یک بازه وابستگی مثبت و در یک بازه وابستگی منفی داریم

مفهوم ? similarity شباهت dissimilarity عدم شباهت

ما باید بتوانیم دسته‌بندی انجام دهیم ← یکی از تکنیک‌های اصلی این درس
داده‌های مشابه را در یک کلاس یا دسته قرار دهیم. چون داده‌ها خیلی زیاداند.
فرض کنید یک پایگاه داده خیلی بزرگ از کل دانشجوهای کشور داریم.
و هر رکورد شامل یک سری صفات است. حالا می‌خواهیم رکوردهای مشابه را
در یک کلاس می‌گذاریم ← بحثی که مطرح می‌شود شباهت از کدام نظر؟

فرض می‌کنیم یک ماتریس داده داریم
n رکورد ، m ویژگی
از روی این ، ماتریس عدم شباهت را تشکیل می‌دهیم.

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}$$

page:( )

Subject:
Year : Month : Day : ( )

ماتریس عدم شباهت:

$$\begin{bmatrix} 0 & & & \\ d_{2,1} & 0 & & \\ d_{3,1} & d_{3,2} & 0 & \\ d_{4,1} & d_{4,2} & d_{4,3} & 0 \end{bmatrix}$$

تفاوت $d_{i,j} = \frac{P - m}{P}$

m: تعداد کل ویژگی هایی که با هم تطابق دارند
P: تعداد کل ویژگی ها

فرض کنید یک پایگاه داده داریم که فقط ویژگی همه داده ها رنگ موی باشد.

اگر دو نفر رنگ موی قهوه ای داشته باشند.

$d_{1,2} = \frac{1 - 1}{1} = 0$ (قهوه ای، قهوه ای)

اگر دو نفر باشند که یکی رنگ موی قهوه ای و دیگری رنگ موی مشکی داشته باشد:

$d_{1,2} = \frac{1 - 0}{1} = 1$ (قهوه ای، مشکی)

شباهت $S_{i,j} = 1 - \frac{P-m}{P} = \frac{m}{P}$

مثال: فرض کنید یک سری داده به شکل زیر داریم:

ID	test1 (اسمی)	test2 (ترتیبی)	test3 (عددی)
1	A	Excellent	45
2	B	Fair	22
3	C	Good	64
4	A	Excellent	28

فرض کنید ماتریس عدم شباهت برای داده های اسمی:

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

test4 (اسمی)
Brown
Blod
Brown
Brown

می توانیم یک ستون دیگر داشته باشیم، در این حالت می توانیم برای دوتا ستون اسمی ماتریس عدم شباهت را بدست بیاریم.

مثلاً $d_{1,2} \rightarrow$ ۲ تا ویژگی ۰ تا مشابه و

$d_{1,2} = \frac{2-0}{2} = 1$

- فرض کنید یک سری داده باینری داریم:

هرجا

$1,1 \rightarrow q$

$0,0 \rightarrow T$

$1,0 \rightarrow r$

$0,1 \rightarrow S$

Subject:
Year : Month : Day : ( ) page : ( )

عدم شباهت بین داده‌ی باینری متقارن: $dis_{(i,j)} = \frac{r+s}{q+r+s+t}$

باینری متقارن ← اهمیت اینکه جواب صفر شود یا یک نیست، یکسان است.

عدم شباهت بین داده‌ی باینری نامتقارن: $dis(i,j) = \frac{r+s}{q+r+s}$

شباهت داده‌ی نامتقارن: $Sim\_Jaccard = \frac{q}{q+r+s}$

فرض کنید دنبال افرادی که hiv دارند می‌گردیم تعداد آن‌ها خیلی کم است ولی پیدا کردن آن‌ها برای ما خیلی مهم است.
مثلاً می‌خواهیم رابطه بین hiv و دیابت را بررسی کنیم این جا برای ما مهم است که هر دو را دارا باشد.
و مثلاً رابطه بین جنسیت و ویژگی‌هایی مثلاً زنان سیگاری این جا معنا ندارد که مردان سیگاری باشند اهمیت ندارد. سوفیا تب

مثال:

	Gender	Fear	Cough	test1 جواب آزمایش (مثبت-منفی)
Jack	M	Y	N	P
Mary	F	Y	N	P
Jim	M	Y	P	N

داده‌ی ما جنس باینری‌اند.

Y ← ۱
N ← ۰
P ← ۱
N ← ۰

$dis(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33$

این داده‌ها نامتقارن‌اند.
(جنسیت را در نظر نمی‌گیریم)
متقارن است

$r = 0$
$s = 1$
$q = 2$

داده‌ی جدول:

test 2	test 3	test 4
N	N ۰	N
N	P ۱	N
N	N	N

Jack, Mary با هم اختلاف زیادی ندارند شباهت تقریباً زیاد است.

dis (Jack, Jim) =

نرم افزار WIKA ، rapid Miner

Data Mining - lecture5 : اسلایدها

در این جلسه می‌خواهیم dissimilarity ، Similarity را برای هر کورد بیابیم.
بخاطر خیلی دلایل می‌خواهیم بفهمیم چقدر دو object می‌توانند شبیه هم باشند
اگر خریدار یک کالا خرید، پیدا کنیم کدام کالای دیگر را امکان دارد بخرد.
خریدارهایی که مشابه خرید می‌کنند را در یک گروه دسته بندی می‌کنیم و تبلیغات را به همه افراد آن دسته می‌فرستیم.
چگونه متن های وب را دسته بندی کنیم. دسته های وب شبیه به هم را پیدا می‌کند.
دنبال تراکنش های متفاوت هستیم که احتمالاً آن ها مشکل دارند.
برای همین مفهوم Similarity و dissimilarity را می‌گوییم.

Similarity : مقدار عددی، اینکه چقدر دو object ما شبیه به هم اند.
دو شی را مقایسه می‌کند و یک مقدار می‌دهد. هرچه نزدیک‌تر باشد، شباهت بیشتر است.
اگر این عدد نزدیک صفر باشد شباهت کم است.
مقدار می‌تواند 1- تا 1 هم باشد

اگر $S(p,q) = S(q,p) \Rightarrow$ یعنی دو object شبیه هم اند.

$$Jac - Similarity = \frac{\text{اشتراک دو مجموعه}}{\text{اجتماع دو مجموعه}}$$

بدست آوردن شباهت بین vector ها:

فرض کنید ۲ تا document داریم. می‌خواهیم Similarity بین این دو را اندازه بگیریم. ← تعداد شباهت هایی که بین این دو وجود دارد را بدست آورده و تقسیم بر تعداد کل کلمات می‌کنیم.

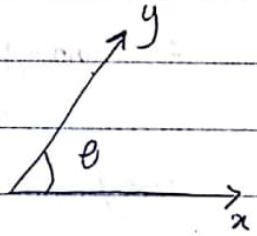
Subject:
Year : Month : Day : ( ) page : ( )

مثلاً ۱/۳۰ متن شده apple . حال یک فرمول پیدا کرده که شباهت بین دو vector را بفهمیم.

apple در متن ۱ ۱۰ بار تکرار شده و در متن دوم 30.

	Apple	Microsoft	obama	Electro
D1	10/30	20/30	0	0
D2	30/90	60/90	0	0

برای پیدا کردن شباهت چندین مورد داریم:



۱- مثلاً داریم: Cos_Similarity

اگر: $\theta = 90$ $\cos\theta = 0$ → عمود برهم → شبیه نیستند

اگر: $\cos\theta = 1$ $\theta = 0$ → برهم منطبق اند → شبیه اند

تصویر هندسی اندازه شباهت:

$\cos(x, y)$ می‌شود شباهت بینشان

برای مقایسه Document ها استفاده از cos رایج‌تر است.

به شرطی می‌توان از cos استفاده کرد که از قبل داده‌ها را نرمال سازی کنیم.

$$\cos(d_1, d_2) = \frac{d_1 * d_2}{\|d_1\| \|d_2\|}$$

۲- distance : مقداری که می‌فهمیم دو object چقدر متفاوت‌اند.

هرچه به صفر نزدیک باشد، شبیه‌ترند.

$(x_{i1}, x_{i2}, \dots, x_{ip})$ و $(x_{j1}, x_{j2}, \dots, x_{jp})$ دو بردار

برای پیدا کردن فاصله چندین روش داریم:

الف) اقلیدسی: فاصله در اقلیدسی $= \sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+\dots}$

ب) منهتن: فاصله منهتن $= |x_{i1}-x_{j1}| + |x_{i2}-x_{j1}| + \dots$?

ج) عین فوکسی: فاصله در عین فوکسی $= \left(|x_{i1}-x_{j1}|^h + |x_{i2}-x_{j2}|^h + \dots \right)^{1/h}$

د) Hamming distance:

تعداد بیت هایی که با هم تفاوت دارند (در دو کد)

چطوری شباهت بین String ها را پیدا می‌کنیم؟ در آنالیز DNA کاربرد دارد

Edit distance: تعداد Delete و Insert هایی که با هم متفاوت هستند که باید شبیه هم باشند.

فصل ۶ کتاب پیدا کردن قوانین انجمنی و الگوهای پرتکرار

مفاهیم: الگوی پرتکرار: مثلاً در یک فروشگاه کالایی هست که بیشتر از همه خریداری شده اند.

چرا الگوهای پرتکرار را باید پیدا کنیم؟

یکی از سوالات ما بعد از چه کالایی، چه کالایی می‌خرم. یعنی توالی در خرید کالا.

مثلاً می‌دانیم هرکس کامپیوتر بخرد، بعد از آن پرینتر می‌خرد.

پس میایم یک تخفیف برای پرینتر می‌زاریم که همراه خریداری میشود.

۱- توالی در خرید آیتم ها

۲- چه آیتم هایی با هم خریداری شده اند. (مثلاً شیر و ماست با هم خریده شده اند پس اینا را کنار هم می‌چینیم)

۳- DNA تاثیر یک چیز بر روی چیز دیگر

Subject:
Year : Month : Day : ( ) page:( )

مثلاً یک دارو روی DNA های مختلف چه تأثیری گذاشته
یا مثلاً چیدمان فروشگاه چه تأثیری در میزان فروش داشته.

۴- طبقه‌بندی اسناد وب مثلاً این خبرگذاری‌ها باید کنار هم باشند
مثلاً هرکسی اخبار ایران را سرچ کرده اخبار افغانستان هم سرچ کرده پس
این دو باید کنار هم باشند.

جاهای زیادی کاربرد دارد مثل سیستم فروش و بازاریابی، علم پزشکی،
پیمایش صفحات وب ← در همه‌ی این جاها نیاز داریم الگوهای پرتکرار را پیدا کنیم.

سوال: اگر کسی کامپیوتر خرید بعد از خرید آن به چه احتمالی آنتی ویروس می‌خرد؟

قانون: باید یک قانون را پیدا کنیم که این اتفاق افتاده.

Computer ⇒ Software [Support = 2% , Confidence = 60%]

$A \Rightarrow B$

باید روابط بسازیم.

فرض کنیم فروشگاه n آیتم به شکل زیر دارد که می‌خواهیم یک کالای A بخریم.
بعد از خرید A، حتماً B را بخریم که اشتراک این دو تهی است.
هیچ وقت سمت راست و چپ نباید اشتراک داشته باشند.

T: مجموعه آیتم‌ها I: هر item

$A \Rightarrow B$ $T = \{I_1, I_2, ..., I_n\}$

$A \cap B = \emptyset$

Soroush

page: ( )

Subject:
Year: Month: Day: ( )

Support: (درجه پشتیبانی) فرض کنید قانون در مجموعه تراکنش‌های D با S درصد support است. در این صورت S درصدی از تراکنش D است که حاوی A و B است و با احتمال P(AUB) نمایش می‌دهند

support(A ⇒ B) ⇒ AUB

یعنی جایی که دو کالا با هم خریده شده‌اند.

Confidence: اگر طرف اول true باشد (یعنی A رخ دهد) چه‌قدر احتمال دارد که طرف دوم نتیجه درست باشد.

$$P(B|A) = \frac{P(AUB)}{P(A)}$$

یعنی اگر A رخ داد چه‌قدر احتمال دارد B رخ دهد؟

این support که می‌گیریم min حالت است.

پس زمان که min=۲٪ یعنی حداقل جاها A و B با هم اند پس حداقل قانون را اینگونه گرفته‌ایم.

اما وقتی می‌گیریم ۹۸٪ confidence باشد. پس داریم قانون ما خیلی سخت می‌گیریم یعنی خیلی کم پیش می‌آید ما می‌گیریم ۹۸٪ باید این دو با هم خریداری شده باشند.

اگر قانون هیچ چیزی پیدا نکند پس این قانون به درد نمی‌خورد.

یا مثلاً قانون ما این بگیریم کسانی که امروز یک کالا خریده‌اند. این احتمال خیلی زیاد است. ← از آن چه می‌خواهیم بدست آوریم ← چیزی به درد نمی‌خورد!

پس زیاد ریتا خارج کردن و کم ریتا خارج کردن به درد نمی‌خورند.

