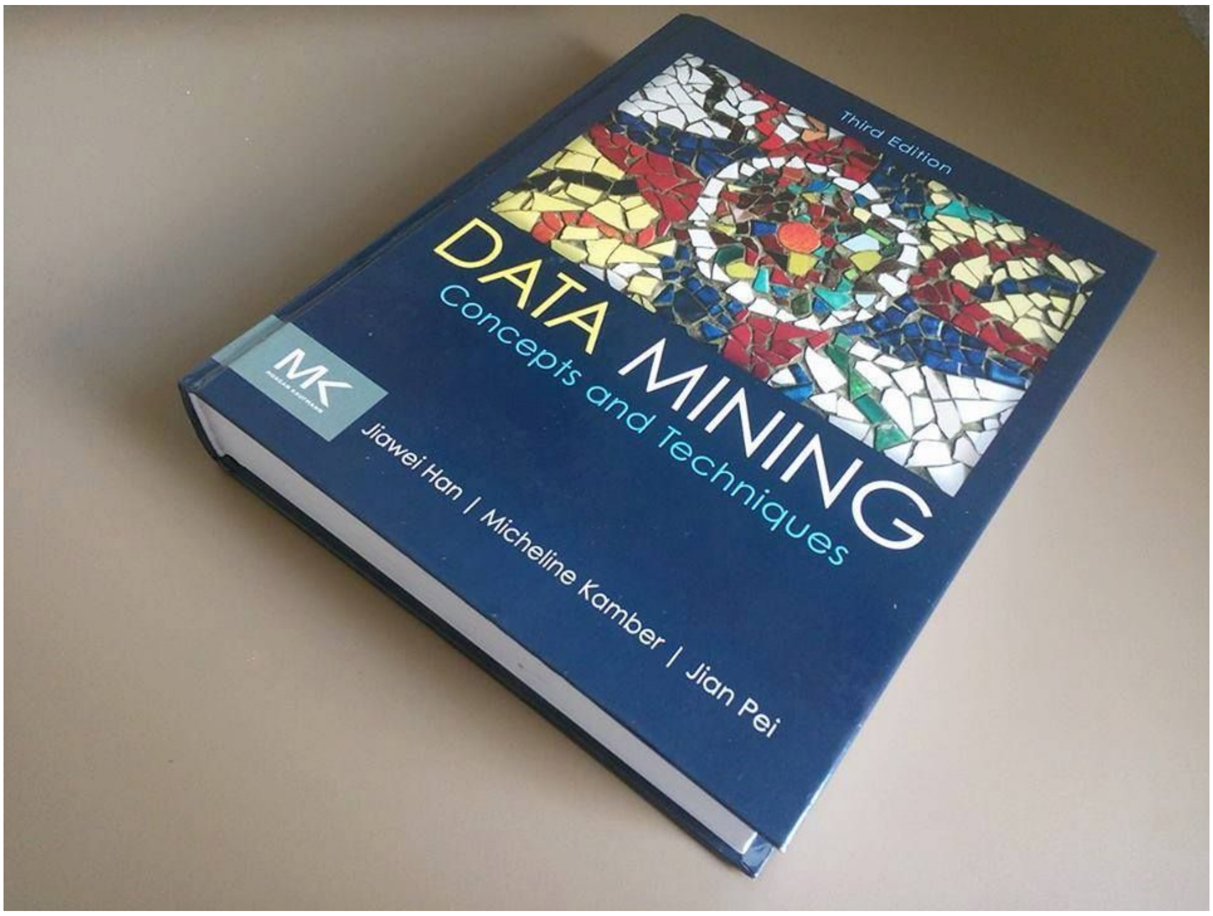
Zdroj:



https://www.amazon.com/Data-Mining-Concepts-Techniques-Management-ebook/dp/B0058NBJ2M

1. Predspracovanie - na co je dobre predspracovanie, preco naco zaco

today’s real-world databases are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size. Low-quality data will lead to low-quality mining results. There are several data preprocessing techniques. *Data cleaning* can be applied to remove noise and correct inconsistencies in data. *Data integration* merges data from multiple sources into a coherent data store such as a data warehouse. *Data reduction* can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering. *Data transformations (normalization)* may be applied, where data are scaled to fall within a smaller range like 0.0 to 1.0. All of these can improve the accuracy and efficiency of mining algorithms involving distance measurements.
Data have quality if they satisfy the requirements of the intended use. There are many factors comprising data quality, including accuracy, completeness, consistency, timeliness, believability and interpretability. Data are incomplete if they lack attribute values or certain attributes of interest, or contain only aggregate data. Data are inaccurate or noisy if they contain errors, or values that deviate from the expected. Data are inconsistent when they have discrepancies in the department codes used to categorize items (for example).
Numerosity reduction is when data are replaced by alternative, smaller representations using parametric models (regression or log-linear models) or nonparametric models (histograms, clusters, sampling, data aggregation)

2. Ako mozu vzniknut chybajuce hodnoty, ako ich nahradit - aspon 3 sposoby a ich vyhody/nevyhody

Imagine that you need to analyze AllElectronics sales and customer data. You note that many tuples have no recorded value for several attributes such as customer income. How can you go about filling in the missing values for this attribute? By some of these:

- Ignore the tuple (row) – this is usually done when the class label is missing (if we do classification). This method is not very effective, unless the tuple contains several attributes with missing values.

- Fill in the missing value manually – in general, this approach is time consuming and may not be feasible given a large data set with many missing values
- Use a global constant to fill in the missing value – replace all missing attribute values by the same constant such as a label like „Unknown“. 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“
- Use a measure of central tendency for the attribute (mean or median...) to fill in the missing value, for normal (symmetric) data distributions, the means can be used, while skewed data distribution should employ the median
- Use the attribute mean or median for all samples belonging to the same class as the given tuple, for example, if classifying customers according to credit_risk, we may replace the missing value with the mean income value for customers in the same credit risk category as that of the given tuple.
- use the most probable value to fill in the missing value, which may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction.

3. Co je to binning a ako funguje

Binning: binning methods smooth a sorted data value by consulting its „neighborhood“ that is, the values around it. The sorted values are distributed into a number of „buckets“, or „bins“. Because binning methods consult the neighborhood of values, they perform local smoothing. In smoothing by bin means, each value in a bin is replaced by the mean value of the bin. Similarly, smoothing by bin medians can be employed, in which each bin value is replaced by the bin median. In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value.

4. Asociacne pravidla

Frequent patterns are patterns (itemsets, subsequences, substructures) that appear frequently in a data set.
frequent itemset: {PC, laptop, mobile}
frequent sequential pattern: {1. PC, 2. laptop, 3. mobile}
frequent substructure: <tree, lattice, graph>
Frequent patterns and association rules are the knowledge that we want to mine in a scenario, when we are a sales manager and we are talking to a customer who recently bought a PC and a digital camera from the store. What should we recommend to him next? Information about which products are frequently purchased by our customers following their purchases of a PC and a digital camera in sequence would be very helpful in making our recommendation. Frequent patterns are patterns that appear frequently in a data set. For example, a set of items, such as milk and bread, that appear frequently together in a transactional data set is a frequent itemset. A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a frequent sequential pattern.

Definícia podpory a spoľahlivosti aplikovaná na príklade odporúčania produktov v e-shope.

Nech je L={I1,I1,...,Im} množina produktov (M-itemset). Takto definovanú množinu budem využívať často, bude ňou napr. množina {I1, I2, I3} v riadku tabuľky s transakčnými dátami.

Nech sú „task relevant“ dáta množinou takých databázových transakcií, kde každá transakcia T je neprázdnou množinou a zároveň podmnožinou množiny produktov L (**nevlastnou**). To znamená, že všetky riadky tabuľky produktov patria do množiny produktov L. Vyzerá to nasledovne:

Transactional ID	N-itemset (budem počítať 1, 2,3 a 4-itemsety
1234 (súčasť transakcie a neprázdnej množiny)	Množina N-itemset teda napr. 1-itemset je 1 produkt {I1}

Každá množina transakcií je jednoznačne spojená so svojim identifikátorom ktorý nazývame TID. Nech je A množinou produktov. Hovoríme, že transakcia T (T je neprázdna množina transakcií, skrátene transakcia) obsahuje množinu produktov A ak je množina produktov A nevlastnou podmnožinou neprázdnej množiny s

transakciami T. Jednoducho povedané, transakcia T obsahuje A, ak sa v T nachádza stĺpec s hodnotou A. Ak je každý prvok množiny A prvkom množiny T potom je logické, že A je podmnožina T a teda T obsahuje A. Asociačné pravidlo je taká implikácia A => B (implikácia, teda odvodenie), kde A je **vlastnou** podmnožinou množiny produktov L (to znamená, že A obsahuje aspoň jeden produkt ktorý nie je v L), kde B je tiež vlastnou podmnožinou množiny L, kde množina A ani množina B (vlastné podmnožiny sú tiež množiny) sú neprázdne množiny a kde ich prienik neexistuje (tomu sa hovorí disjunkcia, neexistuje taký produkt ktorý by bol v A aj v B). Táto implikácia A => B v množine transakcií D (množine množín) **je podporená** podporou S, ak S je percento takých množín transakcií (riadkov v tabuľke) v množine transakcií D (datasete) ktoré vznikli zlúčením množín A a B. Teda ak chceme vypočítať “support” predelíme počet riadkov obsahujúce hodnoty A alebo B počtom všetkých riadkov v datasete. Podpora S je podiel počtu množín transakcií na množine všetkých transakcií D pre ktoré (pre ten čitateľ zlomku) platí, že počet transakcií v čitateli je počet transakcií z A + počet transakcií z B (pretože zlúčenie dvoch množín je ich sčítanie). Podpora S máva priradenú vlastnú pravdepodobnosť P(A alebo B). Pravidlo A => B (pravidlo, implikácia, odvodenie je to isté...) je vierohodné a má priradenú vierohodnosť C v množine transakcií D, ak C je percento takých množín transakcií z množiny D pri ktorých platí, že tieto transakcie vznikli prienikom množín A a B. Keď počítame pravdepodobnosť prieniku dvoch udalostí (výskytu niakych hodnôt) tak násobime, násobime pravdepodobnosť A B-čkovou. Keď to napíšeme inak vznikne toto:
Podpora(A=>B) = P(A alebo B) #toto je tzv. relative support, absolute support je frekvencia výskytu konkrétneho N-itemsetu
Vierohodnosť(A=>B)=P(B za predpokladu A)

Poznáme dva typy frekventovaných množín, vysvetlím ich na tom istom príklade e-shopu.

Uzatvorená množina transakcií (produktov) je množina produktov X v množine transakcií D, ak je táto množina transakcií X uzatvorená **a** frekventovaná na množine transakcií D.
Množina produktov X sa nazýva uzatvorenou na množine transakcií D, ak k množine X neexistuje „proper super-set Y“. **Táto definícia platí len v prípade, že množina X má také isté početnosti/frekvencie ako množina Y. Keď chcete hovoriť odborne nehovorte, že množina X má rovnaké frekvencie ako Y ale povedzte že, množiny X a Y majú rovnakú support count. Fakt neviem ako to normálne preložiť.**
Proper super-set Y sa definuje nasledovne:
Y is a proper super-itemset of X if X is a proper sub-itemset of Y, that is, if X is a subset of Y. In other words, every item of X is contained in Y but there is at least one item of Y, that is not in X.

Takže to nie je vôbec zložité, definícia hovorí o vlastnej podmnožine. X je teda vlastnou podmnožinou množiny Y pretože v množine X existuje aspoň jeden taký produkt (z e-shopu), ktorý sa nenachádza v množine produktov Y. That's all.

Pravidlo sa nazýva silné ak spĺňa podmienku minimálnej vierohodnosti a podpory. Počítam to v 50. otázke. Support a confidence pravidla väčšinou nedostanete v tvare 0.1 ale v percentách. Za ich stanovenie sú zodpovední znalostní inžinieri, neexistuje vzorec na ich stanovenie, aké by mali byť.

Pri počítaní support count teda frekvencie výskytu niakeho vzoru **sa vždy pozerajte do datasetu** a nepočítajte ho z pomocných tabuliek. Ukazujem to v 50. otázke.
Ak bude zadaná otázka, ktoré z nasledujúcich itemsetov je frekventované, pozrite sa na zadanú minimálnu absolútnu početnosť Support Count (volá sa to threshold) a ak je vaša početnosť vyššia ako zadaná hodnota maximálne rovná, potom je váš itemset frekventovaný.

3-itemsets (N je vždy počet prvkov)	Support Count
{I2,I3,I4}	číslo ktoré hovorí, koľkokrát sa nachádza {I2, I3, I4} v trénovacom datasete
{I5,I1,I3}	- || -

Vyššie definujem ako vyzerá, keď je itemset closed teda uzatvorený. Stane sa to vtedy, keď vo vašej tabuľke neexistuje taký N-itemset ktorý by mal takú istú absolútnu početnosť ako ten váš. Ak máte vybrať frekventovaný a uzatvorený N-itemset potom najprv porovnajte absolútnu početnosť so zadanou a ak je itemset frekventovaný a nenájdete iný s tou istou početnosťou, potom je váš vybraný itemset uzatvorený (closed) a frekventovaný zároveň. Itemset sa nazýva maximálne frekventovaným, ak nenájdete iný s vyššou absolútnou početnosťou. Toto však platí iba ak svoj itemset porovnávate s frekventovanými. Ono to aj platiť bude, pretože by sa nemalo stať že by bola niaka početnosť vyššia ako tá vaša a ešte by bola menšia ako zadané "minimum support threshold". Stane sa to iba vtedy, ak znalostnému inžinierovi stačí malá vierohodnosť/frekvencia N-itemsetov.

5. Odpoveď na otázku týkajúcu sa rozdielu medzi dolovaním v relačnej a transakčnej databáze.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.438.6020&rep=rep1&type=pdf

Odpoveď: na dolovanie *frequent patterns* a na využitie asociačných pravidiel **sú potrebné alternatívy k ich implementácií v transakčných databázach.** Ako alternatívy technológií, ktoré sa používajú v transakčných databázach môžeme uviesť:

- Loose-coupling cez SQL cursor-interface
- Encapsulation of a mining algorithm in a stored procedure
- Caching the data to a file system on-the-fly and mining
- Tight-coupling using primarily user-defined functions
- Implementácie SQL pre ich processing v DBMS.

A relational database is a collection of tables, each of which is assigned a unique name. Each table consists of a set of attributes and usually stores a large set of tuples. Each tuple in a relational table represents an object identified by a unique key and described by a set of attribute values. A semantic data model, such as an entity-relationship data model is often constructed for relational databases. **When mining relational databases**, we can search for trends or data patterns. For example, data mining systems can analyze customer data to predict the credit risk of new customers based on their income, age, and previous credit information. **Relational databases are accessed by database queries which are transformed into a set of relational operations for efficient processing. We usually build Data Warehouses for this kind of databases and then use query and analysis tools to discover patterns from them. Relational databases could be mined by these methods:**

- **multidimensional data mining methods**

A transactional database captures a transactions, such as a customer's purchase, a flight booking, or a user's clicks on a web page. A transaction typically includes a unique transaction identity number (TID) and a list of the items making up the transaction, such as the items purchased in the transaction. **Transactional data are mined by mining frequent patterns from them, which represent sets of items that are frequently coupled together.**

6. Nevýhody Apriori a ako ich FP nahrádza

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules. Boolean rules are those rules, which have on:

- left side variable representing (for example) presence or absence of items in a store. So if 1 corresponds to “available” and 0 “absence” this could be Boolean vector X=(1,0,1,1,1,0)
- right side variable which can be also represented as on a left side

The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties. This algorithm employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets. We do this by conjucting first columns from tables (50. otázka).First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item (frequency of each itemset), and then collecting those items that satisfy minimum support threshold (minimum frequency). The resulting set is denoted by Lx. Next, Lx is used to find Lx+1, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found. **The finding of each Lx requires one full scan of the database.**

Tento algoritmus sa dá vysvetliť na nasledujúcom príklade:
Majme databázu produktov D obchodu predávajúceho elektroniku. Táto tabuľka vyzerá nasledovne:

TID	Zoznam produktov so svojimi Idckami
T1	I1,I2,I5
T2	I2, I4
T3	I2, I3
T4	I1, I2, I4
T5	I1, I3
T6	I2, I3
T7	I1, I3

Je to databáza transakcií, každý riadok tabuľky je transakcia.
V počiatočnom stave je každý produkt Ix prvkom svojej jedno-prvkovej množiny. Tieto jedno-prvkové množiny nazývame 1-itemsety. V nich sú produkty. Na vytvorenie prvej frekvenčnej tabuľky potrebuje prehľadať celú databázu D. Vytvorí prvú frekvenčnú tabuľku C1, v každom riadku tejto tabuľky bude jednoprvková množina s produktom obchodu a k nej prislúchajúca početnosť výskytu tohto produktu v databáze D

Tabuľka C1

1-Itemset	Support count
{I1}	7 #7 krát sa nachádza v databáze D
{I2}	6
{I3}	8 #8 krát sa nachádza v databáze D, C1 nie je zoradená
{I4}	2
...	