یه چیز متوسط می‌خواهیم نه زیاد سخت‌گیرانه که چیزی پیدا نشه و نه زیاد آزادی بدهیم.

Item set مجموعه آیتم‌ها

تعداد تراکنش‌ها = N ، هر تراکنش = A آیتم

تعداد آیتم‌هایی که می‌توانیم $(2^K - 1) \times N \times A$ ← استخراج کنیم.

Soroush

Subject:
Year : Month : Day : ( ) page : ( )

Item

K-Itemset اگر Itemset ها K عضو داشته باشند

فراوانی (تکرار) : نرخ تکرار یک Item در مجموعه داده است.

Support → Absolute Support تعداد رخدادهایی که A و B همزمان با هم رخ میدهند.

Support → Relative Support احتمال اینکه A و B با هم رخ میدهد

confidence با Relative فرق دارد چون confidence همیشه اولی حتماً رخ داده و ...

Relative ← احتمال = تعداد رخدادی A و B / تعداد کل رخدادها

مثال : فرض کنید یک فروشگاه چنین تراکنش دارد.

TID	
10	Coke, nuTs, diaper
20	Coke, Coffe, diaper
30	Coke, diaper, egg
40	Nut, Egg, milk
50	Nut, coffee, diaper, egg, milk

min_Support=2 قرارمیدهیم.
پس باید بیشتر از ۲ بار آمده باشد
تا ما بنویسیمش.
الان میخواهیم 1-Itemset هایی را
پیدا کنیم که بیشتر از ۲ باشند:

1-Itemset = {Coke} {nuts} {diaper} {coffee}
{egg} {milk}

پس این روش خیلی کارایی دارد.
اگر minSupport = 4 باشد.
هیچ چیزی برای 1-Itemset پس خیلی بهینه است.

2-Itemset = {Coke, diaper}
{nut, egg} {egg, milk}

3-Itemset = {Nut, egg, milk}

حالا 4-Itemset چیزی پیدا نمی‌کنیم

پس قانون سختگیرانه می‌شه، چون تراکنشی هر ۴ تا را باهم خریده باشند پیدا نمی‌شود

اما اگر min support = 1 باشد این ۴ تایی هم پیدا می‌شود.

پس min support را جوری در نظر می‌گیرند که این مجموعه‌ای که بدست می‌آید متناسب باشد.

support confidence

Coke ⇒ diaper $(\frac{3}{5} = 60\%,\ \frac{3}{3} = 100\%)$

diaper ⇒ Coke $(\frac{3}{5} = 60\%,\ \frac{3}{4} = 75\%)$ اینجا diaper خریده و بعد Coke می‌خرد

تعیین ضرایب Confidence و Support.

اما محیط داده کاوی خیلی بزرگ است و این روش‌ها کم میارند :(

اگر یک مجموعه پرتکرار باشد، آیا زیر مجموعه‌ی آن هم پرتکرار است؟ بله

فرض کنید یک مجموعه ۱۰۰ تایی پیدا کردیم که پرتکرار است.

تعداد زیرمجموعه‌های این مجموعه: (زیرمجموعه تک عضوی، زیرمجموعه ۲ عضوی، ...)

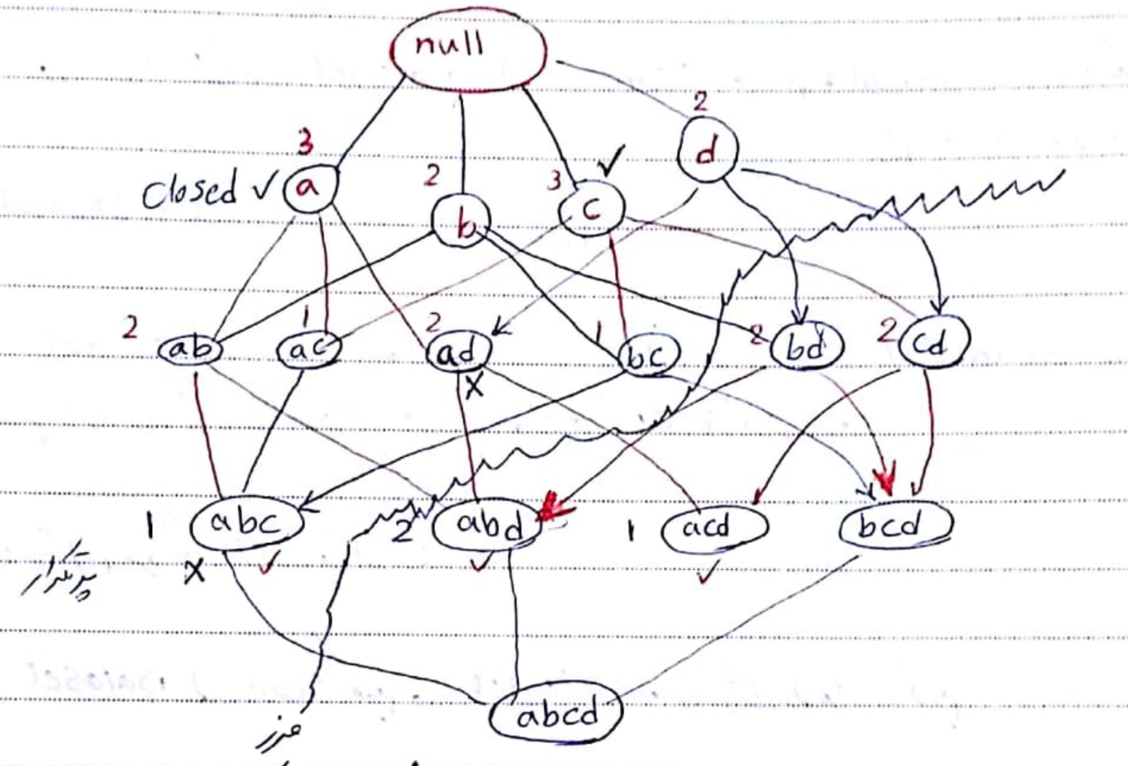
$$\binom{100}{1} + \binom{100}{2} + \binom{100}{3} + \cdots + \binom{100}{100} = 10^{30}$$

پس 10^{30} مجموعه پرتکرار پیدا کردیم.

اما نیازی نیست همه این‌ها را ذخیره کنیم فقط همان مجموعه ۱۰۰ تایی را ذخیره می‌کنیم.

Closed Frequent Itemset (بسته): یک Itemset مانند x را Closed می‌گویند اگر هیچ Super Itemset ای مانند y وجود نداشته باشد که support آن مشابه x باشد.
(Super Itemset: آیتم ست بالاتر از خودش - فراتر از خودش)

مثلاً در مثال قبل از مجموعه 3-Itemset ای چیزی بیشتر پیدا نکردیم پس این مجموعه Closed شد.



در این گراف یک مرز بین پرتکرارها و کم تکرارها می‌کشد: کشیدن آن خیلی راحت نیست نسبی می‌کشند

(ما) این مرز ممکن است دقیق نباشد.

آیتم‌هایی که روی مرز هستند support هایشان را بررسی می‌کنیم.

اگر آیتمست همه زیرمجموعه هایش در قسمت پایین مرز باشد به آن ماکسیمال فری کوئنت دیتاست می‌گویند مثل ad و abc

۳ تا الگوریتم برای پیدا کردن الگوهای پرتکرار:

Apriori - FP Growth - ECLAT

Apriori براساس دانش قبلی ما یک الگوی پرتکرار استخراج می‌کند.

اگر مجموعه‌ای پرتکرار باشد ← زیرمجموعه های آن پرتکرار است.

این الگوریتم از این قانون استفاده می‌کند.

اگر یک K-Itemset پرتکرار داشته باشیم از روی آن می‌توانیم یک (K+1) Itemset پرتکرار پیدا کنیم.

Subject:
Year. Month. Date. ( )

مراحل الگوریتم:

۱- در ابتدا مجموعه Dataset را Scan می‌کنیم: اول باید دیتاست را شناسایی کنیم بینیم کجا این اتفاق افتاده

۲- 1-Itemset های پرتکرار را استخراج می‌کنیم: L_1

۳- 2-Itemset ها را با استفاده از join داخلی 1-Itemset ها بدست آورده، تعداد پرتکرارهای آن را استخراج می‌کنیم و اسم آن را C_2 می‌زاریم.

حال مثال فروشگاه را در نظر بگیرید:

۱- Dataset را Scan کنیم و 1-Itemset های پرتکرار را بدست آوریم:

$L_1 = \{\{cok\}\ \{nut\}\ \{diaper\}\ \{coffee\}\ \{egg\}\ \{milk\}\}$

$L_2 = \{\{cok, nut\}\ \{cok, diaper\}\ \{cok, coffee\}\ \{cok, egg\}\ \{cok, milk\}\ \{nut, diaper\}\ \{nut, coffee\}\ \{nut, egg\}\ \{nut, milk\}\}$ ← ۲

حال هرکدام که تعداد تکرار آن از minsupport کمتر باشد حذف می‌کنیم. minsupport = 2

برای پیدا کردن 3-Itemset ها L_2 را با خودش join می‌کنیم (همیشه از مرحله قبلی) پرتکرارهای L_2 را با خودش join می‌کنیم ← اگر کاندیدی ننوشتیم فقط سه تایی‌ها را می‌نویسیم

$L_3 = \{cok, nut, diaper\}$

این مثال کامل نیست

این روند را تا جایی ادامه می‌دهیم که دیگر مجموعه پرتکرار پیدا نکنیم و می‌گوییم Closed شده.

Subject:
Year. Month. Date. ( )

مثال: مجموعه های پرتکرار را بدست آورید با استفاده از الگوریتم.

minsupport = 2

TID	
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

$L_1 = \{\{A\}^2 \{B\}^3 \{C\}^3 \{D\}^1 \{E\}^3\}$

هرس می کنیم Pruning

چون minsupport برابر 2 است

$L_2 = \{\{A,B\}^1 \{A,C\}^2 \{A,E\}^1 \{B,C\}^2 \{B,E\}^3 \{C,E\}^2\}$

$L_3 = \{\{A,B,C\}^1 \{A,C,E\}^1 \{B,C,E\}^2\}$

پس مجموعه {B, C, E} پرتکرار است ← زیر مجموعه های آن هم پرتکرار است.

مجموعه D حذف شده در مراحل بعدی D را نداریم.

پس اگر یک مجموعه پرتکرار نبود در مراحل بعدی نیز پرتکرار نیست.

مثال: * در صفحه کلید کاربرد دارد. minsupport = 2

TID	
T100	I_1, I_2, I_5
T200	I_2, I_4
T300	I_2, I_3
T400	I_1, I_2, I_4
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3
T800	I_1, I_2, I_3, I_5
T900	I_1, I_2, I_3

Subject:
Year. Month. Date. ( )

عیب این روش این است که هربار باید کل Dataset را Scan کنیم.
یا مثلاً یک سری مجموعه های اضافه داریم که باید آن ها را هرس کند پس بیا چیزایی
تولید کنه که به درد بخور باشند و همه را تولید نکند.
در دیتابیس های بزرگ خیلی بدتر می‌شود ← یعنی زمان بر - زیاد اینجا مثلاً دیتابیس کوچک بود.

اصلی‌ترین چالش های محاسباتی:
۱- مرور چندباره پایگاه داده تراکنش ۲- تعداد زیاد کاندیداهای تولید شده
۳- محاسبه support برای کاندیداها زمان‌بر است.
به دنبال راه حلی هستیم که این چالش ها را برطرف کنیم
راه حل که: ← ۱- کاهش مرور (اسکن) پایگاه داده
۲ کاندیداهای مفید تولید شوند
۳ محاسبه سریع support ها

۵ روش بهبود Apriori: ۱ درهم سازی ۲ کاهش تراکنش ها ۳ پارتیشن بندی
۴ نمونه گیری ۵ شمارش پویا

درهم سازی: روی یک مثال یاد می‌گیریم.
ایده اصلی: ۱ هم زمان با اینکه 1-Itemset ها تولید می‌شوند 2-Itemset ها هم تولید بشوند
۲ تابع درهم سازی تعریف می‌کنیم و عناصر 2-Itemset را به مقدار تابع hash
نسبت می دهد
۳ به هر مجموعه یک bucket گفته می‌شود.
۴ اگر شمارش اعضای bucket از min-support بیشتر باشد
به عنوان الگوی پرتکرار یا frequent شناخته می‌شود.

Subject:
Year. Month. Date. ( )

مثال: این مثال در صفحه قبل نوشته شده.

TID Items

I_1, I_2, I_3

$hash = (order\ of\ x) \times 10 + (order\ of\ y)\ \%\ 7$

باقی مانده تقسیم بر ۷ می‌تواند اعداد ۰ تا ۶ باشد. پس این حالت‌ها را می‌توانیم داشته باشیم

مثلاً $(I_1, I_4) \Rightarrow 1 \times 10 + 4 = 14\ \%\ 7 = 0$

bucket Address	0	1	2 ✓	3	4	5 ✓	6 ✓ Frequent
bucket count (تعداد باکت‌ها)	2	2	4	2	1	4	4
bucket content (محتوای باکت)	$\{I_1, I_4\}$ $\{I_3, I_5\}$	$\{I_1, I_5\}$ $\{I_1, I_5\}$	$\{I_2, I_3\}$ $\{I_2, I_3\}$ $\{I_2, I_3\}$ $\{I_2, I_3\}$	$\{I_2, I_4\}$ $\{I_2, I_4\}$	$\{I_2, I_5\}$	$\{I_1, I_2\}$ $\{I_1, I_2\}$ $\{I_1, I_2\}$ $\{I_1, I_2\}$	$\{I_1, I_3\}$ $\{I_1, I_3\}$ $\{I_1, I_3\}$ $\{I_1, I_3\}$

اگر $minSupport = 3$ ← مجموعه‌هایی که bucket count آن‌ها بیشتر یا برابر 3 باشد Frequent است.

تابع hash یک سری تداخل‌ها دارد. دو تا چیز ممکن است یک Frequent بخورند.
این روش حذف دارد. می‌توانیم تابع hash را جوری انتخاب کنیم که تا حد ممکن این تداخل برطرف شود.

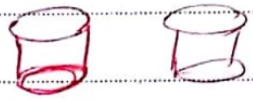
تداخل ← مثلاً ۴ تا مجموعه داریم اما این ۴ تا مجموعه همجنس نیستند مثلاً دو تا $\{I_1, I_2\}$ و دو تا $\{I_2, I_5\}$ با فرض اینجا غلط دو تا Frequent را جدا بنویسیم.

کاهش تراکنش: اگر تراکنش در مجموعه k-Itemset الگوی پرتکرار ندارد پس در مجموعه k+1 Itemset هم حتماً الگوی پرتکرار ندارد بنابراین آن تراکنش را حذف می‌کنیم.

مثلاً $\{I_2, I_4\}$ پرتکرار است ← مربوط به کدام تراکنش است؟ تراکنش 200
من تراکنش 200 را حذف می‌کنم و دفعات عدی آن را بررسی می‌کنیم.
اما تراکنش 400 را حذف نمی‌کنم! چون I_1 را هم دارد.
تراکنش‌هایی را حذف می‌کنیم که فقط I_2, I_4 را دارند.

Subject:
Year. Month. Date. ( )

پارتیشن بندی: فرض کنید یک دیتابیس داریم که 10G است اما فضایی که داریم 1G است. اگر قرار باشد هر دفعه 1G از دیتابیس را بیاریم تو حافظه، 1-Itemset پیدا کنیم؛ به این شکل باید 10 بار از دیسک اطلاعات را بیاریم توی حافظه اصلی.
مثلاً فرض کنید minSupport ما 20 باشد.



دیتابیس را 10 قسمت کردیم. اگر هر مجموعه را در هر قسمت 2 بار تکرار شده باشد در نهایت در 10 قسمت 20 بار تکرار می‌شود و پرتکرار است.
اما ممکن است توزیع داده‌ها نرمال نباشد مثلاً یک جا 10 بار تکرار شده و یک جا صفر بار.
پس نمی‌توان نتیجه گرفت اگر در یک قسمت پرتکرار باشد حتماً در نهایت هم پرتکرار است.

local ← در یک پارتیشن ، global ← در کل

اگر یک Item در کل پرتکرار باشد در یک قسمت یا local هم پرتکرار می‌شود.
این قسمت مثل قسمت‌های قبلی نیست که Data Base کار داشته باشد بلکه با انتقال از دیسک به حافظه کار داریم.

local	global
frequent در یک local	⟸ frequent
frequent	⟹

پارتیشن بندی در دو فاز انجام می‌شود:
در فاز اول: داده‌ها به چند بخش تقسیم می‌شوند (به طوری که هر بخش بتواند در حافظه اصلی قرار گیرد)

sup > min support نیست ~ ~ sup > min support

تعریف min support نسبی: یا درصد پشتیبان نسبی = min تکرار برای هر پارتیشن برابر است با min support × تعداد تراکنش‌های یک بخش

مثلاً minsupport = 20 است و دیتابیس به 10 قسمت تقسیم می‌شود به طور متوسط در هر قسمت 2 بار تکرار باید

min support نسبی

چند نتیجه می‌گیریم ← ۱. الگوهای پرتکرار هر بخش (الگوهای محلی) ممکن است نسبت به کل پایگاه داده پرتکرار باشند یا نباشند

۲. هر Itemset که در پایگاه پرتکرار باشد باید حداقل در یکی از پارتیشن های پایگاه داده پرتکرار باشد.

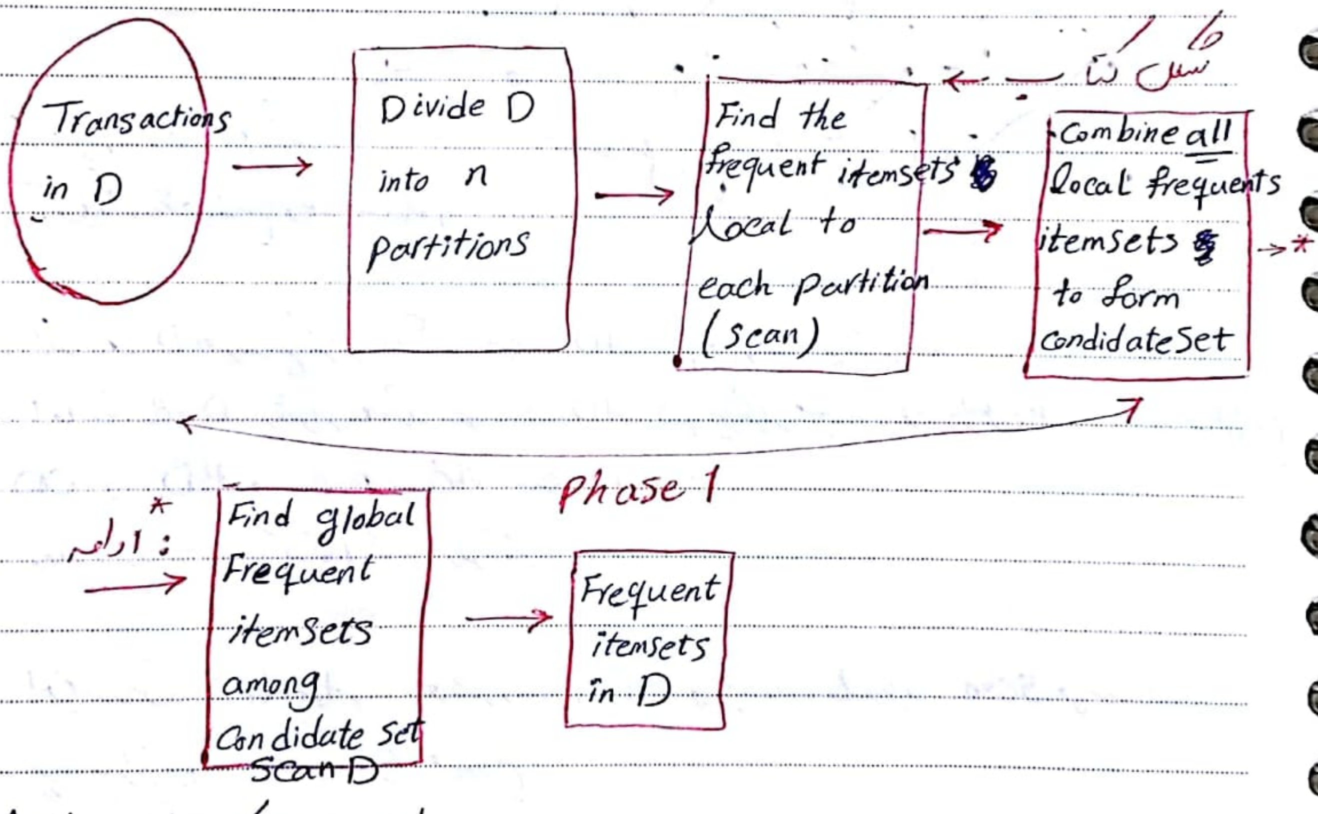
۳. نتیجه: هر itemset پرتکرار در یک بخش می‌تواند در مجموعه کاندید کل پایگاه داده باشد.
(یعنی هنوز کاندید است و اینکه پرتکرار باشد یا نه در فاز دوم مشخص می‌شود)

فاز دوم ← مرور دوم پایگاه داده

برای هر itemset کاندید باید support واقعی را محاسبه می‌کنیم

در واقع این روش هم داره پایگاه داده را مرور می‌کنه چندبار.

شکل کتاب ←



support کل هر itemset را با min support کل مقایسه کنیم

Subject:
Year. Month. Date. ( )

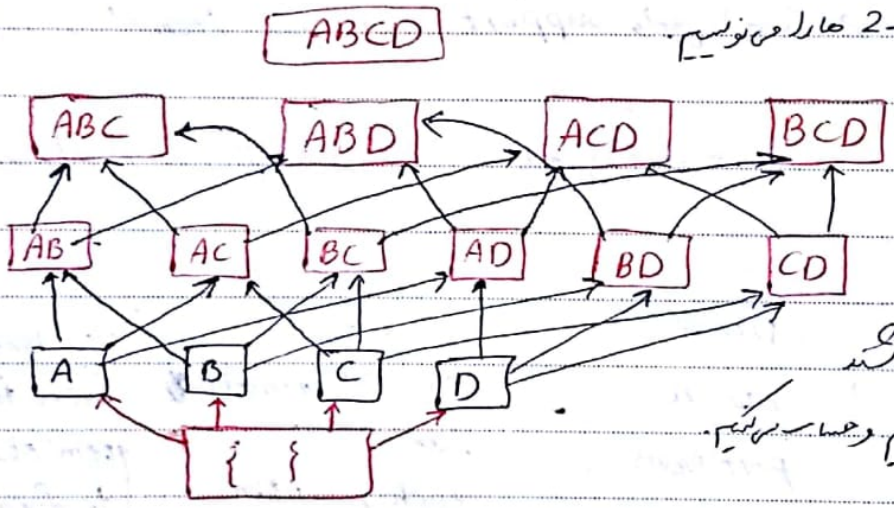
نمونه برداری :

مشکل : در حالتی نمونه برداری می‌کنیم که نمونه‌های حاضلی به هم نزدیک باشند اما اگر نمونه‌ها از هم فاصله داشته باشند این کار قابل قبول نیست.

شمارش پویا : هدف این کار برای کاهش تعداد اسکن هاست.

مثال : اول کار null است، هیچی نداریم. بعد 1-Itemset ها را می‌نویسیم.

مرحله بعد 2-Itemset ها را می‌نویسیم.



اگر زیری‌ها پرتکرار باشند آن را در نظر می‌گیریم و حساب می‌کنیم.

اگر A و D هر دو پرتکرار باشند ⇐ AD هم می‌تواند پرتکرار باشد.

اما اگر A و D پرتکرار نباشند ⇐ AD هم پرتکرار نیست و در مراحل بعد آن را در نظر نمی‌گیریم.

BCD ⇐ ~~AC~~ ، BC ، BD و CD

زمانی پرتکرار است که تمام مجموعه زیر آن پرتکرار باشد.

این روش Apriori و همبودهای آن پایگاه داده را زیاد Scan می‌کنند

پس سراغ روش های دوم می‌رویم.

Subject:
Year. Month. Date. ( )

روش FP-Growth روشی برای ساخت الگوی پرتکرار از طریق گسترش آن
Frequent pattern Growth Approach

این الگوریتم به صورت افزایشی رشد می‌کند Growth
هدف آن یافتن Itemset های پرتکرار بدون تولید مجموعه کاندید
انگیزه ایجاد آن به دلیل معایب روش Apriori بوده است.

معایب ← ۱- جستجوی سطح به سطح: تراکنش‌ها را یکبار بررسی می‌کند و پس از آن مجموعه کاندید تولید می‌کند و سپس از بین مجموعه کاندید، پرتکرار بودن را مشخص می‌کند.
۲- تعداد کاندیدها زیاد بود.

FP-Growth ← ۱- جستجوی عمق
۲- مجموعه کاندید را سریع تولید می‌کند

فازهای روش FP-Growth ← ۱- درخت FP را تولید کند.
۲- Conditional fp-tree را مشخص کند

نحوه ساخت درخت FP ←
۱- تمام 1-Itemset ها را پیدا کرده و Support آن‌ها را محاسبه کند
۲- " " براساس مقدار Support مرتب کند

مثال * در چند صفحه قبل را در نظر بگیرید:
اول 1-Itemset ها را تعیین کنید.

1-Itemset	Support
I_1	6
I_2	7
I_3	6
I_4	2
I_5	2

مرحله ①

مرحله ② مرتب کردن →

I_2	7
I_1	6
I_3	6
I_4	2
I_5	2

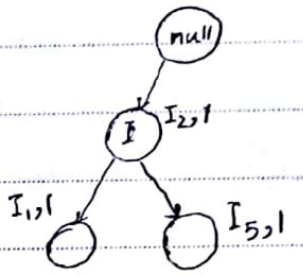
Subject:
Year. Month. Date. ( )

ساخت درخت: تراکنش ها را مرتب می‌کنیم.