{I1} až {I4} sú množiny produktov, spolu tvoria množinu kandidátov. Funkcia algoritmu zvyčajne príjima parameter **min_sup**, ktorý je rovný niakemu celému číslu. Dajme tomu že bude rovný dvom. Skontrolujme, či sa každý kandidát z tabuľky C1 (vyššie) nachádza v databáze D minimálne 2 krát. Ak by tomu tak nebolo, musíme ten riadok vymazať. Keďže tomu tak je tak všetky riadky tabuľky C1 obsahujú **frekventované množiny** a spoj prvý stĺpec tabuľky C1 sám so sebou. Sprav C11 x C11 = C2. C2 bude vyzerať napr. takto:
Tabuľka C2
{I1,I2}
{I2,I3}
{I3,I4}
...
znovu prehľadaj **databázu** D, vyhľadaj v nej riadok po riadku z tabuľky C2 a zapamätaj si, koľko krát si každý riadok našiel. Výsledky početností priraď do stĺpca Support Count tabuľky C2.
Tabuľka C2

Frequent 2-Itemset	Support count
{I1,I2}	5
{I2,I3}	4
{I3,I4}	5
{I4,I5}	1

Skontroluj, či všetky riadky spĺňajú podmienku min_sup=2. Vidíme že jeden kandidát má početnosť nižšiu ako 2, takže z tabuľky C2 vypadne. Vytvor tabuľku L2 do ktorej vložíš záznamy z C2 ktoré tam po vymazaní posledného riadku z C2 zostali. Spoj prvý stĺpec tabuľky L2 sám so sebou, takže L21 x L21 = C3. Dostaneme množiny s troma prvkami. Prehľadaj **databázu** D a hľadaj v nej všetky riadky tabuľky C3. Pamätaj si pritom výsledky hľadania, početnosti výskytov riadkov z C3 budú priradené do nového stĺpca C3 (tabuľka C3 ešte bez pridaného nového stĺpca vyzerá takto):
Tabuľka C3
{I1,I2,I3}
{I2,I3,I1}
{I3,I4,I2}
...
po vyhľadaní jej riadkov v databáze D dostávame (hodnoty sú príkladom skutočných, reálny príklad je v 50. otázke/kapitole):

Tabuľka C3	Support Count
{I1,I2,I3}	2
{I2,I3,I1}	2
{I3,I4,I2}	2

...
Všetky riadky spĺňajú podmienku minimum support count = 2. Všetky riadky v tabuľke zostávajú. Spoj prvý stĺpec tabuľky C3 sám so sebou a dostaneš štvorprvkové množiny.
Tabuľka L4
{I1,I2,I3, I1}
{I2,I3,I1, I2}
{I3,I4,I2, I4}
...
znovu prehľadaj **databázu** D a vyhľadaj v nej riadky tabuľky L4. Po prehľadaní databázy D musí byť splnená podmienka:
„all subsets of a frequent itemset must also be frequent"
inak povedané, ak nastane prípad, že sa niektoré riadky tabuľky L4 v databáze D nenašli (čo budem aj predpokladať) priraď im support count = 0, vymaž ich z tabuľky L4 a výpočet algoritmu skonči. Definícia hovorí o **subsets of a frequent itemsets** pretože každý riadok z tabuľky L4 sa dá prepísať na množiny o veľkosti o jeden prvok menšie ako sú množiny v L4. Reálny príklad som vypočítal v 50. otázke.

Ako FP-growth zvyšuje efektívnosť Apriori algoritmu
Prečo vznikol FP-growth? Pretože ak máme v predošlom príklade ako vstup rozsiahlu databázu, Apriori generuje veľmi veľa kandidátov (riadkov pomocných tabuliek). Ďalšou nevýhodou Apriori je to, že neustále prehľadáva celú databázu D (a ešte k tomu aj rozsiahlu množinu kandidátov). Apriori algoritmus má vo svojej pôvodnej verzii nevýhodu v tom, že pri jeho spustení programom na veľkom datasete generuje príliš veľa dát/tabuliek/množín.
FP-growth nie je jedinou možnosťou ako zvýšiť jeho efektivitu, sú aj ďalšie:

- Hash-based technique, ktorá robí z n-produktových množín tzv. „buckets". Dá sa použiť iba pre množiny o počte prvkov **viac ako 1.** Dá sa povedať, že optimalizuje tak, že namiesto 1-itemsetov rovno vypočíta 2-itemsety pre ktoré si ukladá **bucket_address, bucket_count a bucket_contents. V bucket_count sú početnosti.**
- Transaction reduction, ktorá hovorí že ak sme pre konkrétny riadok databázy zistili, že táto transakcia neobsahuje žiadnu frekventovanú množinu, tak ju nebude už obsahovať vôbec pre ďalšie kroky a prestane s týmto riadkom počítať.
- Partitioning – databáza obchodu sa rozdelí a následne sa počíta v dvoch paralelných cykloch, resp. viacerých. V prvej fáze rozdelí databázu D do disjunktných častí. Ak rozdelí databázu na 2 časti, pre každú vypočíta početnosť (prvá_časť=>druhá_časť) = P(prvá **alebo** druhá) pre každý itemset v každej časti, túto početnosť porovná s minimálnou (zadanou). Táto metóda považuje itemset za frekventovaný, ak sa nachádza aspoň v 1 časti databázy D.
- Sampling – algoritmus počíta len nad podmnožinou zo skutočného datasetu obchodu
- Dynamic itemset counting

Princíp FP-growth alebo finding frequent itemsets without candidate generation
Teorka hovorí, že prvý krok FP-growth je rovnaký ako pri Apriori, máme teda:
Tabuľka C1

Frequent 1-Itemset	Support count
{I1}	7 #7 krát sa nachádza v databáze D
{I2}	6
{I3}	8 #8 krát sa nachádza v databáze D, vidno že C1 nie je zoradená
{I4}	2

pri minimum support count rovnej 2. Zoradí riadky podľa početnosti, takže vznikne tabuľka:
Tabuľka C1x (sort prijíma arg. descending)

Frequent 1-Itemset	Support count
{I3}	8 #8 krát sa nachádza v databáze D
{I1}	7 #7 krát sa nachádza v databáze D
{I2}	6
{I4}	2

Vytvorí FP-tree:
<null> #toto je jeho koreň
Prehľadá databázu D a keďže má k dispozícii len jednoprvkové množiny hľadá tie. Ak by bol prvý riadok databázy D takýto:
Databáza D

Transakcia TID	Items
T100	I1, I2, I3 #nie sú v množine!

zistí, že riadok obsahuje D obsahuje 3 rovnaké produkty ako Tabuľka C1x. Tá má riadky zoradené a preto bude podľa tohto poradia vytvárať vetvy. Prvá vetva pod koreňom stromu bude:
<I3:1><I1:1><I2:1> #ano je to správne, vynechal I4 lebo nie je v prvom riadku D!
Prvý uzol zľava <I3:1> vybral preto, lebo sa pozerá na stĺpec Support Count tabuľky C1x. Najvyššiu absolútnu početnosť má {I3}. Druhý uzol zľava je tam preto, lebo menšiu početnosť od predošlej má len {I1}. Tretí uzol vybral preto isté. Nevybral uzol {I4} pretože o ňom nevie, zaujíma ho databáza D. **Zatiaľ prečítal len jeden riadok databázy D, v ňom {I4} nenašiel, nevie o ňom.**
Uzol <I3:1> je dcérsky uzol koreňa, <I1:1> zase dcérsky uzol <I3:1> atď.
Tento postup sa zopakuje na ďalšom riadku databázy D avšak takto:
Databáza D

Transakcia TID	Items
T200	I4, I2 #nie sú v množine!

Vznikne nová vetva <I2:1><I4:1> pretože si ešte v žiadnom kroku neprepočítal absolútne početnosti v tabuľke vyššie. Takže produkt I2 sa nachádza s väčšou početnosťou ako I4, preto bude prvým uzlom v novej vetve. Produkt I4 má od neho jedinú nižšiu početnosť. O ostatných nevie.
Vznikne teda nová vetva <I2:1><I4:1> kde bude prvý uzol patriť koreňu a druhý prvému len s tým rozdielom, že dostane prefix ktorým sa napojí na vetvu z T100. Tie jednotky ktoré sú za názvami produktov sú **suffixy** a názvy produktov **prefixy** uzlov. Jednotka je tam vždy, zvyšuje sa vždy o ďalšiu jednotku len vtedy, ak by nastal prípad (v našom príklade) že dve vetvy by začínali tým istým uzlom. Druhá vetva by potom v prvom uzle mala dvojku <I2:2>

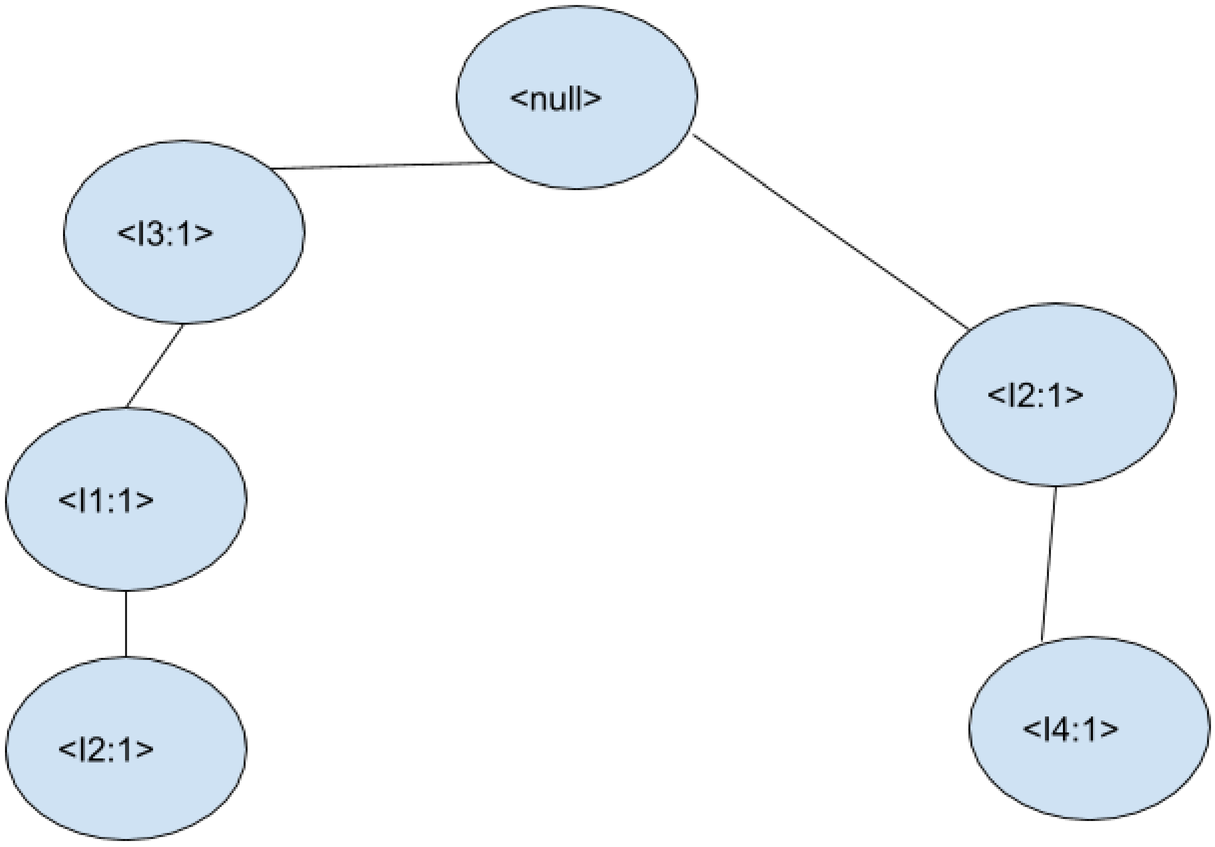
Algoritmus FP-growth je jedinečný tým, že vytvára pomocnú tabuľku pre transakcie z databázy D. Ku každému riadku tejto tabuľky priradí zodpovedajúce uzly z vytvoreného stromu.

The FP-growth method transforms the problem of finding long frequent patterns into searching for shorter ones in much smaller conditional databases recursively and then concatenating the suffix. It uses the least frequent items as a suffix, offering good selectivity. The method substantially reduces the search costs.

Ako hľadá FP-growth vzory:
Zoberie tabuľku C1x:
Tabuľka C1x

Frequent 1-Itemset	Support count
{I3}	8 #8 krát sa nachádza v databáze D
{I1}	7 #7 krát sa nachádza v databáze D
{I2}	6
{I4}	2

Berie riadky odspodu. Produkt {I4} sa nachádza v jednej vetve FP-tree. Nájde cesty k výskytu {I4}. Táto cesta je {<null>,<I2:1>,<I4:1>} . Posledným uzlom pred <I4:1> sú {<null>,<I2:1>}. Množina {<null>,<I2:1>} je **conditional pattern base**. Strom vyzerá takto:



Z tejto conditional pattern base vytvorí databázu, ďalej vytvorí nový strom zvaný I4-conditional FP-tree. Produkt {I4} našiel len v jednej vetve, tento produkt má početnosť rovnú 2 (v tabuľke C1x). Takže strom I4-conditional FP-tree má / bude mať len jednu vetvu {<null>,<I2:1>}. Z tejto vetvy vygeneruje všetky možné frekventované vzory, tými sú:

- {<null>,<I4:2>}
- {<I2:2>, <I4:2>}

Tieto dve množiny (vyššie) sú frekventované vzory alebo "frequent patterns".

7. Získanie asociačných pravidiel z frekventovanej množiny

Once the frequent itemsets from transactions in a database D have been found, it is straightforward to generate strong association rules from them (where strong association rules satisfy both minimum support and minimum confidence). This can be done using following equation for confidence of some rule, which I show again here for completeness:

Confidence(A=>B)=P(B|A) = support_count(A or B) / support_count(A)

The conditional probability P(B|A) is expressed in terms of itemset support count, where support_count(A or B) is the number of transactions containing the itemsets A or B,support count(A) is the number of transactions containing the itemset A. Based on this equation, association rules can be generated as follows:

- for each frequent itemset ***l*** generate all nonempty subsets of ***l*** (of them)
- for every nonempty subset of ***l*** , output the rule „s => (***l*** -s)" if (support_count(***l***) / support_count(s)) >= min_conf where min_conf is the minimum confidence threshold.

Because the rules are generated from frequent itemsets, each one automatically satisfies the minimum support. Frequent itemsets can be stored ahead of time in hash tables along with their counts so that they can be accessed quickly.

Príklad na výpočet asociačných pravidiel z frekventovanej množiny
The data contain frequent itemset X = {I1, I2, I5}. What are the association rules that can be generated from X?

For each frequent itemset which is only one in our example and equal to X = {I1, I2, I5} generate all nonempty subsets of ***l***. These nonempty subsets are:
{I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2} and {I5}.
For every nonempty subset **s** of ***l*** , output the rule „**s** => (***l*** - **s**)" if (support_count(***l***) / support_count(s)) >= min_conf where min_conf is the minimum confidence threshold stated as 70 % by a domain knowledge engineer.