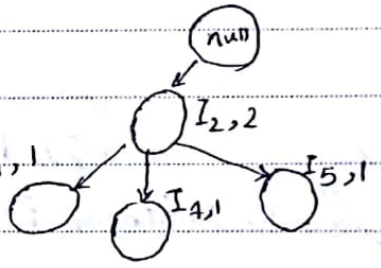
TID	Items	مرتب
100	I_1, I_2, I_5	I_2, I_1, I_5
200	I_2, I_4	I_2, I_4
300	I_2, I_3	I_2, I_3
400	I_1, I_2, I_4	I_2, I_1, I_4
500	I_1, I_3	I_1, I_3
600	I_2, I_3	I_2, I_3
700	I_1, I_3	I_1, I_3
800	I_1, I_2, I_3, I_5	I_2, I_1, I_3, I_5
900	I_1, I_2, I_3	I_2, I_1, I_3

ساخت درخت:

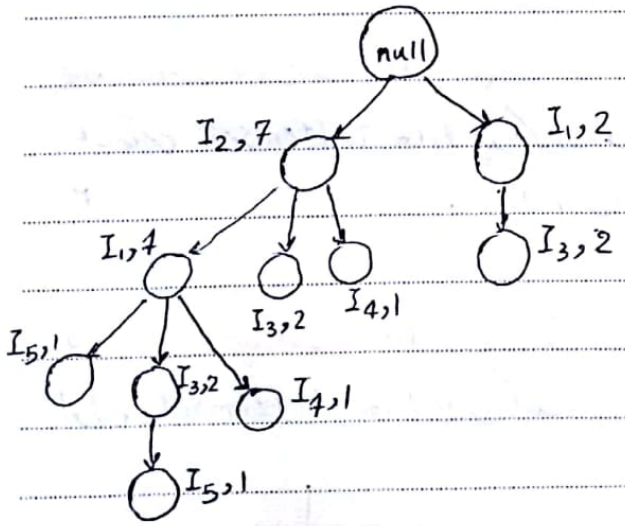
مرحله اول



مرحله دوم



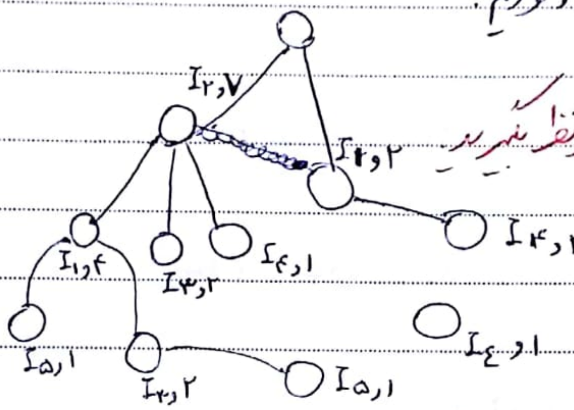
همین ترتیب بقیه تراکنش ها را به درخت اضافه می‌کنیم.



عمده مشکل الگوریتم Apriori این است که سطح به سطح کار می‌کند

الگوریتم FP_Growth به جای سطحی کار کردن، عمقی کار می‌کند و از روی درخت فراوانی را بدست می‌آوریم.

I_2	7
I_1	6
I_3	6
I_4	2
I_5	2



مثال صفحه قبل را در نظر بگیرید

این شکل ادامه داره که همچینشو نفهمیدم :)

$(I_2\ I_1\ I_5 : 1)$ $(I_2\ I_1\ I_3\ I_5 : 1)$ از برگ های درخت شروع به حذف می‌کنیم

I_5 را حذف کرده و به عنوان پیوند الگو قرار می‌دهیم بنابراین دو مسیر پیشوندی زیر حاصل می‌شود

① $(I_2, I_1 : 1)$

② $(I_2\ I_1\ I_3 : 1)$

$\Rightarrow$ Conditional pattern Base پایه‌های الگوهای شرطی

الگوریتم Eclat:

کاوش الگوی پرتکرار براساس چیدمان داده‌ها به صورت عمودی

در الگوریتم Eclat قالب داده‌ها را عوض می‌کنیم (افقی ← عمودی)

در حالت عادی برای هر تراکنش مجموع کالاها را به صورت سطری در مقابل آن می‌نویسیم

به این چینش اطلاعات قالب افقی یا horizantal می‌گوییم.

در مقابل آن قالب عمودی یا Vertical به صورت زیر است:

تمام آیتم‌ها را لیست کن و بگو در چه تراکنشی قرار دارد؟

I_1 {T100, T400, T500, T700, T800, T900}

I_2 {T100, T200, T300, T400, T600, T800, T900}

I_3 {T300, T500, T600, T700, T800, T900}

Subject:
Year. Month. Date. ( )

I_4 {T200, T400}

I_5 {T100, T800}

: 2- ItemSet

{I_1, I_2}	{T100, T400, T800, T900}
{I_1, I_4}	{T500, T700, T800, T900}
{I_1, I_4}	{T400} ✗ حذف می‌کنیم
{I_1, I_5}	{T100, T800}
{I_2, I_3}	{T300, T400, T800, T900}
{I_2, I_4}	{T200, T400}
{I_2, I_5}	{T100, T800}
{I_4, I_5}	{T800} ✗ حذف می‌کنیم

پرتکرارها راحت‌تر پیدا می‌کند 1- ItemSet

مجموعه اختلاف را بدست می‌آوریم
میزان ذخیره سازی را کاهش می‌دهد
جایی که نوعدن را می‌نویسیم.

: 3- Itemset

{I_1, I_2, I_3}	{T800, T900}
{I_1, I_2, I_5}	{T100, T800}

آیا تمام الگوی پرتکرار مورد علاقه ماست؟

Subject:
Year. Month. Date. ( )

۱۰,۰۰۰ تراکنش خرید ۶۰۰۰ خرید بازی های کامپیوتری
۴۰۰۰ خرید بازی کامپیوتری و فیلم ویدیویی
۷۵۰۰ خرید فیلم های ویدیویی

min support = 30 Confidence = 60

دو پارامتر به صورت منفی با هم وصل شده اند.

خرید (X تراکنش و بازی کامپیوتری X) → خرید (فیلم ویدیویی X)

Support %40 Confidence = %66

قانون Strong (قوی) را پیدا میکنیم و به مدیر فروشگاه می دهیم.
قانون گمراه کننده است زیرا اگر یک نفر ویدیویی بخرد ۷۵٪ احتمال دارد

ویدیویی → کامپیوتری ٪۶۶

احتمال خرید کم شده

پس هر قانون قوی منجر نمی شود به این که یک قانون جذاب پیدا کردیم.
برای پیدا کردن قوانین جذاب بایستی علاوه بر Support و Confidence معیار دیگری را هم در نظر گرفت که شامل همبستگی بین دو متغیر می باشد.
از چند روش می توانیم همبستگی را به دست آوریم:

۱) Lift بالابری ۲) توزیع χ^2 (کای دو) ۳) all Confidence
۴) ماکسیمم - اطمینان ۵) کولزمنسکی ۶) کسینوسی

مثال: ۱۰۰۰ دانشجو ۶۰۰ دانش آموز (شنا) S
۷۰۰ دانش آموز (دوچرخه سواری) B ۴۲۰ دانش آموز (شنا و دوچرخه سواری)
$S \cap B$

$P(S \cap B) = \frac{420}{1000} = 0.42$ $P(S) = \frac{600}{1000} = 0.6$

$P(S) \times P(B) = 0.6 \times 0.7 = 0.42$ $P(B) = 0.7$

Subject:
Year. Month. Date. ( )

$P(S \cap B) = P(S) \times P(B)$ مستقل

$P(S \cap B) > P(S) \times P(B)$ ارتباط مثبت

$P(S \cap B) < P(S) \times P(B)$ ارتباط منفی

$$lift(A,B) = \frac{P(A \cup B)}{P(A) \times P(B)}$$

$lift = 1 \Rightarrow$ استقلال مستقل

$lift < 1 \Rightarrow$ اعداد A به صورت منفی با رخداد B در ارتباط است

$lift > 1 \Rightarrow$ A و B به صورت مثبت با یکدیگر هم بسته هستند

پیدا کردن مدل‌هایی که داده‌ها را پیدا کند و روی آن‌ها عملیات انجام دهیم:

۱- دسته‌بندی داده‌ها Classification

۲- پیش‌بینی predict

مثلاً در پزشکی ۳ دسته داریم A, B, C باید طبقه‌بندی به گونه‌ای باشد که وقتی فرد دیگری مبتلا شد بدانیم در کدام دسته قرار می‌گیرد.

مثلاً در بانک از روی داده‌ها باید بدانیم که آیا باید به آن فرد وام بدهیم یا نه؟ risk/safe

تیم درمان، وام، دیتای تجارت مثال‌های مهم هستند.

مشتری جدید با این ویژگی‌ها (خصوصیات) چقدر احتمال دارد که کامپیوتر بخرد یا نه؟ yes/No

Classification ـ ۱

دو تا فاز داریم: ۱- فاز یادگیری / آموزش / training / learning

۲- فاز طبقه‌بندی

Subject:
Year. Month. Date. ( )

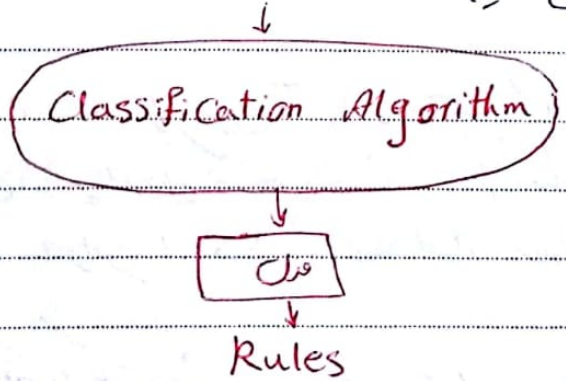
باید مدلی درست کنیم که داده‌ها را طبقه‌بندی کند classifier

باید به فاز اول بگیم که کار طبقه‌بندی را انجام دهد بعد در فاز دوم تست می‌کنیم که درست کار می‌کند یا نه؟
و در کدام دسته قرار می‌گیرد؟

فاز اول: Training Data Set

name	Age	income	loan decision
Sandy	youth	low	risky
Bill	"	low	risky
Cavolina	middle Aged	high	safe
Rick	"	low	risky
Susan	se	low	safe
Clarie	"	Med	safe
Joe	middle	high	safe
Ali	youth	Med	risky
Reza	"	low	safe

مجموعه داده‌ها را به یک الگوریتم می‌دهیم، این الگوریتم یک مدل درمی‌آورد در این مدل یک سری قوانین استخراج می‌کنیم



Subject:
Year. Month. Date. ( )

قوانین: if age = youth then loan_decision = risky
if incom = high " " = safe
if age = middle-aged ~~then~~ and incom = low
the loan decision = risky

این قوانین باید در مجموعه دیتاست مثال نقض داشته باشند.
الان رضا یک قانون را نقض می‌کند. یعنی قانون ریسک نیست در مورد او
پس این قانون ها با یک درجه‌ی احتمال ریسک بودن را دارند. ← دقت

دو نوع یادگیری داریم: ۱- با نظارت Superwised learning
۲- بی نظارت non " "

۱- آن فیلدی که نظارت می‌کند در مجموعه باشد یادگیری با نظارت می‌شود
دیتا را به چندین دسته تبدیل می‌کنیم.
۲- فیلدی که نظارت می‌کند را نداریم باید base ویژگی های کلاس آن را تعیین کنیم
براساس ویژگی ای کلاس دسته‌بندی می‌کنیم هر آدمی که جدید می‌آید شبیه به کدام دسته است
و در کدام دسته قرار می‌گیرد. بحث خوشه‌بندی

یادگیری با نظارت: داده‌ای آموزشی که شامل مشاهدات، اندازه‌گیری و... همراه با
برچسب هستند که کلاس آن مشاهدات را نشان می‌دهد داده‌ی جدید براساس
مجموعه آموزشی دسته‌بندی می‌شود.

یادگیری بدون نظارت: برای تابلی که داده برچسب کلاس ناشناخته است اغلب
تعداد و نوع کلاس‌ها نیز از قبل مشخص نیست. طور مثال وضعیت وام گیرنده
مشخص نباشد بنابراین از لحاظ شباهت افراد می‌توان ریسک داشتن وام را
بررسی کرد.

هر سطر tuple گفته می‌شود instance / sample
ممکن است چند تای قوانین یک درجه با هم برابر شد.

Subject:
Year. Month. Date. ( )

اینکه مثلاً به ازای هر تاپل یک قانون استخراج کنیم اصلاً خوب نیست

فاز اول: ارتباط بین داده‌ی و کلاس را گاهی اوقات با یکسری قانون، درخت تصمیم و یا یک فرمول ریاضی می‌توان نشان داد.

این قاعده‌ها به چه منظوری مورد استفاده قرار می‌گیرد؟

۱- قاعده‌ها و قوانین پیدا شوند و برای دسته‌بندی داده‌ی جدید می‌توانند به کار آید

۲- دید واضح‌تری نسبت به داده‌ی موجود می‌دهند

۳- نمایش داده‌ی را فشرده‌سازی می‌کنند.

بعد از ایجاد classifier بایستی یک مجموعه داده جدید به نام test انتخاب کرده و جهت test و ارزیابی به مدل بدهیم.

دقت: accuracy از مقایسه نتایج مدل و نتایج واقعی مجموعه داده test بدست می‌آید.

باید داده‌ها ۷۰٪ train و ۳۰٪ test

درصدی از مجموعه test که classifier درست طبقه‌بندی می‌کند

بنابراین زمانی که صحت classifier تأیید شد می‌توانند برای تشخیص برچسب داده‌های جدید بدون lable استفاده شود.

چگونه مدل را بسازیم؟

درخت تصمیم‌گیری Decision Tree

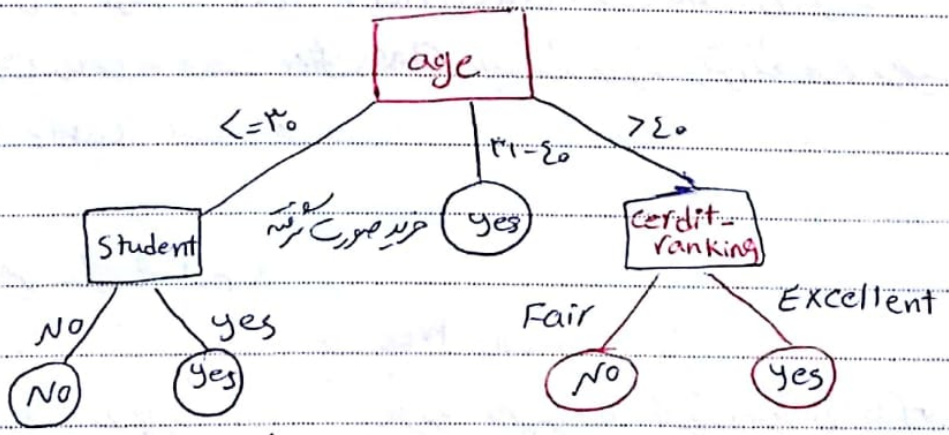
ساختار فلوچارتی دارد هر گره به جز برگ‌ها کار یک شرط را انجام می‌دهد

به عبارتی بر روی یک از ویژگی‌ها test را انجام می‌دهد.

هر شاخه یا یال پس از شرط، نتیجه شرط می‌باشد و هر برگ برچسب یا lable کلاس است.

Subject:
Year. Month. Date. ( )

age	income	Student	cerdit_ranking	buy_com
<=۳۰	hi	no	Fair	no
<=۳۰	hi	no	Excellent	no
۳۱_۴۰	hi	no	F	yes
>۴۰	med	no	F	yes
>۴۰	low	yes	F	yes
>۴۰	low	yes	Ex	No
۳۱_۴۰	low	yes	Ex	yes
<=۳۰	med	no	F	No
<=۳۰	low	yes	F	Yes
>۴۰	med	yes	F	yes
<=۳۰	med	yes	Ex	No
۳۱-۴۰	med	no	Ex	yes
۳۱-۴۰	hi	yes	F	yes
>۴۰	med	no	Ex	No



برای اینکه ببینیم فرد جدید کامپیوتر می‌خرد یا نه باید از ریشه شروع کنیم و ببینیم در کدام دسته قرار می‌گیرد.
یک مسیر از ریشه تا برگ باید پیدا کنیم.

Subject:
Year. Month. Date. ( )

آنرا یک جایی پیدا کنیم که همه افراد جزء یک دسته، کلاس باشند در آن جا توقف داریم

مزایای درخت تصمیم: ۱- نیازمند دانش و تخصص زیاد نیست ۲- می تواند داده با ابعاد زیاد را پوشش دهد ۳- درک آن آسان است ۴- آموزش دسته بندی ساده است
۵- دقت خوب ۶- کاربردی

مشکلات: ۱- تعیین ویژگی ها برای هر سطح از درخت
۲- زمانی که درخت ساخته شد، بعضی از شاخه ها حاوی نویز داده خارجی باشند.
۳- چگونه این شاخه ها را شناسایی و هرس کنیم.

الگوریتم درخت تصمیم: یک رویش حریصانه بدون بازگشت از عقب به صورت بالا به پایین و تقسیم و حل است.

نحوه ساخت: در آغاز همه رشته و نمونه های آموزشی را در ریشه قرار می دهد و در مرحله ۲ یک ویژگی را انتخاب می کنیم و بر اساس آن تست را انجام می دهیم اگر ویژگی چند مقدار داشته باشد، چند شاخه داریم.
مرحله ۳: داده ها بر اساس ویژگی داده شده پارتیشن بندی می شوند (کل داده آموزشی به شاخه ها انتقال می یابند)

مرحله ۲ و ۳ را به صورت بازگشتی انجام می دهیم تا به شرط توقف برسیم.

شرایط توقف: ۱- همه نمونه ها برای یک گره داده شده متعلق به یک کلاس هستند
۲- هیچ ویژگی ای برای پارتیشن بندی بیشتر باقی نمانده باشد.
۳- هیچ نمونه ی دیگری نداشته باشیم.

مهمترین مسئله در درخت تصمیم این است که ویژگی ها را چگونه انتخاب کنیم.
در هر سطح چه ویژگی هایی را انتخاب کنیم.
برای اینکه کدام ویژگی را انتخاب کنیم ۳ تا الگوریتم داریم.
برای انتخاب بهترین ویژگی در هر سطح: ID3 - C45 - CART

Subject:
Year. Month. Date. ( )

ID3 : ۱ فرض کنید بهترین ویژگی تقسیم برای گره بعدی A باشد
۲ A را به عنوان ویژگی تقسیم برای گره قرار بده
۳ برای هر مقدار از A یک فرزند جدید ایجاد کن
۴ مرتب سازی نمونه‌های آموزشی برای هر گره با توجه به مقدار ویژگی ساخته
۵ اگر همه‌ی نمونه‌های آموزشی کاملاً طبقه‌بندی شدند (همان مقدار ویژگی هدف) متوقف شو، در غیر این صورت برای گره‌ی جدید تکرار کن

در هر گره‌ی تقسیم یک معیار داریم که مشخص می‌کند در هر سطح کدام ویژگی انتخاب شود.
مثلاً در ID3 معیار Informationgain
C45 Gain Ratio
CART Gini

حالا می‌خواهیم برای دیتا بیس بالا درخت تصمیم را بسازیم:
Age:

$$Information\ gain_{(A)} = Info(D) - Info_A(D)$$ → A روی فرم Info D

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

حل Info: باید ببینیم چند تا کلاس داریم. اینجا دو تا کلاس داریم NO, yes
$P(yes) = \frac{9}{14}$ $P(NO) = \frac{6}{14}$ احتمال هرکدام چقدر هست؟
به تعداد کلاس‌هایی که داریم این فرمول را داریم.

$$Info(D) = -\frac{9}{14}\log_2 \frac{9}{14} - \frac{6}{14}\log_2 \frac{6}{14}$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|Dj|}{|D|} \times info(Dj)$$ A = Age

حالا دسته Age داریم؟ ۳ تا پس ۳ بار فرمول را بنویسم
برای هرکدام yes ها و NO ها را حساب می‌کنیم.

$$Info_A(D) = \frac{5}{14}\left(-\frac{2}{5}\log_2 2/5 - \frac{3}{5}\log_2 3/5\right) + \frac{4}{14}\left(-\frac{4}{14}\log_2 4/4 - 0/4 \log_2 0/4\right)$$

$$+ \frac{5}{14}\left(-\frac{3}{5}\log_2 3/5 - \frac{2}{5}\log_2 2/5\right)$$

Subject:
Year. Month. Date. ( )

$$Info(A) = Info(D) - Info_A(D) = 0.264$$

$Info_{(income)} = 0.029$ $Info(Stu) = 0.151$ $Info(Cr) = 0.048$

$Info(Age)$ از همه بیشتر است پس اولین ویژگی در درخت تصمیم Age است.

مقدار اطلاعاتی که نیاز هست برای اینکه تقسیم بندی صورت بگیرد

عیب این روش: اگر این دیتا بین ID باشد، اول کار روی ID شاخه بندی می‌کند و تعداد شاخه خیلی زیاد می‌شود.

اگر مقادیر یک فیلد پیوسته باشد مثلاً Age پیوسته باشد و هر کس سن دقیق اش ثبت شده باشد.
در این صورت باید Split Point پیدا کنیم نقطه شکست

محاسبه Information gain برای مقادیر پیوسته:

باید نقطه Split point برای ویژگی پیوسته A تعیین کنیم بنابراین مقدار A را به صورت صعودی مرتب کرده به طور معمول می‌توان نقطه وسط بین هر جفت از مقادیر مجاور را به عنوان نقطه تقسیم در نظر بگیریم.

و بین نقطه‌ای با کمترین $\left(\frac{a_i + a_{i+1}}{2}\right)$
expected Infor... requirement را به عنوان نقطه‌ی تقسیم انتخاب می‌کنیم.

پیدا کردن این نقاط کار راحتی نیست. مثلاً هر بار باید بیاییم بین دو تا عدد یکی را به عنوان نقطه تقسیم در نظر بگیریم. بهترین کار این است که خودمان نسبت به داده‌ها شناخت داشته باشیم.

پس میریم سراغ معیار بعدی:

Split Info

$$A = income \Rightarrow -\frac{4}{14}\log\frac{4}{14} + \frac{6}{14}\log\frac{6}{14} - \frac{4}{14}\log\frac{4}{14} = 1.557$$

برای نرمالایز کردن $Gain\ Ratio = \frac{0.029}{1.557}$

Subject:
Year. Month. Date. ( )

هر چقدر Gain Ratio بیشتر باشد، این ویژگی را انتخاب می‌کنیم.

Gini: هر دفعه درخت را به دو قسمت تقسیم می‌کند و برای هر کدام Gini را بدست می‌آورد

اگر مجموعه A، V تا مقدار داشته باشد 2^V تا مقدار می‌توان از آن ساخت:

income = {low, med, high} {}

{low, med}^{D_1} {high}^{D_2} → این مجموعه برای تقسیم‌بندی بهتر است

{low, high} {med} → gini = 0.458

{med, high} {low} → gini = 0.450

با مجموعه‌ها باید برابری شوند
- {low}
- {med}
- {high}

چون می‌خواهیم به ۲ دسته تقسیم شود.

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

$$gini_A(D) = \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) = 0.443$$

D_1 برای D_2 برای

۰/۴۵۹ − ۰/۴۴۳ =

هر چقدر gini عدد بیشتر باشد آن تقسیم‌بندی بهتر است.

Subject:
Year. Month. Date. ( )

بهترین دسته‌بندی سن: {youth, senior} & {middle} → gini = 0.357

St → 0.367 C-R → 0.429

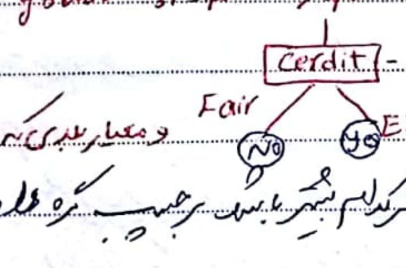
Age-gini از همه کمتر است پس Age را برای دسته بندی استفاده کنیم.
عیب این روش محاسبات زیاد است.
اما با وجود همه‌ی این‌ها درخت تصمیم‌گیری روش خوبی است.

مشکلات درخت تصمیم: زیاد شدن بیخود شاخه‌ها که باید هرس کنیم.
ممکن است برای مدل داده آموزشی درخت تصمیم درست کار کند اما برای مجموعه داده تست درست کار نکند یعنی درخت overfiting باشد که ناشی از ۲ مشکل است:
۱- تعداد شاخه‌های درخت تصمیم زیاد می‌باشد ۲- درخت تصمیم برای مجموعه داده تست، خوب عمل نمی‌کند

← راه حل: هرس کردن pruning
۱- prepruning درهنگام ساخت درخت اگر نیاز هست شروع به هرس کردن کنیم.
۲- postpruning بعد از اینکه درخت ساخته شد آن را هرس می‌کند که این روش هزینه‌بر است.

در هر گره تصمیم بگیریم که اگر تقسیم مناسب نیست ادامه ندهیم.
در واقع در صورتی که تشخیص داده شود، تقسیم بیشتر یک شاخه بهبودی در دقت نخواهد داشت آن شاخه هرس شده و دیگر تقسیم انجام نخواهد شد. با بستن آن گره را تبدیل به برگ نموده و کلاس آن شاخه را براساس بیشترین تعداد اعضای آن محاسبه کرد.

مثلاً:



اینجا مثلاً تقسیم بدتر می‌شود.
این را تبدیل به برگ می‌کنیم و معیار بعدی که تقسیم می‌کنیم income باشد.
بسته به اینکه تعداد yesها و Noها هر کدام بیشتر باشد برچسب گره‌ها را می‌زنیم.