We’re going to compute following rules:
{I1, I2} => I5 #na pravú stranu sa pýtate, ktorý produkt chýba na ľavej strane a je v X?
{I1, I5} => I2
{I2, I5} => I1
{I1} => {I2, I5} #pretože toto pravidlo a {I2, I5} => I1 nie je to isté
{I2} => {I1, I5}
{I5} => {I2, I1}

{I1, I2} => I5 has confidence 2/4 = 0,5 * 100 = 50 % because
s = {I1, I2} and ***l*** = {I1, I2, I5} so (***l***-s) = {I5} has support count(***l***) equal to 2 and support_count(s) is equal to 4. These numbers have to be previously calculated and are always picked up from frequency tables:

Some frequent 3-itemset	Its support count
{I1,I2,I5}	2

{I1, I5} => I2 has confidence 2/2 = 1 * 100 = 100 % because
s = {I1, I5} and ***l*** = {I1, I2, I5} so (***l***-s) = {I2} has support count(***l***) equal to 2 and support_count(s) is equal to 2. These numbers have to be previously calculated and are always picked up from frequency tables:

Some frequent 3-itemset	Its support count
{I1,I2,I5}	2
Some frequent 2-itemset	Its support count
{I1,I5}	2

{I2, I5} => I1 has confidence 2/2 = 1 * 100 = 100 % because
s = {I2, I5} and ***l*** = {I1, I2, I5} so (***l***-s) = {I1} has support count(***l***) equal to 2 and support_count(s) is equal to 2. These numbers have to be previously calculated and are always picked up from frequency tables.

Some frequent 3-itemset	Its support count
{I1,I2,I5}	2
Some frequent 2-itemset	Its support count
{I2,I5}	2

{I1} => {I2, I5} has confidence 2/6 = 0,3333 * 100 = 33,33 % because
s = {I1} and ***l*** = {I1, I2, I5} so (***l***-s) = {I2, I5} has support count(***l***) equal to 2 and support_count(s) is equal to 6. These numbers have to be previously calculated and are always picked up from frequency tables.

Some frequent 3-itemset	Its support count
{I1,I2,I5}	2
Some frequent 2-itemset	Its support count
{I1}	6

{I2} => {I1, I5} has confidence 2/7 = 0,2857 * 100 = 29 % because
s = {I2} and ***l*** = {I1, I2, I5} so (***l***-s) = {I1,I5} has support count(***l***) equal to 2 and support_count(s) is equal to 7. These numbers have to be previously calculated and are always picked up from frequency tables.

{I5} => {I1, I2} has confidence 2/2 = 1 * 100 = 100 % because
s = {I5} and ***l*** = {I1, I2, I5} so (***l***-s) = {I1,I5} has support count(***l***) equal to 2 and support_count(s) is equal to 2. These numbers have to be previously calculated and are always picked up from frequency tables.

If the minimum confidence threshold is say, 70 %, then only the second, third and last rules are output , because these are the only ones generated that are strong.

8. Klasifikácia - Bayes, ako súvisí s podmienenou pravdepodobnosťou a ako funguje

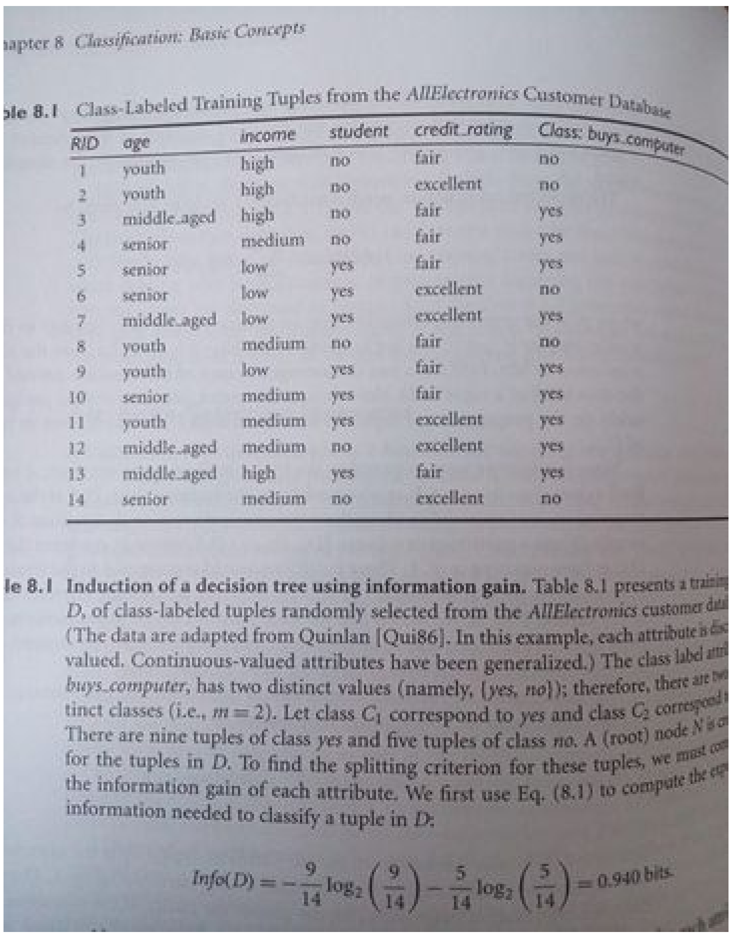
Bayes Classification Methods
Bayesian classifiers are statistical classifiers. They can predict class membership probabilities (prekladaj ako pravdepodobnosť príslušnosti znaku k určitej triede) such as the probability that a given tuple (tuple je iný názov pre observation, row,...) belongs to a particular class. Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class-conditional independence.

Bayes Theorem
Let X be a data tuple (niaky riadok). In Bayesian terms this data tuple is called evidence. This data tuple has n attributes/columns/features. Let H be some hypothesis such as that the data tuple X belongs to a specified class C. This means, we want to determine probability P(Hypothesis|data_tuple_X) which means we are looking for the probability that tuple X belongs to class C, given that we know the attribute description of X.

P(Hypothesis|data_tuple_X) is the posterior probability of H conditioned on X. For example, suppose our data tuples are confined to customers described by the attributes **age** and **income**, and that X is a 35-year-old customer with an income of $40000. Suppose that H is the hypothesis that our customer will buy a computer. Then P(H|X) where X is a data tuple and H stands for hypothesis reflects the probability that customer X will buy a computer given that we know the customer age's and income. Knowing the customer's age and income is a conditional part of our probability. Predicting that he will buy a computer from our shop is our hypothesis.

In contrast, P(H) is the prior probability of hypothesis H. For our example, this is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter. It's because we do not have conditional part X in our probability so we do not specify customer, which would buy a computer from our shop.

Example



ble 8.1 Class-Labeled Training Tuples from the *AllElectronics* Customer Database

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

le 8.1 **Induction of a decision tree using information gain.** Table 8.1 presents a trainin
D, of class-labeled tuples randomly selected from the *AllElectronics* customer dat
(The data are adapted from Quinlan [Qui86]. In this example, each attribute is dis
valued. Continuous-valued attributes have been generalized.) The class label attri
buys_computer, has two distinct values (namely, {*yes*, *no*}); therefore, there are tw
tinct classes (i.e., $m = 2$). Let class C_1 correspond to *yes* and class C_2 correspond t
There are nine tuples of class *yes* and five tuples of class *no*. A (root) node *N* is cr
for the tuples in *D*. To find the splitting criterion for these tuples, we must com
the information gain of each attribute. We first use Eq. (8.1) to compute the exp
information needed to classify a tuple in *D*:

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

We wish to predict a **customer decision** from a dataset tuples using naive Bayesian classification given the training data. The data tuples are described by the attributes **age, income, student, credit_rating.** Customer decision attribute **buys_computer** has two distinct values: **yes** or **no.** Let C1 correspond to the class **buys_computer = yes** and C2 correspond to **buys_computer = no.** The tuple from the training data set we wish to classify is:

X = (age = youth**, income =** medium**, student =** yes**, credit_rating =** fair**)**

We need to maximize **a posterior probability for 2 classes so:**
P(X_tuple|Class_i)*P(Class_i) for i = 1,2. We want to maximize numerator of the following formula:
Conditional probability of data tuple X which says that X will belong to class C is computed by:
P(Class|X) = (P(X_tuple|Class_i)*P(Class_i))/(P(X_tuple))

For its computing we need to compute **a priori** probability **P(Class_i)** firstly (with no conditional part) of each class (we have classes C1 and C2). This probability is a part of a numerator stated above:
P(Class1) = P(buys_computer=yes) = 9yes/14rows
P(Class2) = P(buys_computer=no) = 5no/14rows

For computing another part of our numerator **P(X_tuple|Class_1)** we compute the following conditional probabilites. These probabilites are conditional, so they are a posterior:

P(age = youth | buys_computer = yes) = 2rows/9yes = 0.222
P(age = youth | buys_computer = no) = 3/5no = 0.600
P(income = medium | buys_computer = yes) = 4rows/9yes = 0.444
P(income = medium | buys_computer = no) = 2rows/5no = 0.400
P(student = yes | buys_computer = yes) = 6rows/9yes = 0.667
P(student = yes | buys_computer = no) = 1row/5no = 0.200
P(credit_rating = fair | buys_computer = yes) = 6rows/9yes = 0.667
P(credit_rating = fair | buys_computer = no) = 2/5no

Pravdepodobnosť, že riadku tabuľky bude priradená hodnota **yes** k triede **buys_computer** je rovná násobku pravdepodobností v ktorých vystupuje podmienka **buys_computer = yes**, teda:
0.222 * 0.444 * 0.667*0.667 = 0.044

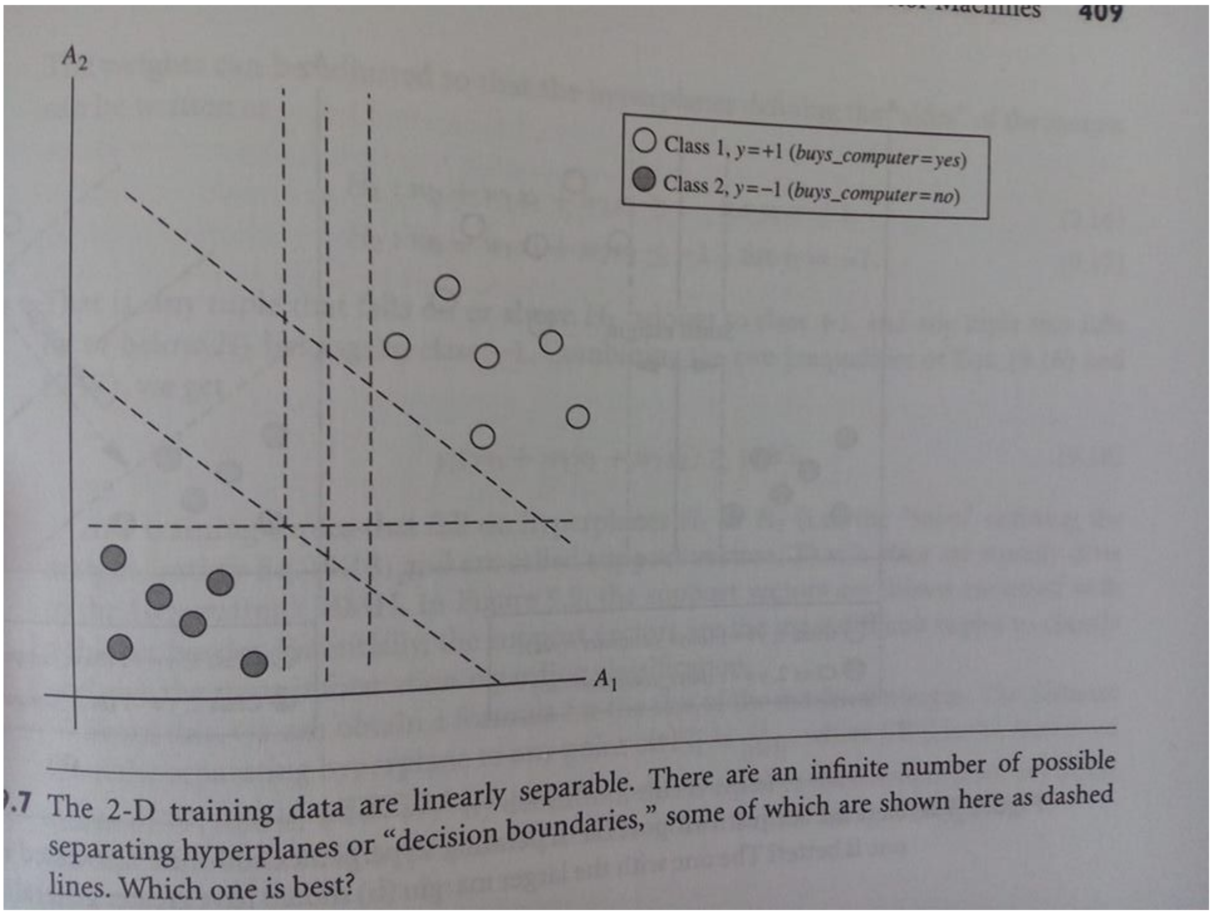
V prípade, že by nám v niakej pravdepodobnosti vyšla nula použijeme Laplaciovu korekciu (“Laplacian correction or Laplace estimator”).
Predpokladajme, že:
P(income = medium | buys_computer = no) = 0
z nuly spravíme jednotku, ale túto jednotku musíme pridať aj k
P(income = low | buys_computer = no)
P(income = high | buys_computer = no)

takže pri predpoklade, že pôvodné hodnoty sú takéto:
P(income = low | buys_computer = no) = 7/14
P(income = high | buys_computer = no) = 7/14
budú z nich:
P(income = low | buys_computer = no) = 8/17
P(income = high | buys_computer = no) = 8/17
P(income = medium | buys_computer = no) = 1/17

9. Popísať SVM

SVM je skratka Support Vector Machines:
SVM is a method for the classification of both linear and nonlinear data.
In a nutshell, an SVM is an algorithm that works as follows. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (hyperplane je anglický názov pre nadrovinu) so it separates the tuples of one class from another with this hyperplane. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM finds this hyperplane using support vectors (support vectors are some training tuples) and by margins (margins are defined by a support vectors).
How SVM works?
To explain SVM, let’s first look at the simplest case – a two-class problem where the classes are linearly separable. Let the data set D be given as (X1, Y1),(X2,Y2),...,(Xd,Yd) where d must be |d| and where Xi is the set of training tuples with associated classes Yi.

Table of Linearly Separable Data	
Tuples X	Class Labels Y
X1	Y1 = +1 **or buys_computer = yes**
X2	Y2 = -1 **or buys_computer = no**


Machines 409
A_2
A_1
Class 1, y=+1 (buys_computer=yes)
Class 2, y=−1 (buys_computer=no)

.7 The 2-D training data are linearly separable. There are an infinite number of possible separating hyperplanes or "decision boundaries," some of which are shown here as dashed lines. Which one is best?