Subject:
Year. Month. Date. ( )

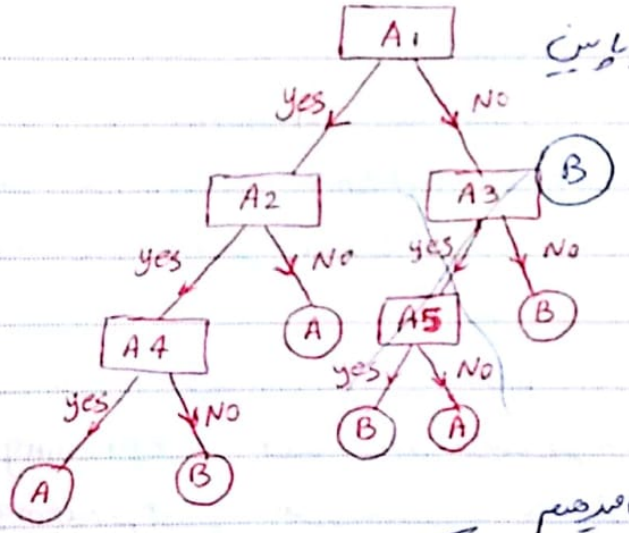
مثلاً دو تا قانون داریم:

Cerdit_rang = Fair NO

income = high yes

حالا یک مثال نقض پیدا می‌کنیم مثلاً Fair high NO پس از یک جایی به بعد تقسیم‌بندی به نفعمان نیست

معیار تقسیم‌گیری: انتخاب یک مقدار آستانه برای معیار information gain



هرس کردن عرا اساس از پایین درخت شروع می‌شود و بر اساس معیار هرس شدن یا نشدن یک ساخته تعیین می‌شود.

یک مجموعه داده تست به آن میدهیم ببینیم روی چند درصد از داده‌ها خطا پیش می‌کند. درصد خطا

مثلاً میزان خطا از ۱۰ درصد بیشتر است یا مثلاً ۹۰ درصد خطا دارد. این ساخته را هرس می‌کنیم.

به این مجموعه داده که برای تعیین درصد خطا استفاده می‌شود مجموعه pruning set می‌گویند.

مشکل بیشتر درخت تصمیم: ۱ Replication زیر درخت تکراری

یک زیر درختی توی درخت عیناً داره تکرار میشه.

۲ repetition ویژگی تکراری چندین بار روی یک ویژگی شرط داریم.

ویژگی‌های تکراری در عمق‌های مختلف

برای این مشکلات از هرس کردن می‌توان استفاده کرد.

Subject:
Year. Month. Date. ( )

برای حل این مشکلات از multi-attribute یا multi variant استفاده میکنیم.
به صورت ترکیبی استفاده میکند. شرط ها را روی هم دو قرار میدهد.

مشکل بعدی: اگر داده ها زیاد باشند در حافظه اصلی جا نمی شوند.
اصطلاحاً مقیاس پذیر نیست (Scalable)

راه حل ←

۱- Rain forest یک سری ست ها درست کن

Attribute value class label

سپس به جای اینکه کل اطلاعات را بیاریم توی حافظه اصلی یک سری ست میسازیم و آن ها را میاریم.

۲- BOAT

از معیار آماری Boot Strapping سریع تر است.

Arc-list on Age

Age	Buy_comp yes تعداد	No تعداد
<30	2	3
30< <40	4	0
>40	3	2

روش دسته بندی درخت:

روش بیزین: بین آن روی قانون های احتمال و قانون های بیز است
مجموعه داده قبلی را داریم. اگر یک تاپل داشته باشیم میتوانیم احتمال تعلق آن به یک کلاس خاص را تعیین کنیم.

ساده ترین بیزین ← Naiir Beysian classifier ← Independent

فرض اولیه: ویژگی ها از یکدیگر مستقل هستند و هیچ تاثیری بر روی یکدیگر ندارند
(مهم ترین مزیت این روش ← سرعت بالا (چون با یک مزول مشخص می کند)
و معمولاً دقت بالایی دارد.

$P(H|X)$ اگر X وجود دارد با خ داده احتمال پیدا کنیم کامپیوتر بخرد

X: تاپل evident

H: فرضیه (مثلاً کامپیوتر بخرد)

posterior probability

Subject:
Year. Month. Date. ( )

likelihood

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

prior probability ← $P(X)$, $P(H)$

مثال: دکتر می‌داند که سرطان در ۵۰ درصد مواقع باعث خشکی گردن می‌شود

H : فرضیه: خشکی گردن

X : جوانی ناشی از سرطان.

$$P(H|X) = \frac{\frac{50}{100} \times \frac{1}{50000}}{\frac{1}{20}}$$

تابع X یک ویژگی است.

مثلاً X باشد ← Age<30

حالا بگو احتمال اینکه کامپیوتری خرید باشد؟ income = low , Fair , ...

$$P(C|A_1 A_2 \dots An) = \frac{P(A_1 A_2 \dots An|C)\,P(C)}{P(A_1 A_2 \dots An)}$$

هرکدام جداگانه در P(C) ضرب می‌شود

$$= \frac{[P(A_1|C)\,P(A_2|C) \dots P(An|C)]\,P(C)}{P(A_1 A_2 \dots An)}$$

می‌خواهیم صورت را max کنیم

دو تا کلاس در نظر می‌گیریم ←

C_1 = yes C_2 = NO

X=(age<=30 , income = med, student = yes, Cerdit = Fair)

هرکدام را جدا حساب می‌کنیم

$$P(C_1) = \frac{9}{14} \qquad P(C_2) = \frac{5}{14}$$

$$P(age<=30|C_1) = \frac{2}{9} = 0.222$$

$$P(income=med|C_1) = \frac{4}{9} = 0.444$$

Subject:
Year. Month. Date. ( )

$= 0.028$

$P(\text{Student} = yes \mid C_1) = \frac{6}{9} = 0.667$ ضرب اینا $\rightarrow 0.047 \times P(C_1)$

$P(cr = F \mid C_1) = \frac{6}{9} = 0.667$ تک تک این ویژگی ها، احتمالشان را برای کلاس C_1 حساب کردیم. حالا همین کار را برای کلاس C_2

$P(age <= 30 \mid C_2) = 3/5 = 0.6$

$P(income = med \mid C_2) = 2/5 = 0.4$ حالا همه این احتمال ها را در هم ضرب کن

$P(stu = yes \mid C_2) = 1/5 = 0.2$ ضرب اینا $\rightarrow 0.019$ و در $P(C)$ اش ضرب کن $\times P(C_2) = 0.007$

$P(cr = F \mid C_2) = 2/5 = 0.4$

از C_1 بیشتر است پس این تاپل عضو کلاس C_2 یعنی yes است

عیب این روش: اگر یکی از احتمال ها صفر باشد وقتی در بقیه ضرب شود، حاصل ضرب صفر می‌شود. و کل احتمال را صفر می‌کند و باعث می‌شود تاپل عضو یکی دیگر از کلاس ها شود. چون فرض بر این است که ویژگی ها مستقل هستند بنابراین تاپل متعلق به آن کلاس نیست.

برای حل این مشکل: اگر Dataset خیلی بزرگ باشد مثلا به این احتمال

laplace correction: و در این روش برای صفر نشدن احتمال، یک تاپل به صورت تصنعی وارد می‌شود. چون مجموعه داده بسیار بزرگ است اضافه کردن یک تاپل تأثیری نخواهد داشت.

$$\text{original: } P(A|C) = \frac{N_{ic}}{N_c}$$

$$\text{laplace correction} = \frac{N_{ic} + 1}{N_c + 1}$$

مثلا income: low, med, high

مثلا income=low کامپیوتر نخریده و احتمال آن صفر است.

بنابراین برای اینکه عدالت رعایت شود به هر کلاس یک تاپل اضافه می‌کنیم پس صورت $+1$ می‌شود و مخرج $+3$ می‌شود $\frac{+1}{+3}$ برای وقتی که مجموعه داده کم باشد و نیاز به وقت بیشتری داریم.

Subject:
Year. Month. Date. ( )

مزایا: نسبت به درخت تصمیم راحت تره (فقط زمونه) که باعث میشه سرعت هم زیاد شه.

۱- پیاده سازی آسان

۲- سرعت بالا

۳- نسبت به داده نویز و پرت حساس است

۴- داده Miss . (بعضی از اطلاعات تابل از دست رفته باشد)

معایب: ۱- فرض مستقل بودن باعث کاهش دقت خواهد شد

۲- در عمل میان متغیرها وابستگی وجود دارد

Beysian Belief Network

روش های دسته بندی: ۱- درخت تصمیم ۲- روش بیزین ۳- Rule_Based

Roule_Based: قوانین را استخراج می‌کنیم

if (age = youth && student = yes) then buy-comp = yes

حالا می‌توانیم برای تابل x که جدید می‌آید بگوییم جزو کدام کلاس است.

(age = youth) && (student = yes) ⇒ buy-comp = yes

مقدم (شرط) — (استعاره از قوانین - تولید قوانین (تابل))

Condition

پوشش Cover: تمام شرط های یک تابل در یک قانون صدق کند می‌گیم Rule ،

Rule cover Tuple تابل را cover می‌کند.

روش ارزیابی: اینکه بتوانیم تعداد تابل های زیادی پیدا کنیم که این قانون آن ها را پوشش دهد.

شرط های ارزیابی یک قانون:

پوشش: $Coverage = \frac{n_{cover}}{D}$ (کل مجموعه) — به شرط اینکه درست cover کند.

دقت: $\frac{n_{correct}}{n_{cover}}$

Subject:
Year. Month. Date. ( )

حالا یک تاپل جدید X می‌آید. می‌خواهیم تشخیص دهیم متعلق به کدام کلاس است؟
آیا شرط‌های آن در چندین قانون صدق می‌کرد؟!

$X = (age < 30 , income = med , stu = yes , cre = F) = ?$

if (age = youth & cr = F)

IF (age = youth && Studen = yes) then buy-comp = yes

می‌خواهیم براساس قوانین تشخیص دهیم که x متعلق به کدام دسته است؟

۱- اگر R_1 تنها قانون برآورده شده باشد، قانون با بیشترین پیش کلاس برای X دسته آن را تعیین می‌کند

۲- اگر بیش از یک قانون trigger شود نیاز به برطرف کردن تصادم داریم چون هر قانون ممکن است یک دسته مختلف برای x مشخص کند.

برای رفع اختلاف دو استراتژی وجود دارد:

۱- size ordering بالاترین اولویت را به قانونی می‌دهد که دشوارترین شروط را برآورده کرده باشد یعنی بیشترین تعداد شرط‌ها را داشته باشد

از بالا به پایین قانون‌ها را مرتب می‌کنیم (اول قانون‌هایی که قبلاً ۴ تا شرط بعدی ۳ تا و ...)

۲- Rule ordering قانونی قابل قبول است که کلاس آن شایع‌تر است.
prevalent
Class based ordering

اگر تاپل x در هیچ قانونی برآورده نشود در این حالت باید یک دیفالت رول تعریف کنیم که احتمال آن خیلی کم است.

این نحوه استفاده از قوانین بود.

درخت تصمیم

نحوه تولید قوانین: با استفاده از قوانین یا استراج کنیم.

حالا چرا در روش rule-base از درخت تصمیم استفاده می‌کنیم؟
چون ممکن است درخت تصمیم پیچیده شود. تا آن را تبدیل به قانون می‌کنیم.

Subject:
Year. Month. Date. ( )

در درخت تصمیم: برای هر مسیر از ریشه تا برگ یک قانون ایجاد می‌شود.

* هر زوج مقدار خصوصیت در امتداد مسیر یک ترکیب عطفی می‌سازد

برگ، برچسب کلاس است.

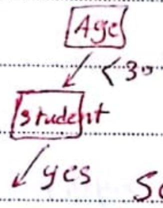
مهم ترین مزیت: قانون‌ها دو به دو ناسازگار می‌شوند Mutual exclusive

یعنی یا این قانون یا آن. دو قانون trigger نمی‌شوند برای یک تاپل

جامع است و کامل یعنی هرچه بخواهیم می‌توانیم در این قانون‌ها پیدا کنیم.

مشکل size order و ... نداریم با قانون درنهایت.

Age<30 && student = yes مثلا



منظور از * : مثلاً

الگوریتم پوشش ترتیبی: Sequential Covering Algorithm

مرحله اول هیچ شرطی نمی‌ذاریم

همه اطلاعات دیتابیس cover می‌شوند اول کار

هر مرحله یک شرط می‌ذاریم.

برای مثال وام.

در سطح ۲ کدام قانون را قبول کنیم؟

قانونی که تعداد تاپل بیشتری را با دقت خوب پوشش دهد

از بالا به پایین پوشش کاهش می‌یابد

اما دقت افزایش می‌یابد

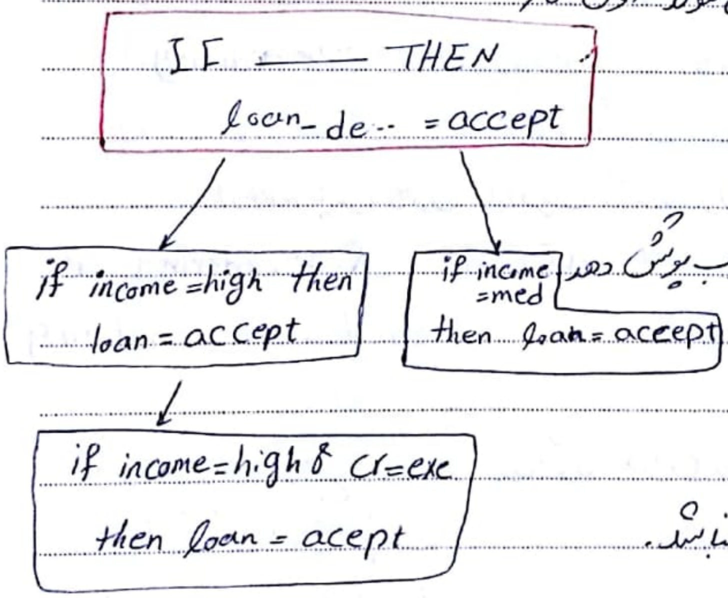
تا کجا شرط بذاریم؟

یا اینکه از تاپل هایی که داریم هیچ مثالی نباشد.

بین دقت و پوشش به یک تعادلی برسند

در هر مرحله یک سری شرط به آن اضافه می‌کنیم.

قوانین را به صورت مستقیم از داده‌های آموزشی استخراج می‌کنیم.



گام های الگوریتم:

۱- از مجموعه داده آموزشی مرحله به مرحله قانون ایجاد کنند.

۲- هر قانونی که آموخته می‌شود تاپل‌هایی که توسط آن قانون آموخته می‌شوند حذف می‌شوند

این کارها تا زمان شرط پایان ادامه دارند.

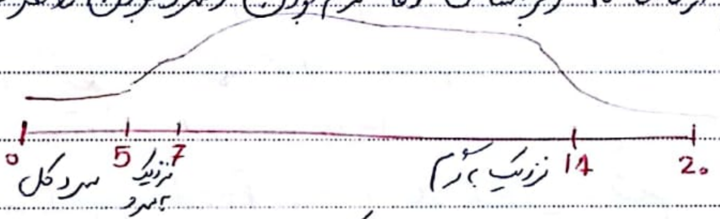
شرط های پایان: تا جایی که هیچ مثال آموزشی دیگری باقی نمانده باشد

یا جایی کیفیت قانون ایجاد شده کمتر از حد مشخص نباشد.

پوشش و دقت

منطق فازی: مثلاً می‌گیم هوا گرمه یا سرده که این گرم یا سرد بودن نسبی است

یک مقیاس دیتا می‌ذاریم از ۰ تا ۲۰ و بر اساس آن گرم بودن و سرد بودن را تعریف می‌کنیم.



پس منطق فقط صفر و یک نیست و می‌توان درجه‌بندی کرد.

مثلاً: در یک رستوران می‌خواهند به یک نفر انعام بدهند. تحت تاثیر دو عامل سرویس و کیفیت غذا

بر اساس ترکیب این دو مقدار انعام را مشخص می‌کنیم، این ها متوانین فازی می‌شوند.

Service	Food	Tip
poor	deliciam	✓
avg	⋮	
good	⋮	

• یک جعبه ابزاری داریم برای فازی

fuzzy

مثلث: اینجا می‌توانیم سیستم رستوران‌ها را بنویسیم. می‌زنیم

تو صفحه کامند ویندو اینجا وارد فازی تول می‌شیم

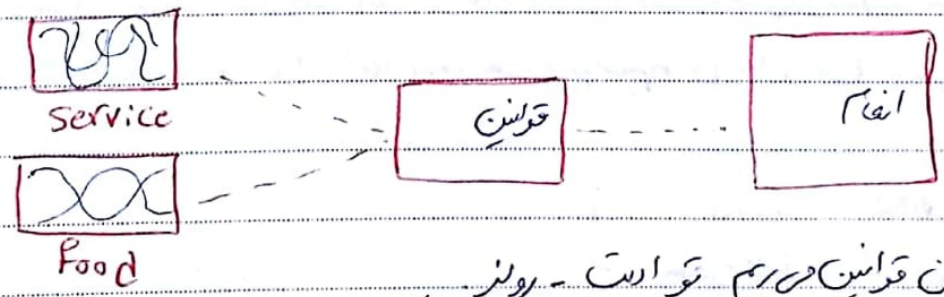
Subject:

Year. Month. Date. ( )

پنجره FIS که همان فازی تول است. یک گزینه فایل داریم که می‌توانیم ایمپورت و اکسپورت کنیم.

الان می‌خواهیم مسئله انعام را پیاده‌سازی کنیم. از سربرگ ادیت متغیر اضافه می‌کنیم.
۲ ورودی و یک خروجی داریم. متغیر اول سرویس. متغیر دوم غذا. خروجی T
در هر کدام باید رنج تعریف کنیم و شکل ها را مشخص کنیم.

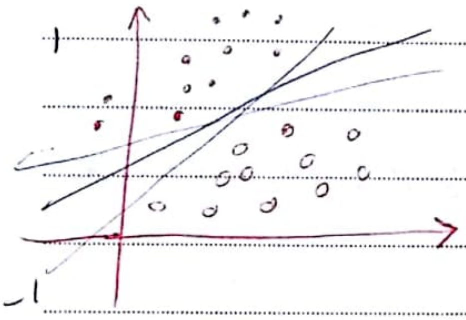
این متغیرها یک سری توابع دارند. میایم بازه ها را تغییر بدهیم. می‌گیم سرویس از ۰ تا ۰.۴ poor است
از 0.5 متوسط است.
می‌توانیم این رنج ها را جابه‌جا کنیم. (از روی شکل توی نرم‌افزار)
اگر این ها را اشتباه کردیم می‌توانیم روی آن کلیک راست کنیم و بزنیم Remove
می‌توانیم شکل های دیگری هم اضافه کنیم. سربرگ ادیت گزینه ادد. که این شکل ها فرمت های متفاوتی دارند. مثلاً هلالی - ذوزنقه‌ای
این رنج ها را برای food و service مشخص می‌کنیم.
برای انعام هم باید رنج را مشخص کنیم.
قوانین که فازی روی آن ها کار می‌کند را بیان می‌کنیم



برای نوشتن قوانین می‌ریم تو ادیت - رولز.
ارتباط بین دو عامل را قسمت پایین مشخص می‌کنیم. مثلاً ارتباط or باشد یا And
اینکه قوانین به چه شکلی باشند بستگی به ما دارد.
قبل از مشخص کردن قوانین باید یک دید نسبت به داده‌ها داشته باشیم.
می‌توانیم نمودار این قوانین را ببینیم. plot
با این plot به ما نشان می‌دهد که خروجی های ما روی چه حالاتی هستند.

Subject:
Year. Month. Date. ( )

سایت freepaper ـ لینک مقاله‌هایی که از توی سایتش داریم پیدا کردیم را کپی می‌کنیم و اینجا پیست می‌کنیم به این صورت مقاله را می‌دزدیم :)



SVM: یکی از الگوریتم‌های دسته‌بندی است

فرض کنید نقاط در ویژگی x_1 و x_2 داریم.

ساده‌ترین راه دسته‌بندی این داده‌ها این است که یک خط بکشیم ـ تمام الگوریتم‌های دسته‌بندی کارشان این است.

دو تا کلاس داریم کلاس ۱ و کلاس ۱-.

دیتاها را طوری جدا کنیم که توی دو تا کلاس بیفتند.

تعداد بی‌نهایت خط بین این دو کلاس می‌توانیم داشته باشیم که دسته‌بندی را درست انجام می‌دهد.

در تمام الگوریتم‌های قبلی مهم نبود که کدام خط رسم می‌شود و فقط رسیدن به یک خط مهم بود.

اما در SVM رسیدن به یک خط بهینه مهم است؟ اما خط بهینه کدام است؟

یک خط وسط ـ که حداکثر فاصله را از هر دسته داشته باشد.

فاصله خط از اولین دیتای این کلاس بیشترین باشد.

در هر کلاس داده‌هایی که به هم خیلی نزدیک‌اند برای ما مهم هستند. چون آن‌ها تعیین می‌کنند که خط کجا باشد؟

نیاز است که حداقل ۲ تا دیتا روی این خط اثر بگذارند (حداقل) هر دیتا مال یک کلاس

اما اگر نقاط بیشتری هم پیدا کردیم، مثلاً تا بی‌نهایت، اشکال ندارد.

هیچ دیتایی حق ندارد در margin قرار بگیرد که این بسیار سخت‌گیرانه است hard margin

margin یک حاشیه است.

ما الان مسئله را ساده‌سازی کردیم و گفتیم تفکیک‌پذیر است با یک خط

اما در واقع اینگونه نیست

soft margin ← یک درصد خطا توی دیتای آموزشی داریم. یعنی یک سری داده‌ها توی margin هستند.

معادله خط $w_0 + w^t x = y^i$ (w^t برای اینکه توی بعد بیشتری می‌بریم)

y^i همان lable ها هستند.

w_0 و w^t براساس داده‌ای که داریم بدست میاریم و وقتی داده جدید می‌آید x است.

Subject:
Year. Month. Date. ( )

حالا دوتا فرمول را ارائه میدیم برای مثالمون : if: $wx + w_0 \geq +1$ class +1

if $wx + w_0 \leq -1$ class −1

هرچقدر به margin بزرگتری برسیم بهتر است ← درست تر کلاس بندی میکنیم.

این دو تا معادله را تبدیل به یک معادله میکنیم، طرفین آن را در y^i ضرب میکنیم
برای راحتی مساوی هم در نظر داریم.

$$\left.\begin{matrix} wx + w_0 \geq 1 \\ wx + w_0 \leq -1 \end{matrix}\right\} \Rightarrow y^i(wx + w_0) \geq 1$$

فرض کنید دو تا نقطه داریم که روبروی هم اند (ساده ترین حالت) چگونه خط بین آن ها را
پیدا کنیم که margin آن ماکزیمم باشد؟

این سه خط موازی اند ← یعنی بردار نرمال هر ۳ یکی است.

x^+ و x^- داده هستند.

Subject:
Year. Month. Date. ( )

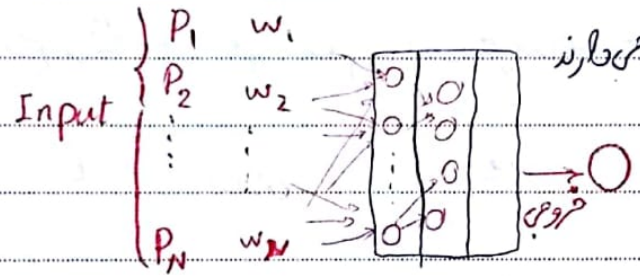
شبکه عصبی: فرض کنید میزان آلودگی هوا - همه عوامل ما برای یک سال متوالی داریم در کنار آن دمای هوا، تعداد ماشین ها و چند تا پارامتر دیگر هم داریم که میزان پراکندگی آن را یقین می کند. چند تا ورودی داریم و یک خروجی.

یک مدل استخراج کنیم که بتوانیم میزان آلودگی فردا، یک هفته بعد و... را بگیم.

مثل هوشناسی. داده های یک سال قبل، دو سال قبل را به عنوان آموزش می دهد.

یک سری ورودی داریم.

هرکدام از ورودی ها یک تأثیری در خروجی دارند که آن را با وزن نشان می دهیم.



شبکه عصبی مجموعه ای نورون هاست.

چند تا لایه داریم. و در هر لایه چندین نورون داریم و در نهایت آن را به خروجی می رسانیم.

نورون ها P_i و w_i ها را می گیرند. ورودی نورون $P_i w_i + b_i$

هر دفعه که ما آموزش می دهیم کاری می کنند که w_i ها و b_i ها پیدا کند.

اینقدر به آن آموزش می دهیم که این w و b ها دارای بیت تنظیم کنیم.

مثلاً بر اساس تعداد پنجره ها و تعداد افراد درون اتاق دمای اتاق را یقین می کند

سفارشیم دما شبه ۲۰ درجه. اول پنجره باز می کنیم و پنج نفر می بینیم دما شد ۳۰ درجه.

بعد دو تا پنجره را باز می کنیم. اینقدر با تعداد پنجره ها و تعداد افراد بازی می کنیم.

تا به خروجی مدنظر برسیم. پس تعداد w_i ها و b_i ها تغییر می کند تا به خروجی مطلوب برسیم یعنی می گوییم آموزش دیدیم.

مدل از روی دمای سال ۲۰۱۷ ساخته شده (پس دمای ۲۰۱۷ جزء دمای ترین)

دمای ۲۰۱۸ را بخش دیدیم دمای ۲۰۱۸ جزء دمای تست.

هر کدام از این نورون ها یک تابع دارند.

تو سلب از داخل اپلیکیشن ها پیدا کنیم یا تو کامند بزنیم nn start

برای حل مسائل چندین روش داریم که این پایین نوشته.