In this method we want to find the best linear line that will separate values from our training data. We want to do this because we hope we will get the minimum classification error on previously unseen tuples (in a testing dataset). An SVM approaches this problem by searching for the **maximum marginal hyperplane.** Let's look at the following photo I provide:

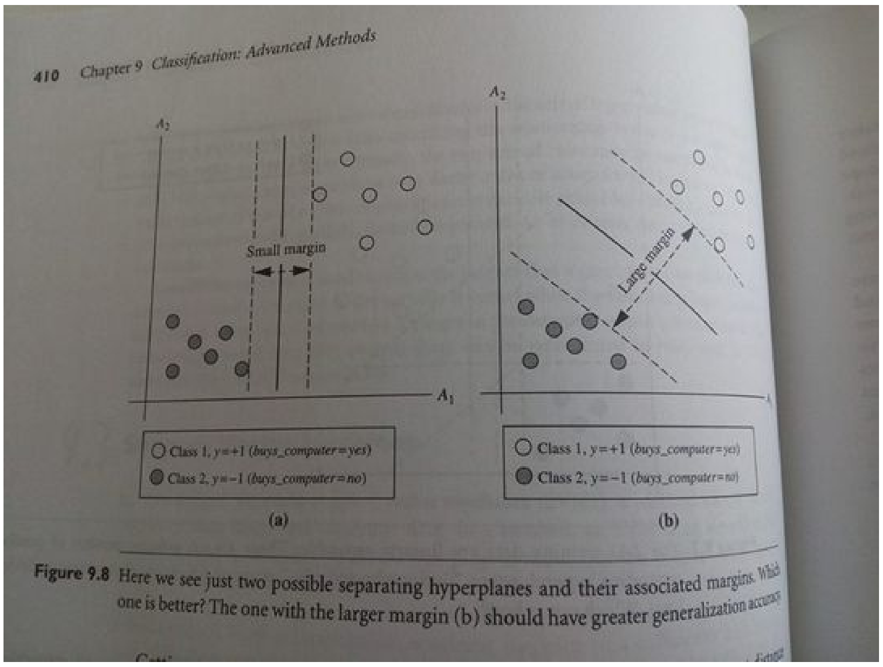

410 Chapter 9 Classification: Advanced Methods
A_2
Small margin
A_1
Class 1, y=+1 (buys_computer=yes)
Class 2, y=−1 (buys_computer=no)
(a)
A_2
Large margin
Class 1, y=+1 (buys_computer=yes)
Class 2, y=−1 (buys_computer=no)
(b)

Figure 9.8 Here we see just two possible separating hyperplanes and their associated margins. Whic one is better? The one with the larger margin (b) should have greater generalization accura

Picture shows two possible separating hyperplanes and their associated margins. Both hyperplanes can correctly classify all the given data tuples. Intuitively, however, we expect the hyperplane with the larger margin to be more accurate at classifying future data tuples than the hyperplane with the smaller margin. This is why the SVM searches for the hyperplane with the largest margin, that is, the **maximum marginal hyperplane.** The associated margin gives the largest separation between classes. We can say that the shortest distance from a hyperplane to one side of its margin is equal to the shortest distance from the hyperplane to the other side of its margin, where the sides of the margin are parallel to the hyperplane. When dealing with the **maximum marginal hyperplane,** this distance is, in fact, the shortest distance from the **maximum marginal hyperplane** to the closest training tuple of either class.

Example

Table of Linearly Separable Data		
Z	X	Class Labels Y
Z1 = 1	X1 = 2	Y1 = +1 **or buys_computer = yes**
Z2 = 3	X2 = 4	Y2 = -1 **or buys_computer = no**

Separating hyperplane of the table above is:
W*X + b = 0
W={attribute_Z_weight,attribute_X_weight}
W*X+b=additional_weight + attribute_Z_weight*its_row_values + attribute_X_weight*its_row_values = 0
Every point above our hyperplane
additional_weight + attribute_Z_weight*row_values + attribute_X_weight*row_values > 0
Every point below our hyperplane
additional_weight + attribute_Z_weight*row_values + attribute_X_weight*row_values < 0
Classification

- any tuple that falls on or above

additional_weight + attribute_Z_weight*row_values + attribute_X_weight*row_values >= 1 has class Y1 or **buys_computer = yes**

- any tuple that falls on or below

additional_weight + attribute_Z_weight*row_values + attribute_X_weight*row_values <= -1 belongs to class Y2 or **buys_computer=no**
***** Hyperplane**
0,1 + 1*1 + 1*2 = f(x) #plus b = 0,1, Z_weight=1, X_weight=1
first row belongs to class Y1 because 0,1 + 1*1 + 1*2 equals 3,1 > 1
second row belongs to class Y1 because 0,1 + 1*3 + 1*4 equals 7,1 > 1
***

10. Zhlukovanie, vstupy, výstupy, vlastnosti

Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a clustering. Different clustering methods may generate different clusterings on the same data set. Because a cluster is a collection of data objects that are similar to one another within the cluster and dissimilar to objects in other clusters, a cluster of data objects can be treated as an implicit class. In this sense, clustering is sometimes called **automatic classification.** Clustering is also called **data segmentation** in some applications because clustering partitions large data sets into groups according to their similarity. Clustering can also be used for **outlier detection,** where outliers may be more interesting than common cases. Clustering is known as **unsupervised learning** because the class label information is not present. For this reason, clustering is a form of **learning by observation.**
These are the requirements for clustering:
Scalability – many clustering algorithms work well on small data sets containing fewer than several hundred data objects. Clustering on only a sample of a given large data set may lead to biased results.
Ability to deal with different types of attributes – many algorithms are designed to cluster numeric (interval-based) data. However, applications may require clustering other data types, such as binary, nominal (categorical), and ordinal data, or mixtures of these data types.
Discovery of clusters with arbitrary shape – many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures. Algorithms based on such distance measures tend to find spherical clusters with similar size and density.
Requirements for domain knowledge to determine input parameters – many clustering algorithms require users to provide domain knowledge in the form of input parameters such as the desired number of clusters.
Ability to deal with noisy data – most real-world data sets contain outliers and/or missing, unknown, or erroneous data.
Incremental clustering and insensitivity to input order – in many applications, incremental updates may arrive at any time. Some clustering algorithms cannot incorporate incremental updates (with its own data) into existing clustering structures, and, instead, have to recompute a new clustering from scratch.
... more

11. Popisat zhlukovanie zalozene na rozdelovani, hierarchicke, na hustote a na modeloch

Zhlukovanie založené na rozdeľovaní – Partitioning Clustering Methods
Given a set of **N** objects, a partitioning method constructs **K** partitions of the data, where each partition represents a cluster and **K <= N** (this is the extreme case in which we would have N clusters). That is, it divides the data into K groups such that each group must contain at least one object. In other words, partitioning methods conduct one-level partitioning on data sets. The basic partitioning methods typically adopt exclusive cluster separation. That is, each object must belong to exactly one group. Most partitioning methods are distance-based. Given desired number of clusters **K** to construct, a partitioning method 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.
Partitioning Clustering Methods:
- k-Means: a centroid-based partitioning technique, this algorithm is not guaranteed to converge to the global optimum and often terminates at a local optimum. The results from this algorithm could depend on the initial random selection of cluster centers
- k-Medoids: A representative Object-Based Technique, major drawback of this algorithm is sensitivity to outliers

Hierarchické zhlukovanie – Hierarchical Clustering Methods
A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either **agglomerative** or **divisive,** based on how the hierarchical decomposition is formed. The agglomerative approach which is also called bottom-up approach, starts with each object forming a separate group. It sucessively merges the objects or groups close to one another, until all the groups are merged into one. The divisive approach, also called the top-down approach, starts with all the objects in the same cluster. In each successive iteration, a cluster is split into smaller clusters, until eventually each object is in one cluster, or a termination condition holds/happens. Hierarchical clustering methods can be distance-based or density- and continuity-based.
Hierarchical methods are:
- BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees
- Chameleon: Multiphase Hierarchical Clustering Using Dynamic Modeling
- Probabilistic Hierarchical Clustering

Zhlukovanie založené na hustote – Density-based Clustering Methods
Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty in discovering clusters of arbitrary shapes. Density-based methods can divide a set of objects into multiple

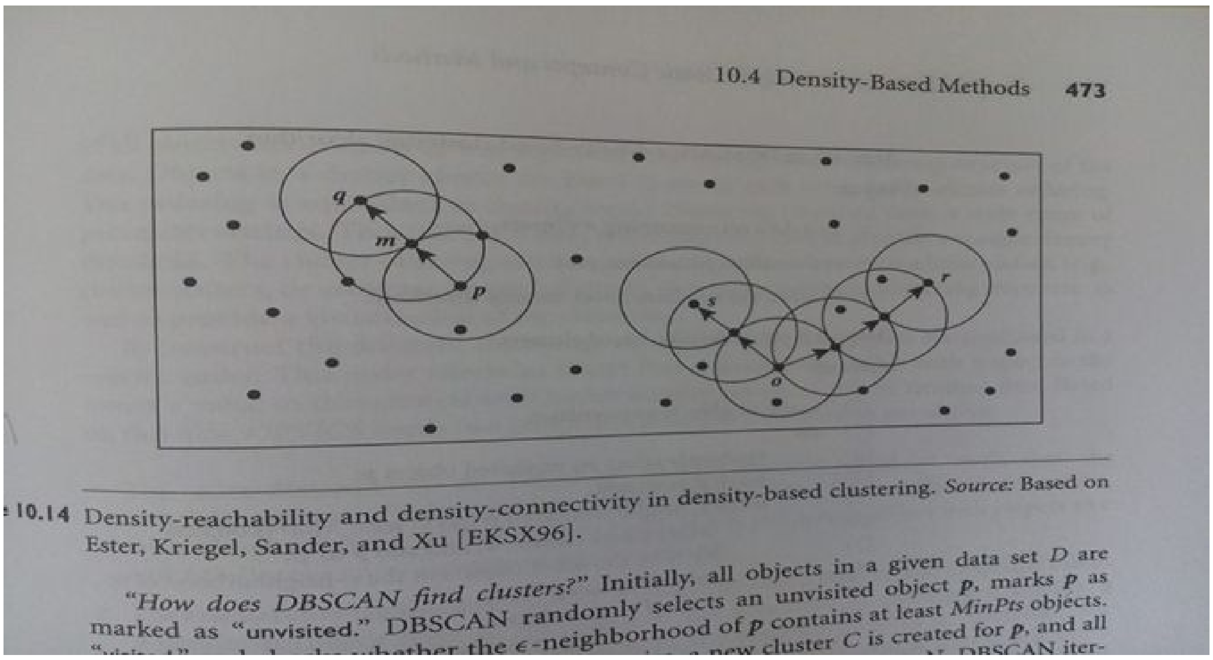
exclusive clusters, or a hierarchy of clusters. Typically, density-based clustering methods consider exclusive clusters only, and do not consider fuzzy clusters.
Density-based algorithms are:
-DBSCAN: Density-Based Clustering Based on Connected Regions with High Density
-OPTICS: Ordering Points to Identifiy the Clustering Structure
-DENCLUE: Clustering Based on Density Distribution Functions

Zhlukovanie založené na modeloch – Grid-based methods (nevolá sa tak, ako si pamätáte otázku)
Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All the clustering operations are performed on the grid structure. The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the grid structure. Using grids is often an efficient approach to many spatial data mining problems including clustering.
Grid-Based methods are:
-STING: Statistical Information Grid
-CLIQUE: An Apriori-like Subspace Clustering Method

12. Vybrat si jeden alg. zalozeny na hustote a popisat

Partitioning and hierarchical methods are designed to find spherical-shaped clusters. They have difficulty finding clusters of arbitrary shape such as the „S“ shape and oval clusters.

DBSCAN: Density-Based Clustering Based on Connected Regions with High Density
The density of an object **o** can be measured by the number of objects close to **o**. DBSCAN find core objects, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters. So how does DBSCAN quantify the neighborhood of an object? A user-specified parameter E > 0 is used to specify the radius of a neighborhood we consider for every object. The E-neighborhood of an object **o** is the space within a radius E centered at **o**. Due to the fixed neighborhood size parameterized by E, the density of a neighborhood can be measured simply by the number of objects in the neighborhood. To determine whether a neighborhood is dense or not, DBSCAN uses another user-specified parameter **MinPts**, which specifies the density threshold of dense regions. An object is a **core object** if the E-neighborhood of the object contains at least **MinPts** objects. Core objects are the pillars of dense regions. Given a set D of objects we can identify all core objects with respect to the given parameters E and MinPts. The clustering task is therein reduced to using core objects and their neighborhoods to form dense regions, where the dense regions are clusters. For a core object **q** and an object **p**, we say that **p** is directly density-reachable from **q** if **p** is within the E-neighborhood of **q.**
It looks like this:



10.14 Density-reachability and density-connectivity in density-based clustering. *Source:* Based on Ester, Kriegel, Sander, and Xu [EKSX96].

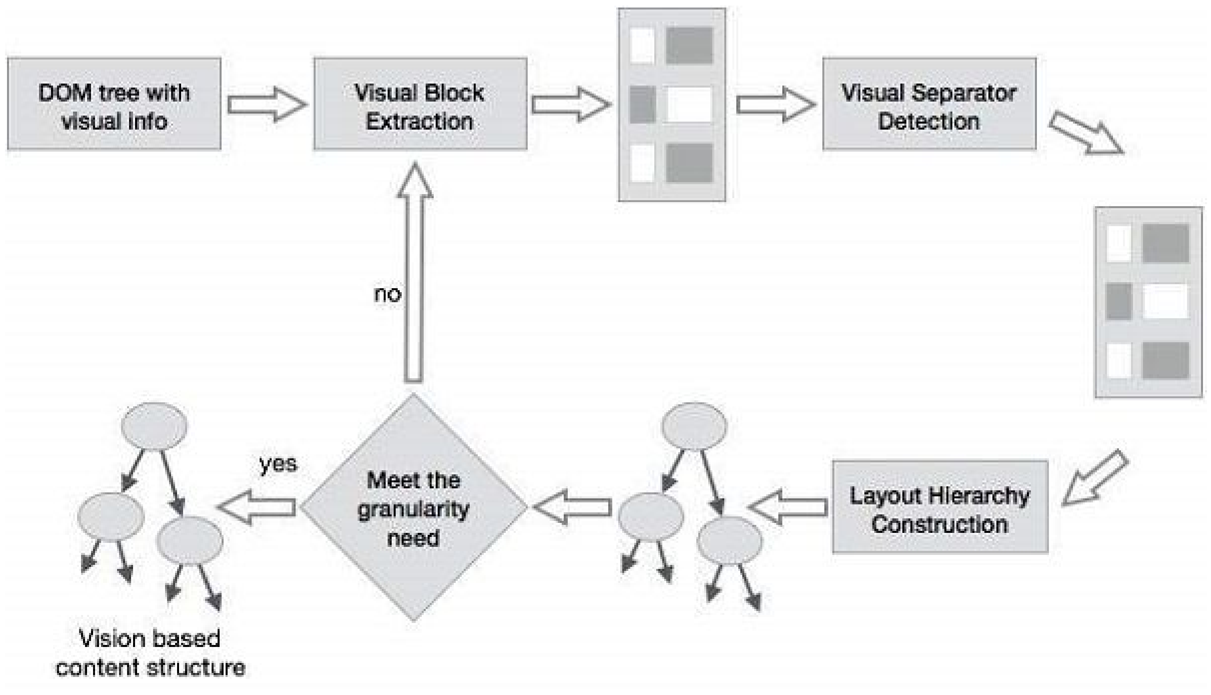
"How does DBSCAN find clusters?" Initially, all objects in a given data set D are marked as "**unvisited.**" DBSCAN randomly selects an unvisited object p, marks p as "visited," and checks whether the ϵ-neighborhood of p contains at least *MinPts* objects. new cluster C is created for p, and all DBSCAN iter-

13. Dolovanie na webe - aké informácie dolujeme?

Web content mining analyzes web content such as text, multimedia data, and structured data (within web pages or linked across web pages). This is done to understand the content of web pages, provide scalable and informative keyword-based page indexing, entity/concept resolution, web page relevance and ranking, web page content summaries, and other valuable information related to web search and analysis. Web pages reside either on the surface web on on the deep Web. The surface web is that portion of Web that is indexed by typical search engines. The deep Web (or hidden Web) refers to web content that is not part of the surface web. Its contents are provided by underlying database engines.
Web structure mining is the process of using graph and network mining theory and methods to analyze the nodes and connection structures on the Web. It extracts patterns from hyperlinks, where a hyperlink is a structural component that connects a web page to another location.
Web usage mining is the process of extracting useful information from server logs. It finds patterns, trends, related to general or particular groups of users. This kind of mining understands users‘ search patterns, trends and associations. It predicts what users are looking for on the Internet.