Subject:
Year. Month. Date. ( )

ما مقلا input-output را می‌بریم. Fiting tool شکل شبکه عصبی را به ما نشان میدهد
می‌زنیم load Dataset جورش یک سری Dataset داره

یک سری مجموعه داده را به عنوان آموزش می‌ذاریم. یک سری برای اعتبارسنجی که مدل
میاد اگه خروجیش با مقدار مورد نظر یکی نبود، w ها را تغییر میده تا به میزان درست برسه.
یک سری دیتا هم برای تست که به مدل می‌دیم برای تست کردن. خروجی‌های واقعی را
داریم و نگاه می‌کنیم ببینیم که مدل چند درصد خطا میده

MSE خطای متوسط مربع :

رگرسیون : ارتباط بین خروجی واقعی و خروجی بدست آمده

Simple script ← کد این دستورات را می‌بینیم

ادامه SVM. : سه تا خط موازی داریم. خط وسط ← خط جداساز
هرکدام از این‌ها باهم فاصله دارند که این فاصله p می‌باشد حالا یا کل فاصله و یا
فاصله بین خط وسط و اول یه خط را p میگن.

(x, y_i) x تاپل ما می‌باشد y_i کلاس آن

+1 −1

برای اینکه ببینیم x در کدام دسته است ←

ویژگی محدود کننده { $w^t x + b \geq +1$ ، $w^t x + b \leq -1$ } این 1+ و 1- هم همان p است آن را با lable اشتباه نگیریم.

هارد مارجین : داده‌ها در بین این دو بازه قرار نگیرند
که این خیلی کم اتفاق میافتد.
سافت مارجین : اشکال نداره که داده‌ها بین این دو بازه قرار بگیرند

Subject:
Year. Month. Date. ( )

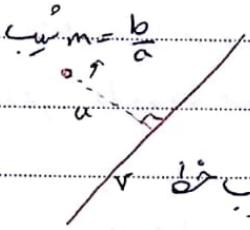
حالا این محدودیت ها را در y_i ضرب می کنیم
هر دو محدودیت تبدیل به یکی می شوند.

$$y_i(w^t x+b) \geq +1$$

می خواستیم فاصله نقاط تا خط حداکثر شود :

$$\frac{|w^t x+b|}{\|w\|} \geq \rho$$ فاصله نقطه و خط

$$\Rightarrow |w^t x+b| \geq \rho \|w\|$$



معادله خط $= ax+by+c=0$

شیب خط $m = -\frac{a}{b}$

فاصله نقطه و خط $= \frac{|ax+by+c|}{\sqrt{a^2+b^2}}$

مطمئن نیستم، خودتان چک کنید.

ρ عادی حداکثر بود برای اینکه فاصله بیشتر شود، ρ را ماکزیمم می کنیم.

فرض می کنیم

$$\rho\|w\|=1 \Rightarrow \rho = \frac{1}{\|w\|}$$

پس $\|w\|$ باید min شود

$$\min \|w\| \rightarrow \min \frac{1}{2}\|w\|^2$$ ← بسته به بهینه سازی

$$s.t. \quad y_i(w^t x+b) \geq 1$$

(برای اینکه از فرمول لاگرانژ استفاده کنیم.)

وقتی می خواهیم $\|w\|$ را min کنیم می توانیم هر ضریبی از آن را min کنیم

این را به مسئله اضافه می کنیم که چقدر نسبت به خطاها حساس باشد.
خطاهای هر کدام از آن ها را با ε در نظر می گیرند. میزان خطا بر اساس خطوط اطراف در نظر می گیرند

ضریب دلخواه

$$C \sum_{i=1}^{n} \varepsilon_i$$ مجموع خطاهای حاشیه

پس می خواهیم خطاها min سازی شوند. برای همین به راحتی نمی توانیم ρ را max کنیم
بلکه $\|w\|$ را min کردیم.

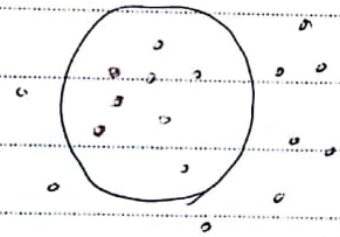
Subject:
Year. Month. Date. ( )

C ← میزان خطایی که در نظر می‌گیریم. اگر بخواهیم سخت‌گیری کنیم به طور مثال C را صفر در نظر می‌گیریم - هر چقدر بخواهیم سخت‌گیری کمتر شود C را بزرگتر در نظر می‌گیریم. C به صورت تجربی بدست می‌آید

داده‌ی آموزشی را می‌دهیم تا این خط مشخص شود.

به وسیله خط می‌توان داده‌ها را از هم جدا کرد ← خطی جدایی پذیر

اما بعضی داده‌ها خطی جدایی ناپذیرند - با خطا می‌توان آن‌ها را جدا کرد.



روش کرنل: تعداد ابعاد را آنقدر بالا می‌بریم که خطی جدایی پذیر شوند.
بتوان با یک صفحه آن‌ها را جدا کرد.
دو ویژگی (x_1, x_2) را داریم. بر اساس این دو ویژگی سوم را می‌سازیم

$$\phi : x \rightarrow \phi(x)$$

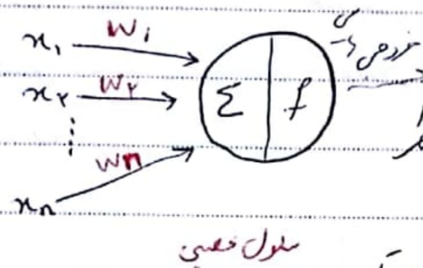
الان ویژگی سوم را داریم.

الان خطی جدایی پذیر شدند $(x_1, x_2, x_1^2 + x_2^2)$

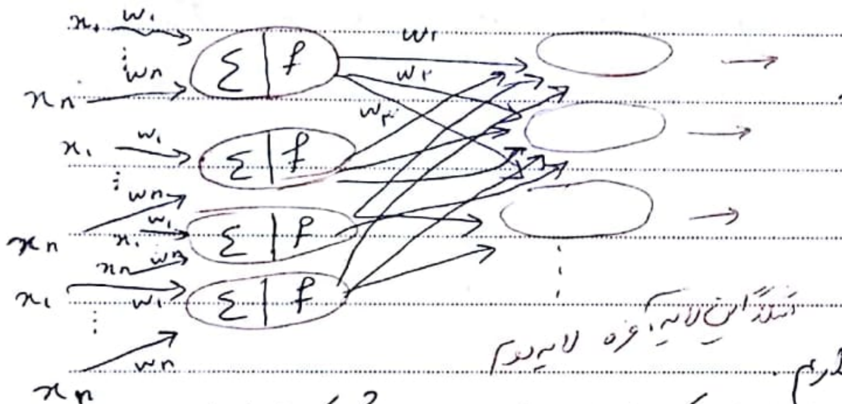
الان ساده‌ترین حالت ممکن بود با اضافه کردن یک بعد به جواب رسیدیم.
اما بعضی وقتا باید خیلی بعد اضافه کنیم.

Subject:
Year. Month. Date. ( )

شبکه عصبی: $w_1x_1 + w_2x_2 + \dots + w_nx_n$



مجموعه‌ای از سلول‌های عصبی می‌شود شبکه عصبی.
براساس مسئله و تجربیات شبکه عصبی را لایه به لایه در نظر می‌گیریم. مشخص می‌کنیم چند لایه داشته باشیم.
این که در هر لایه چند تا سلول داشته باشیم هم تجربی است.



w_i ها کاملاً با هم متفاوت‌اند.

$$f(z) = \frac{1}{1+e^{-z}}$$

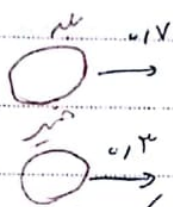
لایه آخر: مثلاً ۴ تا کلاس داریم.
۴ تا نورون در نظر می‌گیریم. هر کدام یک خروجی دارند ← بحث کلاس‌بندی
خروجی براساس تعداد کلاس‌هایی که داریم.
تعداد نورون‌های ورودی براساس تعداد ویژگی‌ها تعیین می‌شود.
برای مسائل پیش‌بینی به تعداد خروجی‌هایی که می‌خواهیم در لایه آخر نورون می‌گذاریم.
فرض کنید $x_1 \dots x_n$ همگی صفر باشد. در این صورت هیچ کدام از w ها و f ها تأثیر ندارد و خروجی نیز صفر می‌شود.
ورودی بین صفر و یک است. اگر بین صفر و یک نباشد نرمالایز می‌کنیم.
اینجا هم یک بایاس اضافه می‌کنیم b که w_b دارد که آن هم تنظیم می‌شود.
b بایاس مقدار یک ضرب در w_b است. بایاس یک مقدار بین ۰ و ۱ است.

$b = 1 \times w_b$

داده‌های آموزشی را به این شبکه می‌دهیم. مقدار اول این w ها را رندوم در نظر می‌گیرد.
خروجی یک عدد بین ۰ و ۱ است (گربه=۱ و غیره=۰)
هر چه نزدیک‌تر به یک خروجی می‌دهد.
یک خطایی داریم. جواب اصلی باید یک باشد اما خروجی ما ۰٫۷ است.

Subject:
Year. Month. Date. ( )

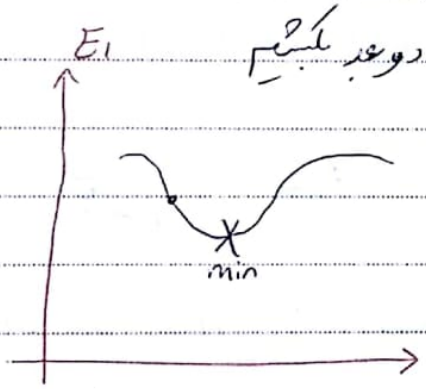
خروجی باید ۰ و ۱ باشد اما نیست



پس خطا داریم. هرچقدر خطا کمتر باشد
یعنی W ها را درست تنظیم کنیم.
این برای یک داده است. مثلاً هزار تا داده ورودی شبکه داشته باشیم.

مثلاً $\Sigma (1 - 0/7) = 0/3 = 0/15$

بر اساس این ها یک نمودار خطا تشکیل می دهد.
فرض می کنیم دو تا وزن داریم که بتوانیم در دو بعد بکشیم



E_1 : میزان خطای w_1
E_2 : میزان خطای w_2

در مرحله ترین خطا را پیدا می کنیم
می خواهیم خطا min شود. برچه اساس w_1 و w_2 را تغییر دهیم که این به سمت کمینه شدن حرکت کند؟ مشتق می گیریم. با ثبت این نقطه (min نسبی و محلی زیر سال) بر اساس این ها مشخص می کنیم مقدار w ها چگونه تغییر کند.
این که چه میزان خطا کم شود، چقدر به min نزدیک شویم به حاشیه دارد.
در بعدی زیاد، تعیین این کار سخت می باشد.
یک تعداد تکرار مشخص می کنیم. مثلاً در این تعداد تکرار خطا را min کنیم
یا اینکه مشخص می کنیم مجموع خطا باید از این مقدار کمتر باشد.
زمانی شبکه آموزش دیده شد یعنی w ها تعیین شدند و بعد شبکه را Test می کنیم.
آموزش در یک مرحله نیست. اینکه چند بار داده به شبکه نشان داده شود را تعیین می کنیم.
در مرحله test جواب خانی را نمی گیرد. خروجی را ارائه می دهد و با جواب خانی مقایسه می کند تا ببیند چقدر خطا دارد.
وقتی داده ها به عنوان تست بیش داریم. یعنی داده جدید به یادگیری پیش بینی می کند.

Subject:
Year. Month. Date. ( )

سؤال ۱: Apriori

1	$\{I_1, I_2, I_3, I_4, I_5, I_6\}$
2	$\{I_2, I_3, I_4, I_5, I_6, I_7\}$
3	$\{I_1, I_7, I_5, I_8\}$
4	$\{I_1, I_4, I_6, I_9, I_{10}\}$
5	$\{I_2, I_4, I_5, I_{10}, I_{11}\}$

Min support = 60%

	Support
$\{I_1\}$	3 *
$\{I_2\}$	3 *
$\{I_3\}$	2
$\{I_4\}$	5 *
$\{I_5\}$	4 *
$\{I_6\}$	3 *
$\{I_7\}$	1
$\{I_8\}$	1
$\{I_9\}$	1
$\{I_{10}\}$	2
$\{I_{11}\}$	1

$\{I_1, I_2\}$	1
$\{I_1, I_4\}$	3 * *
$\{I_1, I_5\}$	2
$\{I_1, I_6\}$	2
$\{I_2, I_4\}$	3 * *
$\{I_2, I_5\}$	3 * *
$\{I_2, I_6\}$	2
$\{I_4, I_5\}$	4 * *
$\{I_4, I_6\}$	3 * *
$\{I_5, I_6\}$	2

1,2,4	0
1,4,5	0
1,4,6	0
2,4,5	3 * *
2,4,6	0
4,5,6	0

اینجا closed شد
دیگر ادامه نمیدهیم.