14. Popísať VIPS, prečo je dobrá segmentácia stránky

Vision Based Page Segmentation extracts the semantic structure of a web page based on its visual presentation. Such a semantic structure corresponds to a tree structure. In this tree each node corresponds to a block. A value is assigned to each node. This value is called the Degree of Coherence. This value is assigned to indicate the coherent content in the block based on visual perception. The VIPS algorithm first extracts all the suitable blocks from HTML DOM tree. After that it finds the separators between these blocks. The separators refer to the horizontal and vertical lines in a web page that visually cross with no blocks. The semantics of a web page is constructed on the basis of these blocks.


DOM tree with visual info
Visual Block Extraction
Visual Separator Detection
no
yes
Meet the granularity need
Layout Hierarchy Construction
Vision based content structure

15. Napísať dve najtypickejšie úlohy pri dolovaní obsahu

Pri dolovani obsahu textových dát či čoho?
Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis (to je keď z komentárov zisťujete správanie používateľa, jeho hnev a pod.), document summarization, and entity-relation modeling.

16. Porovnať dolovanie na www stránke a v texte

Text mining is an interdisciplinary field that draws on information retrieval, data mining, machine learning, statistics, and computational linguistics.
Web mining is the application of data mining techniques to discover patterns, structures, and knowledge from the Web.

17. Metriky stredných hodnôt, 3 opísať, výhody nevýhody, typy atribútov, na ktoré sa používajú // je to v doku arit. priemer, modus, median

Typy atribútov

- Nominal attributes: the values of a nominal attribute are symbols or names of things. They are also referred as categorical. These values do not have any meaningful order.
- Binary attributes: are nominal attributes with only two categories or states: 0 or 1. Zero typically means that the attribute is absent, and 1 means that it is present.
- Symmetric Binary Attributes have both of its states equally valuable and they carry the same weight. One such example could be the attribute **gender** having the states male and female.
- Asymmetric Binary Attributes: do not have equally important values. One such example is the positive and negative outcomes of a medical test for HIV.
- Ordinal Attributes: are attributes with possible values that have a meaningful order or ranking among them, but the magnitude between successive values is not known. One such examples are: school grade, professional rank...
- Numeric Attributes: are quantitative, that is, they are measurable quantified, represented in integer or real values.
- Interval-scaled Numeric Attributes: for example temperature, or dates
- Ratio-scaled Numeric Attributes: are numeric attributes with an inherent zero-point. That is, if a measurement is ratio-scaled, we can speak of a value as being a multiple (or ratio) of another value. Difference is, that we cannot say with interval-scaled attribute such as date, that 2018 is a year/value twice as 1009. We cannot divide years 2018/2=1009
- Discrete attributes: have a finite or countably infinite set of values, which may or may not be represented as integers.

*** Kto mal štatistiku práve zistil, že táto definícia nesedí s diskrétnymi hodnotami v štatistike. Tam sú diskrétne hodnoty reálne čísla, lebo sú na reálnej osi. Tu byť nemusia!***

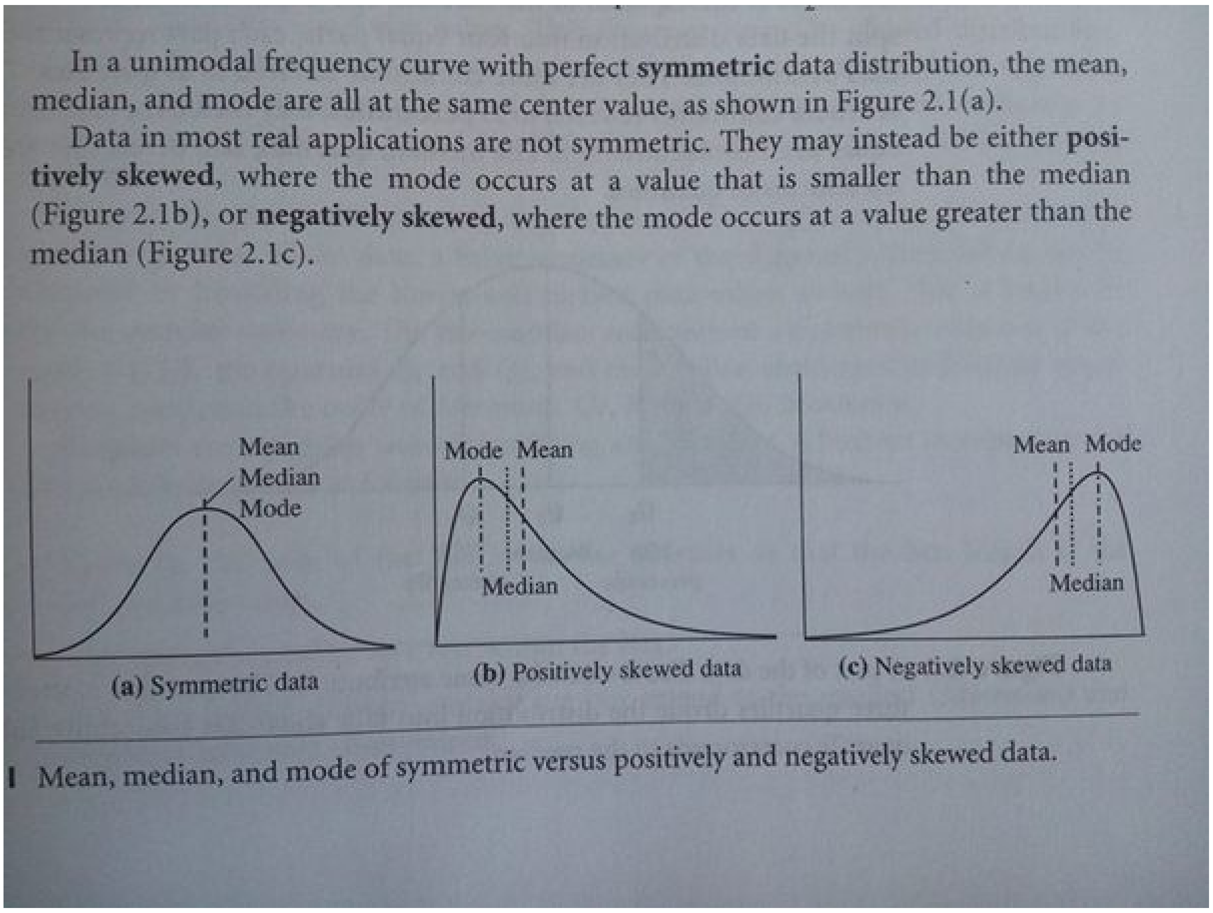
- Continuous attributes: are those, which are not discrete

18. Metriky rozptylu, výhody a nevýhody // rozpytl , rozsah, kvartily, odlahle hodnoty, krabicove diagramy

Mean
Is the most common and effective numeric measure of the center of a set of data. Let x1, x2, ...,xn be a set of N values or observations for a given attribute like salary (for example). The mean of this set of values is:
Sum(xi)/N

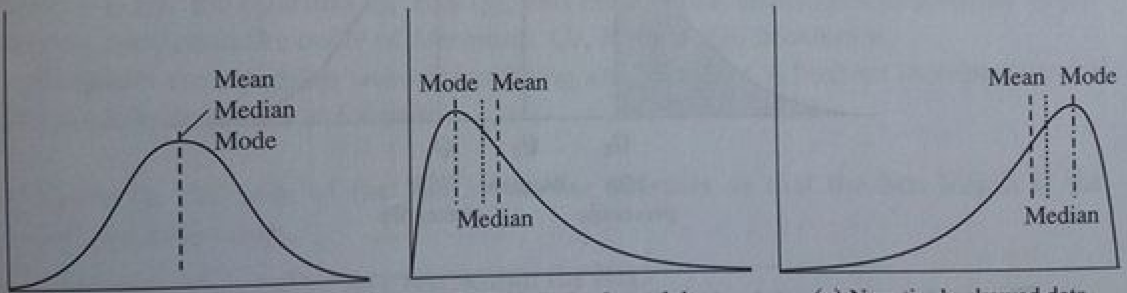
Median
For skewed (asymmetric) data a better measure of the center of data is the median, which is the middle value in a set of ordered data values. Suppose that a given data set of N values for an attribute X is sorted in increasing order. If N is odd (nepárne), then the median is the middle value of the ordered set. If N is even, then the median is not unique, it is the two middlemost values and any value in between. So if we would have attribute of 12 observations (even number), median is a mean of a two middlemost values (súčet 6. a siedmej hodnoty predelený dvoma, lebo sú dve hodnoty v čitateli).
If we would have 11 observations for a one specific attribute, median would be sixth value.

Mode
The mode for a set of data is the value that occurs most frequently in the set. Datasets with one, two, or three modes are respectively called **unimodal, bimodal and trimodal.**



In a unimodal frequency curve with perfect **symmetric** data distribution, the mean, median, and mode are all at the same center value, as shown in Figure 2.1(a).

Data in most real applications are not symmetric. They may instead be either **positively skewed**, where the mode occurs at a value that is smaller than the median (Figure 2.1b), or **negatively skewed**, where the mode occurs at a value greater than the median (Figure 2.1c).



(a) Symmetric data (b) Positively skewed data (c) Negatively skewed data

I Mean, median, and mode of symmetric versus positively and negatively skewed data.

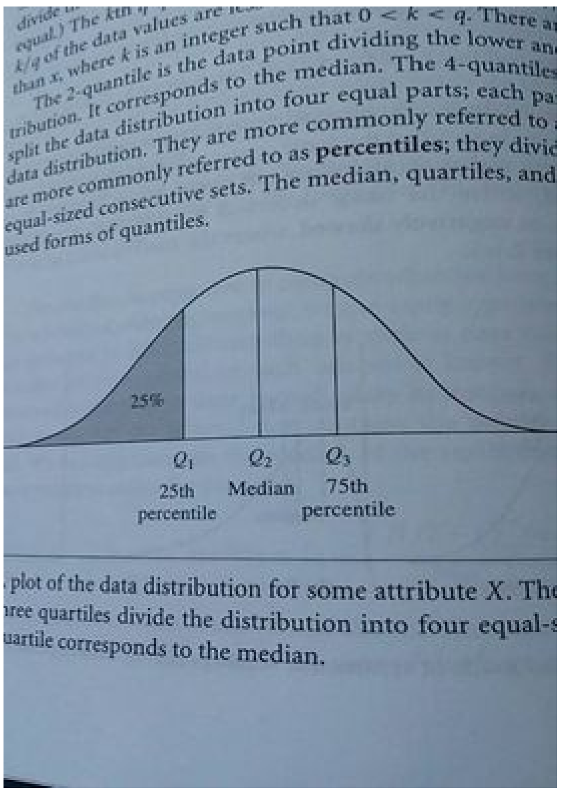
Metriky rozptylu

- **Range**

Let x1, x2, x3,...,xn be a set of observations for some numeric attribute X. The range of the set is the difference between the largest and smallest value. So:
Range <- max(attribute) – min(attribute)

- **Quantiles (Kvantily)**

Quantiles are points taken at regular intervals of a data distribution, dividing it into essentially equal-size consecutive sets.
*** Kvartil je kvantil, percentil je kvantil, kvartily majú svoje poradie, tretí kvartil je 75-ty percentil pretože osekáva rozdelenie dát o 75 % dát pod ním. Prvý kvartil je 25-ty percentil lebo rozdeľuje dáta na 25 % dát pod ním a na 75 % dát nad ním. Druhý kvartil je 50-ty percentil, lebo rozdeľuje dáta na dve polovice. 50-ty percentil je druhý kvartil a druhý kvartil je medián***



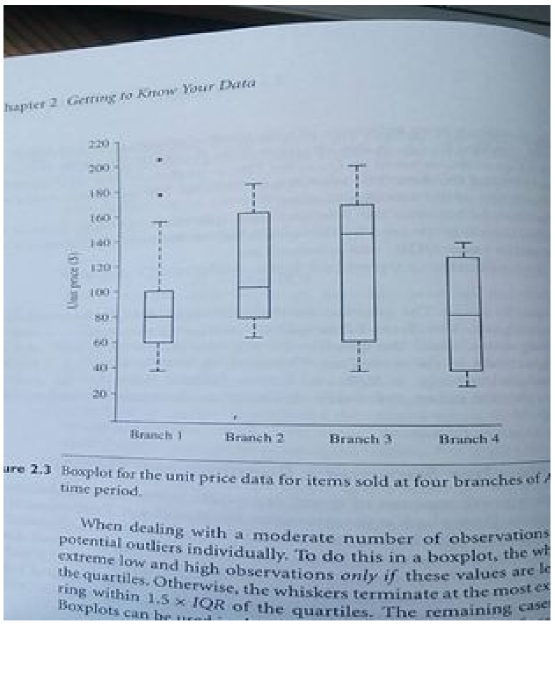
equal.) The kth q
k/q of the data values are le
than x, where k is an integer such that $0 < k < q$. There a
The 2-quantile is the data point dividing the lower an
tribution. It corresponds to the median. The 4-quantiles
split the data distribution into four equal parts; each pa
data distribution. They are more commonly referred to
are more commonly referred to as **percentiles**; they divid
equal-sized consecutive sets. The median, quartiles, and
used forms of quantiles.

plot of the data distribution for some attribute X. The
ree quartiles divide the distribution into four equal-
uartile corresponds to the median.

- **Interquartile Range (Medzikvartilová odchýlka)**

The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. Is defined as follows:
IQR= tretí kvartil – prvý kvartil
štvrtý kvartil sa neudáva lebo nerozdeľuje nič, jedine tak 100 % dát pod a 0 % dát nad ním.

- **Krabicový graf (Box-plot)**



hapter 2 *Getting to Know Your Data*

ure 2.3 Boxplot for the unit price data for items sold at four branches of
time period.

When dealing with a moderate number of observations
potential outliers individually. To do this in a boxplot, the wh
extreme low and high observations *only if* these values are le
the quartiles. Otherwise, the whiskers terminate at the most ex
ring within 1.5 × *IQR* of the quartiles. The remaining case
Boxplots can be

Výška tých obdĺžnikov sú medzikvartilové odchýlky, teda hore je tretí kvartil a dole je prvý kvartil. Čiara ktorá ich rozdeľuje je medián. Tie čiary ktoré z krabíc vychádzajú sú minimálna hodnota a maximálna hodnota. Teda Branch 1 má minimálnu hodnotu 40 a maximálnu skoro 160. Bodky mimo krabíc sú outliers, teda vychýlené hodnoty. Boxplot sa používa na fakt základnú analýzu dát. Vizualizuje tie najjednoduchšie metriky.

19. Zhlukovanie - podla rozdelovania (partition) výstižne opísať, uviesť jeden príklad na takúto metódu: opísať algoritmus, výhody, nevýhody

*** už bolo zodpovedané, pozri vyššie alebo Ctrl + F ***

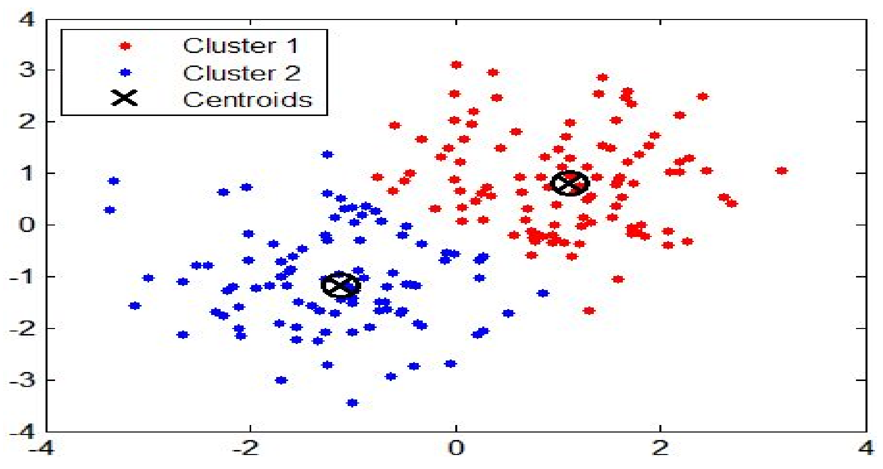
20. Podľa hustoty, to isté + porovnat s a. (teda podla hustoty vs. podla rozdelenia)

*** už bolo zodpovedané, pozri vyššie alebo Ctrl + F ***

21. Metriky porovnania zhlukovania, 1 opísať na čom je založená // aké sú tieto metriky prosím?

Metrika pre k-Means, algoritmus patriaci do Partitioning Methods / Metódy zhlukovania

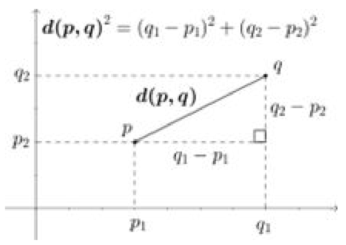
A centroid-based partitioning technique uses the centroid of a cluster C to represent that cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean or medoid of the objects assigned to the cluster.



The difference between an object **p** which is a part of a cluster C and object **c** (**c** is a representative of a cluster) is measured by a function which takes two parameters:

- object p
- object c

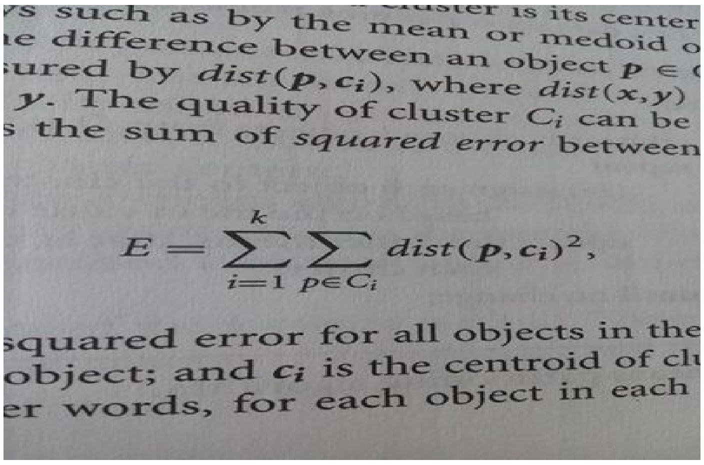
and function is **dist(p,c)** and it is the Euclidean distance between two points x and y. It looks like this:



Where „p“ is our **p** and „q“ is our **c**.

Within - Cluster Variation

- is a sum of squared error between all objects in a cluster and the centroid **c**, defined as:



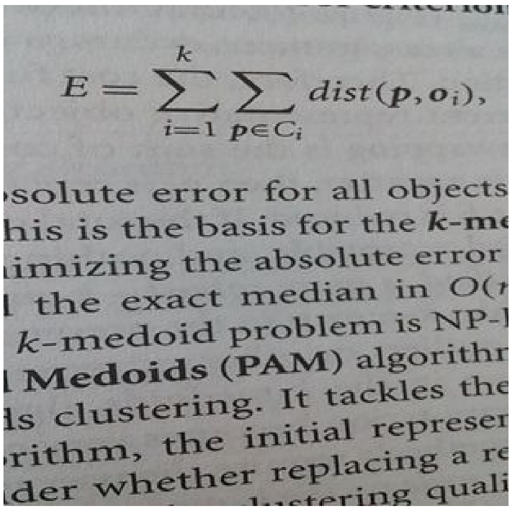
uster is its center
ys such as by the mean or medoid o
ie difference between an object $p \in C$
sured by $dist(p, c_i)$, where $dist(x, y)$
y. The quality of cluster C_i can be
s the sum of *squared error* between

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

squared error for all objects in the
object; and c_i is the centroid of cl
er words, for each object in each

Metrika pre k-Medoids, algoritmus patriaci do Partitioning Methods / Metódy zhlukovania

„How can we modify the k-means algorithm to diminish such sensitivity to outliers“? Instead of taking the mean value of the objects in a cluster as a reference point (centroid), we can pick actual objects to represent the clusters, using one representative object per cluster (one data value). Each remaining object is assigned to the cluster of which the representative object is the most similar. The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object **p** and its corresponding representative object. That is, an **absolute-error criterion** is used, defined as:



criterion

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, o_i),$$

solute error for all objects
his is the basis for the ***k*-m**
imizing the absolute error
l the exact median in O(r
k-medoid problem is NP-l
l Medoids (PAM) algorithm
ls clustering. It tackles the
rithm, the initial represer
der whether replacing a re
lustering quali

*** E is the sum of the absolute error for all objects **p** in the data set, and **o**i is the representative object of cluster Ci.

22. Opísať princíp klasifikácie, všeobecné kroky klasifikácie

A bank loans officer needs analysis of her data to learn which loan applicants are „safe“ and which are „risky“ for the bank. A marketing manager at AllElectronics needs data analysis to help guess whether a customer with a given profile will buy a new computer. In each of these examples, the data analysis task is **classification**, where a model or **classifier** is constructed to predict class, such as „safe“ or „risky“ for the loan application data. These categories can be represented by discrete values, where the ordering among values **has no meaning.** For example: the values 1, 2 and 3 may be used to represent treatments A, B and C, where there is no ordering implied among this group of treatment regimes.

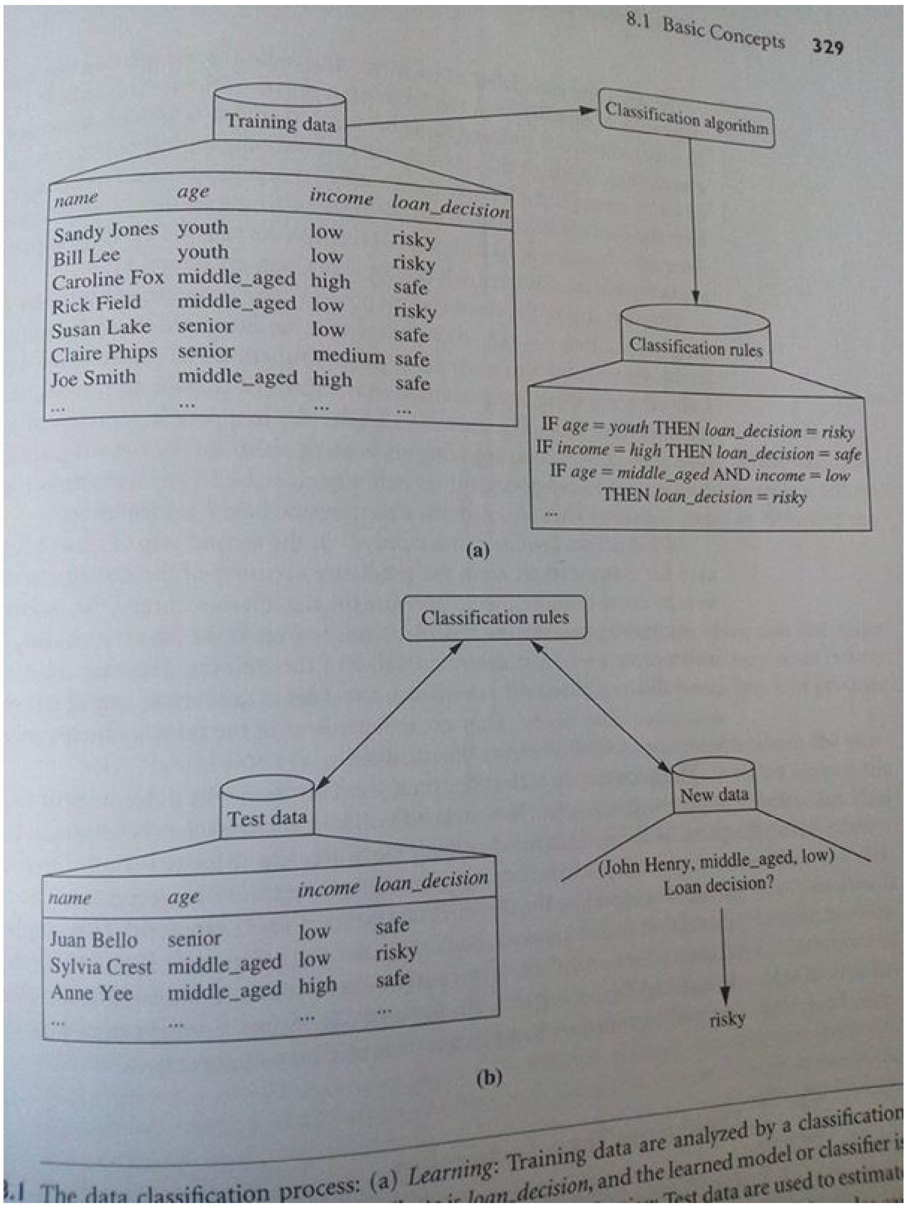
Suppose that the marketing manager wants to predict how much a given customer will spend during a sale at AllElectronics. This data analysis task is an example of **numeric prediction,** where the model constructed predicts a continuous-valued function, or ordered value, as opposed to a class. This model is not a **classifier, it is a predictor.**

Regression analysis is a statistical methodology that is most often used for numeric prediction. Classification and numeric prediction are the two major types of prediction problems.

Všeobecné kroky klasifikácie / General Approach to Classification
Data classification is a **two-step process,** consisting of a learning step (where a classification model is constructed) and a classification step (where the model is used to predict class for given data). Data classification process consists from:

1. Learning – training data are analyzed by a classification algorithm.
2. Classification – test data are used to estimate the accuracy of the classification rules.

It looks like this:



name	*age*	*income*	*loan_decision*
Sandy Jones	youth	low	risky
Bill Lee	youth	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe
...	...	...	...

IF *age* = *youth* THEN *loan_decision* = *risky*
IF *income* = *high* THEN *loan_decision* = *safe*
IF *age* = *middle_aged* AND *income* = *low*
THEN *loan_decision* = *risky*
...

name	*age*	*income*	*loan_decision*
Juan Bello	senior	low	safe
Sylvia Crest	middle_aged	low	risky
Anne Yee	middle_aged	high	safe
...	...	...	...

.1 The data classification process: (a) *Learning*: Training data are analyzed by a classification ... is *loan_decision*, and the learned model or classifier is ... Test data are used to estimat

23. opisat Naive Bayes a ako suvisi s podmienenou pravdepodobnostou

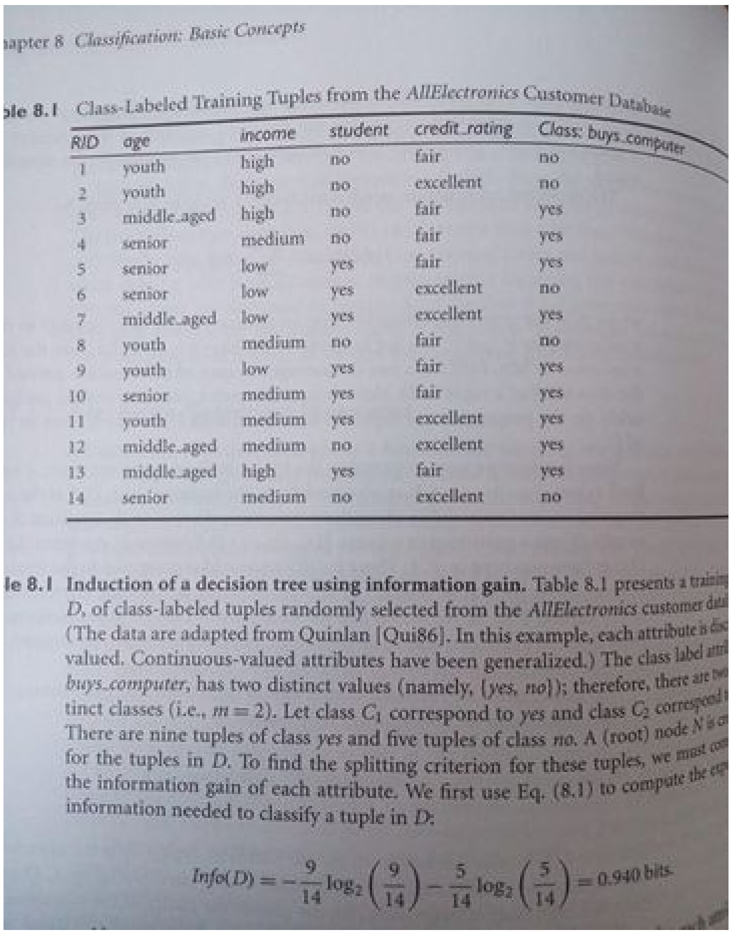
Naive Bayes

*** Okrem časti v ktorej som už tento klasifikátor opísal pridávam ďalšiu časť***

Naive Bayes classifier works as follows:

Let D be a training set of tuples and their associated class labels. These class labels are values in a column: **Class: buys_computer.** As usual, each tuple is represented by an n-dimensional attribute vector X depicting n measurements made on the tuple from n attributes, respectively attributes A.

So the X is every row in a data table and has 6 dimensions.



apter 8 *Classification: Basic Concepts*

ble 8.1 Class-Labeled Training Tuples from the *AllElectronics* Customer Database

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

le 8.1 **Induction of a decision tree using information gain.** Table 8.1 presents a trainin
D, of class-labeled tuples randomly selected from the *AllElectronics* customer dat
(The data are adapted from Quinlan [Qui86]. In this example, each attribute is dis
valued. Continuous-valued attributes have been generalized.) The class label attr
buys_computer, has two distinct values (namely, {*yes*, *no*}); therefore, there are tw
tinct classes (i.e., $m = 2$). Let class C_1 correspond to *yes* and class C_2 correspond
There are nine tuples of class *yes* and five tuples of class *no*. A (root) node *N* is cr
for the tuples in *D*. To find the splitting criterion for these tuples, we must co
the information gain of each attribute. We first use Eq. (8.1) to compute the ex
information needed to classify a tuple in *D*:

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

Suppose that there are m classes C1 to CM. Given a tuple X (row from a table above), the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naive Bayesian classifier predicts that tuple X belongs to the class Ci if and only if:

P(Ci | X) > P(Cj | X) for all of the 1 <= j <= m where j != i

Our classes are made of the two values {yes, no}, so class is only one and is buys_computer. If we want to say, that given row belongs to the class buys_computer=yes, we have to maximize formula above and make sure, that there is no row (rows are customers) which would have probability:

P(buys_computer = yes | given_row_with_another_customer) > P(buys_computer=yes | given_row_with_our_customer) **for all of the rows in a data table we dispose.**

Thus, we maximize P(Ci | X). The class Ci for which is P(Ci | X) maximized is called the **maximum posteriori hypothesis.** By Bayes theorem

Maximum posteriori hypothesis is buys_computer = yes where buys_computer is a class label and „yes“ is a class value.

Formula for computing probability, that our customer (our picked row) would buy a computer is equal to:
P(Ci | X) = (P(X | Ci) * P(Ci)) / P(X)

So if we want to predict it, we have to look on a following formula:

ned by dividing by *P(A)*:

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

and compute part of a formula numerator:
P(buys_computer = yes | our_young_customer) = (2/14)/(5/14)
P(buys_computer = yes **and** our_young_customer) = there are 2 rows with these values in a table, so it equals 2/14 where 14 is the number of rows in a table
P(our_young_customer) = 5 young customers / from 14 customers

We do not know class prior probabilites (there’s no such column) so we will assume that the classes are equally likely, that is their probabilites are same. In a other words, every unique value in a column buys_computer has distinct probability, other from another unique value in this column.

P(Ci | X) = (2/14) / (5/14)

24.Dolovanie frekventovanych grafov(nie mnozin), 2 metody a 1 priklad pouzitia // toto niekto ?

Graph Pattern Mining
Graph pattern mining is the mining of frequent subgraphs in one or a set of graphs.
Methods for mining graph patterns can be categorized into:

- **Apriori-based**
- **Pattern growth-based approaches**

Alternatively, we can mine the set of closed graphs where a graph **g** is closed if there exists no proper supergraph **g'** that carries the same support count (frequency) as **g.**
Moreover, there are many variant graph patterns, including:

- **Approximate frequent graphs**
- **Frequent graphs**
- **Coherent graphs**
- **Dense graphs**

User-specified constraints can be pushed deep into the graph pattern mining process to improve mining efficiency.
Graph pattern mining has many interesting applications. For example, it can be used to generate compact and effective graph index structures based on the concept of frequent and disciminative graph patterns. Approximate structure similarity search can be achieved by exploring graph index structures and multiple graph features. Moreover, classification of graphs can also be performed effectively using frequent and discriminative subgraphs as features.

25. Co je to sekvencia

Mining Sequence Data
A sequence is an ordered list of events. Sequences may be categorized into these three groups, based on the characteristics of the events they describe:

- Time-series data
- Symbolic sequence data
- Biological sequences

In **time-series data,** sequence data consist of long sequences of numeric data, recorded at equal time intervals. Time-series data can be generated by many natural and economic processes such as from stock markets, and scientific, medical, or natural observations.

Symbolic sequence data consist of long sequences of event or nominal data, which typically are not observed at equal time intervals. For many such sequences, gaps (lapses between recorded events) do not matter much. Examples include customer shopping sequence and web click streams, as well as sequences of events in science and engineering and in natural and social developments.

Biological sequences include DNA and protein sequences. Such sequences are typically very long, and carry important, complicated, but hidden semantic meaning. Here, gaps are usually important.

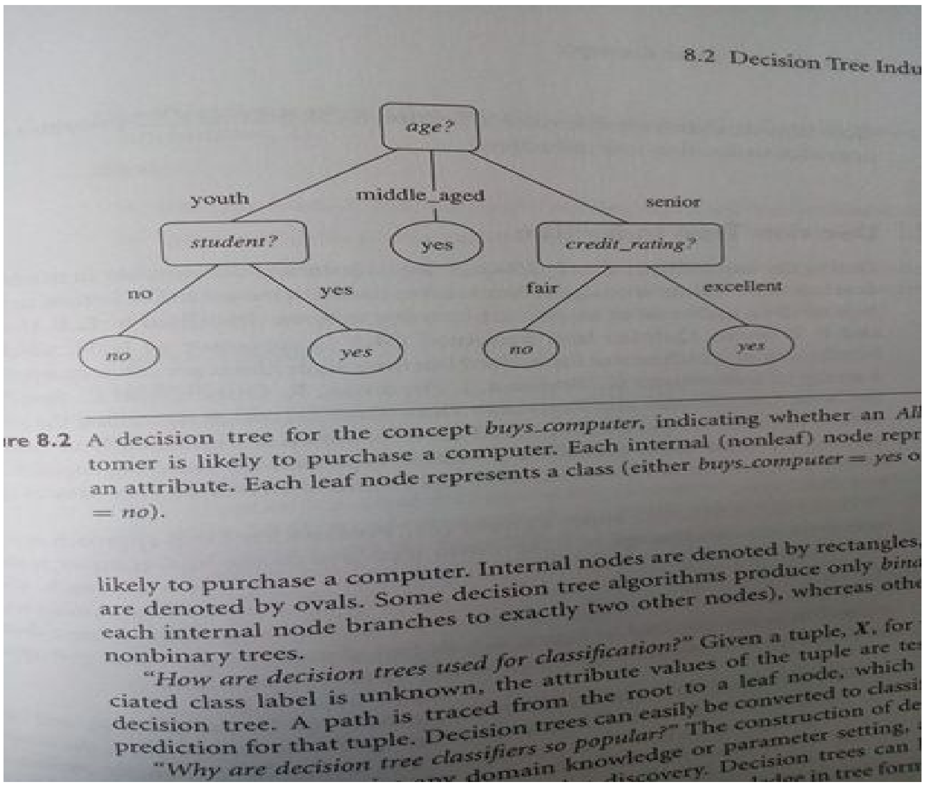
26. Ako fungujú rozhodovacie stromy

Decision Tree Induction
Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, where each **internal node** (non-leaf node) denotes a test on an attribute, each **branch** represents an outcome of the test, and each **leaf node** (or terminal node) holds a class label. The topmost node in a tree is the **root** node. Following tree predicts whether a customer at AllElectronics is likely to purchase a computer. Internal nodes are denoted by rectangles, and leaf nodes are denoted by ovals. Some decision tree algorithms produce only binary trees (where each internal node branches to exactly two other nodes), whereas others can produce nonbinary trees. Given a tuple X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules.

Table with tuples, class labels are unknown (there is no such column for class)

Age	Student	Credit Rating
Youth	Yes	Fair
Middle aged	Yes	Fair
Senior	no	Excellent

And the decision tree for this table is:



8.2 Decision Tree Indu
age?
youth
middle_aged
senior
student?
yes
credit_rating?
no
yes
fair
excellent
no
yes
no
yes

ıre 8.2 A decision tree for the concept *buys_computer*, indicating whether an *All*
tomer is likely to purchase a computer. Each internal (nonleaf) node repr
an attribute. Each leaf node represents a class (either *buys_computer = yes* o
= *no*).

likely to purchase a computer. Internal nodes are denoted by rectangles,
are denoted by ovals. Some decision tree algorithms produce only *bina*
each internal node branches to exactly two other nodes), whereas othe
nonbinary trees.

"*How are decision trees used for classification?*" Given a tuple, *X*, for
ciated class label is unknown, the attribute values of the tuple are te
decision tree. A path is traced from the root to a leaf node, which
prediction for that tuple. Decision trees can easily be converted to classi

"*Why are decision tree classifiers so popular?*" The construction of de
domain knowledge or parameter setting,
discovery. Decision trees can
in tree form

27. Metriky urcene na vyhodnocovanie asociacnych pravidiel

A comparison of Pattern Evaluation Measures
Instead of using the simple support-confidence framework (support is frequency, confidence is calculated) to evaluate frequent patterns, other measures such as **lift** and **chi-square** often disclose more intrinsic pattern relationships. Researchers have studied many pattern evaluation measures even before the start of in-depth research on scalable methods for mining frequent patterns. Recently, several other pattern evaluation measures have attracted interest. These measures are: **all_confidence, max_confidence, Kulczynski** and **cosine.**

Given two itemsets A and B where A and B are sets (množiny), the **all_confidence** measure of A and B is defined as:
All_conf(A, B) = sup(A or B) / max {sup(A), sup(B)} = min{P(A | B), P(B | A)}

Where max {sup(A), sup(B)} is the maximum support (itemset frequency) of the itemsets A and B. Thus all_conf(A, B) is also the minimum confidence of the two association rules related to A and B namely **A => B and B => A.**

Vlastný príklad výpočtu:
už som to v tomto docu počítal, ale ešte raz. Majme tabuľku zo začiatku tohto docu:

Tabuľka C1

Frequent 1-Itemset	Support count
{I1}	7 #7 krát sa nachádza v databáze D
{I2}	6
{I3}	8 #8 krát sa nachádza v databáze D
{I4}	2

a tabuľku C2, ktorá očividne vznikla spojením prvého stĺpca C1 so sebou samým:
Tabuľka C1

Frequent 1-Itemset	Support count
{I1, I2}	2 #2 krát sa nachádza v databáze D
{I2, I1}	4
{I3, I2}	8 #8 krát sa nachádza v databáze D
{I4, I1}	3

Potom all_confidence(tabuľka_C1, tabuľka_C2) vypočítam ako:
Zlúčením množín stĺpcov frekvencií oboch tabuliek získam sup(A alebo B) teda sup(C1)={7,6,8,2} a sup(C2)={2,4,8,3} z toho je ich zlúčenie sup(C1 alebo C2)={7,6,8,2,4,3}. Takže mám čitateľ vzorca, teraz potrebujem max {sup(A), sup(B)} to je po prepísaní max {{7,6,8,2}, {2,4,8,3}}. V prvej množine je najväčšie číslo 8, v druhej množine je najväčšie číslo tiež 8. Takže máme max {8, 8} z čoho vyplýva, že maximum = 8. Máme menovateľa vzorca ale výpočet spravíme tažko, keďže v čitateli je stále množina. Takže vzorec prepíšem na: min {P(A | B), P(B | A)} kde P(A | B) je P (C1 | C2). To čo som získal je minimálna vierohodnosť dvoch asociačných pravidiel ktoré patria tabuľkám C1 a C2. A tie máme kde? Tie si musíme vygenerovať. Takže:
Asociačné pravidlá patriace tabuľke C1
{I2} => {I1, I3, I4}
{I1} => {I3, I2, I4}
{I3} => {I4, I2, I1}
{I4} = {I3, I2, I1}
Predpokladajme, že po výpočte vierohodnosti všetkých týchto pravidiel dostanem, že minimálnu vierohodnosť má pravidlo {I2} => {I1, I3, I4} a jeho vierohodnosť je 2.
Vierohodnosť pre asociačné pravidlo sa počíta zo vzorca:
Vierohodnosť(A=>B) = P(B | A) = support_count (A alebo B) / support_count(A)
Ktorý sa prepisuje na „support_count(podmnožina_frequent_itemset) => (frequent_itemset – jej_podmnožina)“. Z datasetu si vypočítate početnosť riadkov, ktoré obsahujú ľavú stranu pravidiel (v hocijakom stĺpci) a túto početnosť predelíte početnosťou riadkov s pravou stranou pravidiel. Výsledok pridelíte pravidlu ako jeho vierohodnosť.

Toto isté spravíte pre tabuľku C2, vypočítate vierohodnosti pre všetky pravidlá, určíte pravidlo s minimálnou vierohodnosťou a jeho vierohodnosť rovnú 1 strčíte do vzorca. Min{2,1} = 1 a to je **all_confidence**

Given two itemsets A and B the **max_confidence** measure of A and B is defined as
Max_conf(A, B) = max {P(A | B), P(B | A)}
Where max_conf measure is the maximum confidence of the two association rules **A => B, B => A.**

Given two itemsets, A and B, the **Kulczynski** measure of A and B (abbreviated as **Kulc**) is defined as:
Kulc(A, B) = ½*(P(A | B) + P(B | A))
Pozn. Všetky výrazy ktoré sú v týchto vzorcoch som vypočítal a na všetky som ukázal príklad výpočtu, prosím nájsť cez Ctrl + F

28. Cistenie dat (neuplnost, zasumenie, nekonzistencia, duplicity) plus vybrat dve a napisat sposob riesenia

Real-world data tend to be incomplete, noisy, and inconsistent. **Data cleaning** routines attempt to fill in missing values, smooth out noise while identifiying outliers, and correct inconsistencies in the data.

How can we go about filling in the missing values for an attributes?

- We can ignore the tuple. This method is especially poor when the percentage of missing values per attribute varies considerably.
- We can fill in the missing values manually. In general, this approach is time consuming and may not be feasible given a large data.
- We can use a global constant to fill in the missing value by a replacing all missing attribute values by the same constant such as a label like „Unknown“.
- We can use a measure of central tendency for the attribute. For normal (symmetric) data distributions, the mean can be used, while skewed data distribution should employ the median.
- We can use the attribute mean or median for all samples belonging to the same class as the given tuple. For example, if classifying customers according to credit risk, we may replace the missing value with the mean income value for customers in the same credit risk category as that of the given tuple.
- We can use the most probable value to fill in the missing value. This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction.

Techniques for noisy data

Noise is a random error or variance in a measured variable. Some of those techniques are:

- Binning: binning methods smooth a sorted data value by consulting its neighborhood, that is, the values around it. The sorted values are distributed into a number of buckets, or bins. Because binning methods consult the neighborhood of values, they perform local smoothing. In smoothing by bin means, each value in a bin is replaced by the mean value of the bin. Similarly, smoothing by bin medians can be employed, in which each bin value is replaced by the bin median. In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries.
- Regression: data smoothing can also be done by regression, a technique that conforms data values to a function. Linear regression involves finding the best line to fit two attributes so that one attribute can be used to predict the other. Multiple linear regression is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.
- Outlier Analysis: outliers may be detected by clustering, for example, where similar values are organized into groups, or „clusters“.

Process and steps of data cleaning

The first step in data cleaning as a process is discrepancy detection. Discrepancies can be caused by several factors, including poorly designed data entry forms that have many optional fields, human error in data entry, deliberate errors (respondents do not want to divulge information about themselves), and data decay (outdated addresses).

29. Co je to nekonzistencia dat a ako sa riesi

Data Integration is a technique that can help and avoid redundancies and inconsistencies in a dataset

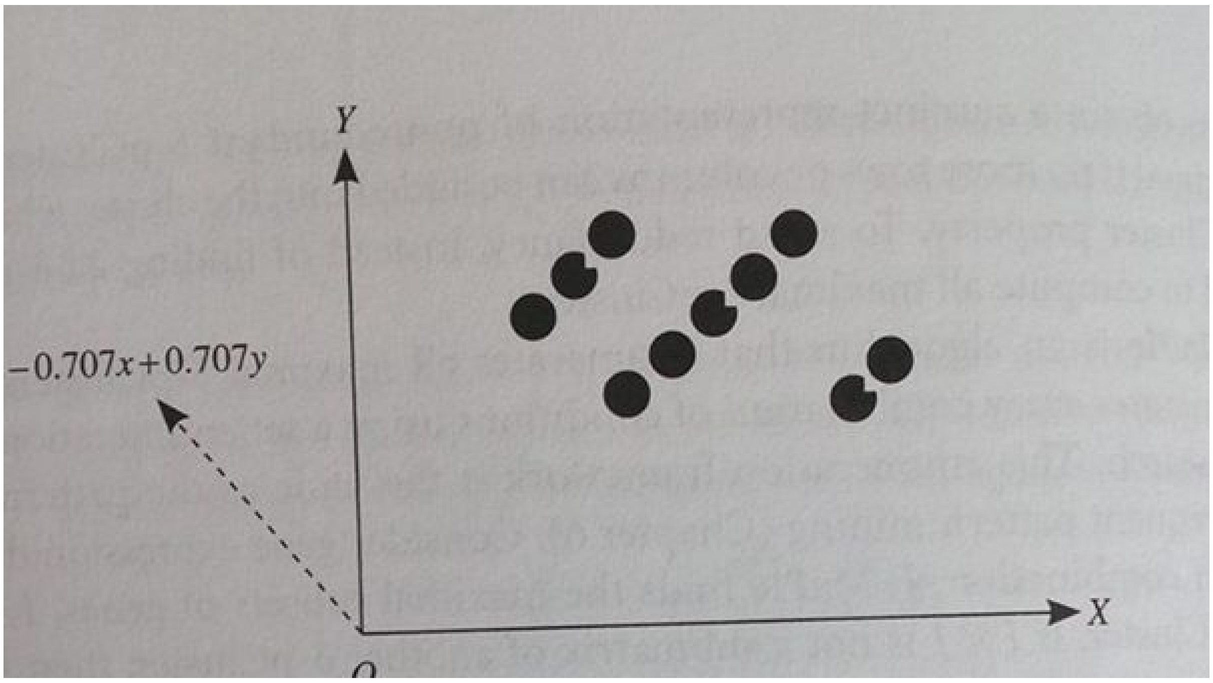
Data integration is merging the data from multiple data stores. There are number of issues to consider during data integration. Schema integration and object matching can be tricky. When matching attributes from one database to another during integration, special attention must be paid to the structure of the data. This is to ensure that any attribute funcional dependencies and referential constraints in the source system match those in the target system. Redundancy of data is another important issue in data integration. Some redundancies can be detected by correlation analysis. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data. In addition to detecting redundancies between attributes, duplication should also be detected at the tuple level. Data integration also involves the detection and resolution of data value conflicts.

30. Znizovanie dimenzionality dat

Dimensionality reduction approaches try to construct a much lower-dimensional space and search for clusters in such a space. Often, a method may construct new dimensions by combining some dimensions from the original data.

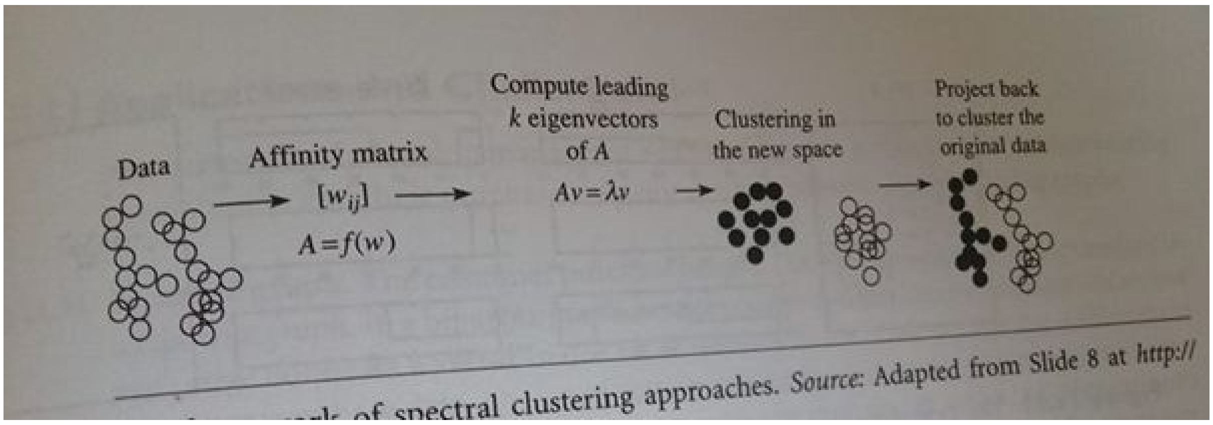
Dimensionality Reduction Methods and Spectral Clustering

Subspace clustering methods try to find clusters in subspaces of the original data space. In some situations, it is more effective to construct a new space instead of using subspaces of the original data. This is the primary motivation behind dimensionality reduction methods. Consider these three clusters:



It is not possible to cluster these points in aby subspace of the original space X x Y because all three clusters would end up being projected onto overlapping areas in the x and y axes. What if, instead we construct a new dimension (-(sqrt(2)/2)*x + (sqrt(2)/2)*y). This formula represents the dashed line in a picture. By projecting the points onto this new dimension, the three clusters become apparent.

There are many dimensionality reduction methods. A straightforward approach is to apply feature selection and extraction methods to the dataset. However, such methods may not be able to detect the clustering structure. **Spectral methods** are a group of methods that are effective in high-dimensional data applications. This is an example of one such method **Ng-Jordan-Weiss algorithm – a special clustering method.**



of spectral clustering approaches. *Source:* Adapted from Slide 8 at *http://*

Other dimensionality reduction methods but of time-series data ara:

- Discrete Fourier transform
- Discrete wavelet transforms
- Singular value decomposition
- Principal Component Analysis

31. Sekvencia a sekvencny vzor

Sequenca data are:

- Time-series data (stock market data)
- Symbolic sequences (customer shopping sequences, web click streams)
- Biological sequences (DNA and protein sequences)

Time-series data are mined by Similarity Search methods
Unlike normal database queries, which find data that match a given query exactly, a similarity search finds data sequences that differ only slightly from the given query sequence. Many time-series similiarity queries require subsequence matching, that is, finding a set of sequences that contain subsequences that are similar to a given query sequence. We use a regression and trend analysis for those data in order to mine them. Trend analysis build an integrated model using the following four major components to characterize time-series data:

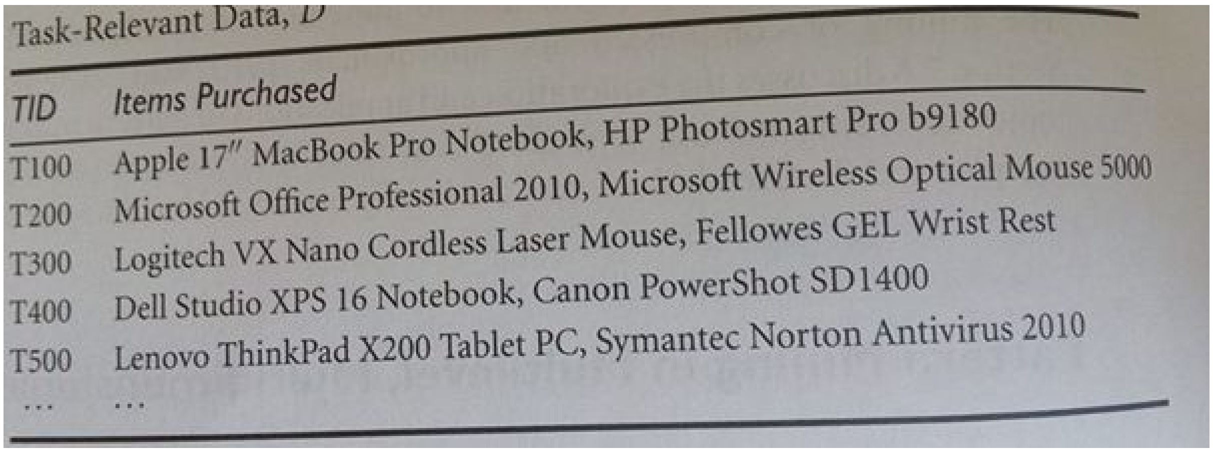
- Trend or long-term movements, which indicate the general direction in which a time-series graph is moving over time
- Cyclic movements, which are the long-term oscillations about a trend line or curve
- Seasonal variations – which are nearly identical patterns that a time series appears to follow during corresponding seasons of successive years such as holiday,...
- Random movements, they characterize sporadic changes due to chance events

Symbolic sequencs are mined by Sequential Pattern Mining methods
A symbolic sequence consists of an ordered set of elements or events, recorded with or without a concrete notion of time.

A sequential pattern is a frequent subsequence existing in a single sequence or a set of sequences. A sequence A=<a1,a2,...,an> is a subsequence of another sequence B=<b1,b2,...,bm> if there exist integers 1 <= j1 < j2 < ... < jn <= m such that a1 is a subset of bj1, a2 is a subset of bj2 and so on. For example, if A=<{ab},d> and B=<{abc},{be},{de},a> where a,b,c,d,e are items, then A is a subsequence of B.

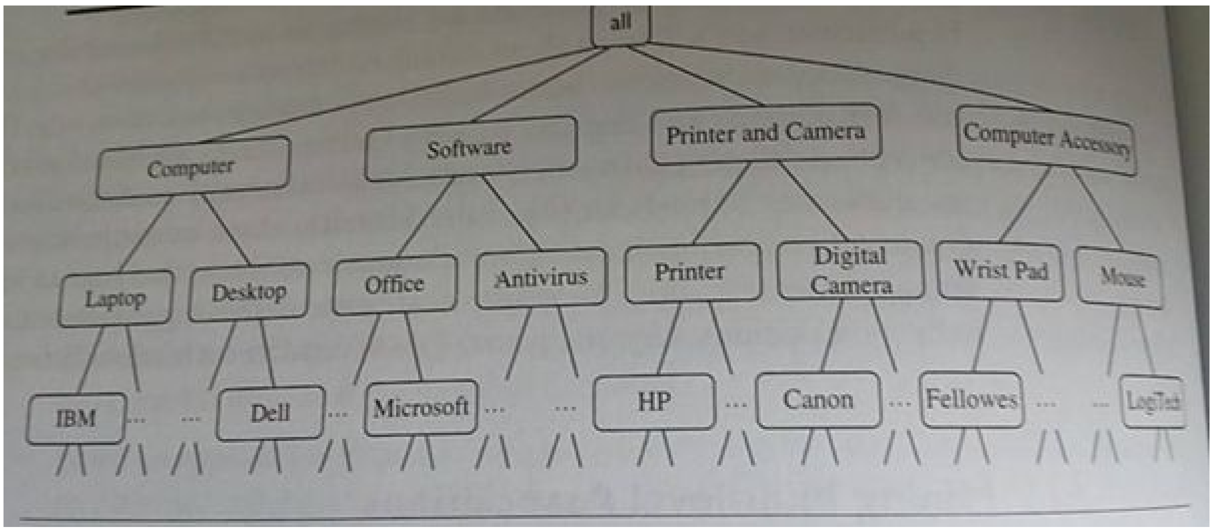
32. Viacurovnove asociacne pravidla - vysvetlit, priklad tabulky

Multilevel associations (or multilevel association rules) involve data at more than one abstraction level („buys computer“or „buys laptop“). These may be mined using multiple minimum support thresholds (to sú minimálne hodnoty početností množín, ktoré si predom zadáme). **Multidimensional associations** contain more than one dimension. Techniques for mining such associations differ in how they handle repetitive predicates. **Quantitative association rules** involve quantitative attributes. Discretization, clustering, and statistical analysis that discloses exceptional behavior can be integrated with the pattern mining process.
How to mine Multilevel Association Rules
Suppose we are given the task-relevant set of transactional data in this table:


Task-Relevant Data, *D*

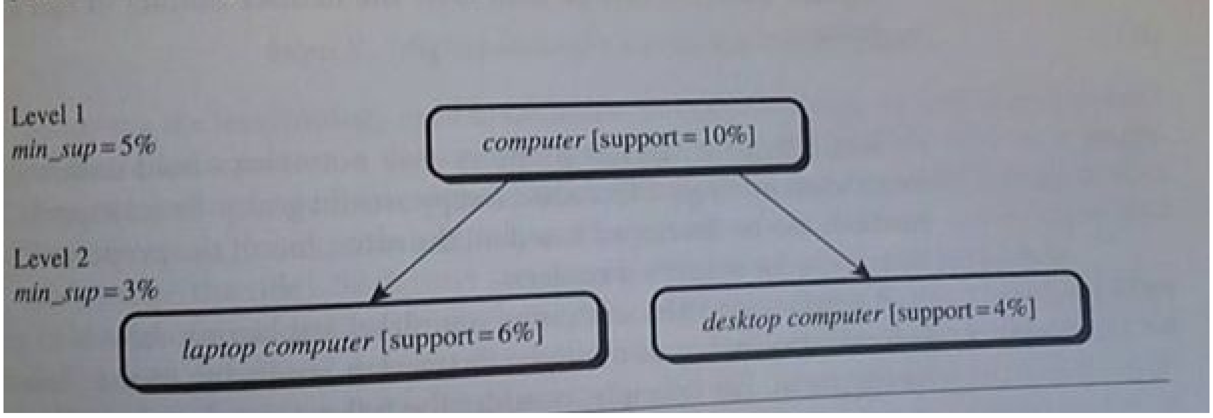
TID	*Items Purchased*
T100	Apple 17″ MacBook Pro Notebook, HP Photosmart Pro b9180
T200	Microsoft Office Professional 2010, Microsoft Wireless Optical Mouse 5000
T300	Logitech VX Nano Cordless Laser Mouse, Fellowes GEL Wrist Rest
T400	Dell Studio XPS 16 Notebook, Canon PowerShot SD1400
T500	Lenovo ThinkPad X200 Tablet PC, Symantec Norton Antivirus 2010
…	…

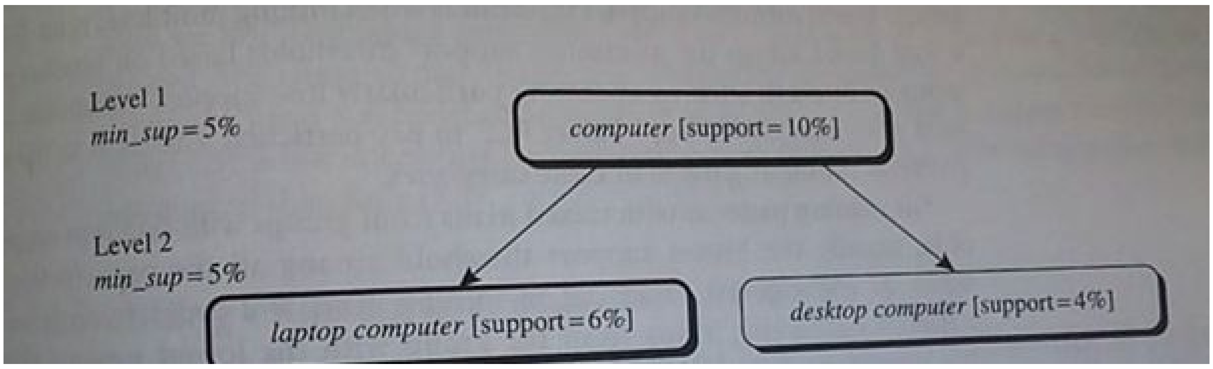
Each row / each transaction in this transactional data table shows the items purchased in each transaction. We will make a concept hierarchy from this table which will define a sequence of mappings from a set of low-level concepts to a higher-level, more general concept set. Data in this table can be generalized by replacing low-level concepts within them by their corresponding higher-level concepts, or ancestors, from a concept hierarchy. It looks like this:



Association rules generated from mining data at multiple abstraction levels are called **multiple-level** or **multilevel association rules.** Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework. In general, a top-down strategy is emplyoed, where counts are accumulated for the calculation of frequent itemsets at each concept level, starting at concept level 1 and working downward in the hierarchy toward the more specific concept levels, until no more frequent itemsets can be found. For each level, any algorithm for discovering frequent itemsets may be used, such as **Apriori** (ktorý som tu už 2 krát vysvetlil) or its variations.
On a next pictures I provide pay attention to minimum support threshold (which is min_sup) and to borders of nodes. Thick borders mean that these nodes are frequent, instead they're not because do not satisfy minimum frequency threshold.

Multilevel mining with uniform support.





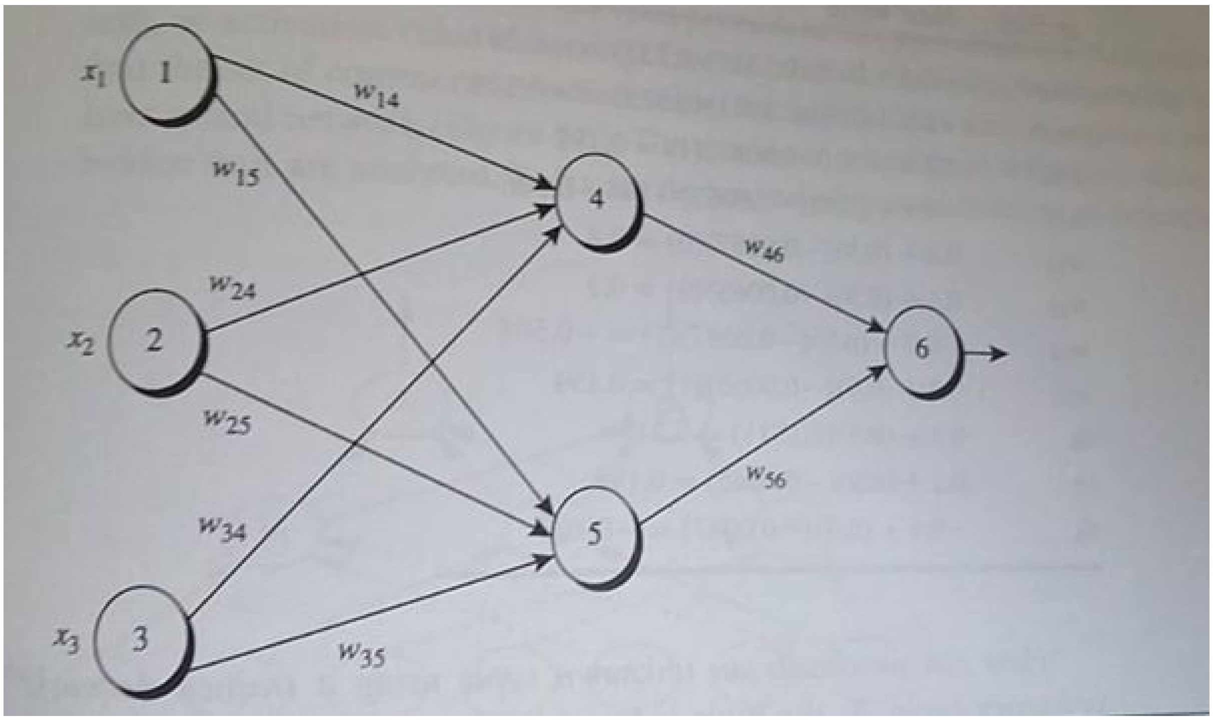
Using uniform minimum support for all levels (second picture) says, that the same minimum support threshold is used when mining at each abstraction level. Multilevel mining does not do this, because has different minimum support thresholds (5 % and 3 %). For example a minimum support threshold of 5 % in a second picture is used throughout mining from computer downward to laptop computer. Both computer and laptop computer are found to be frequent (because they satisfy a condition), whereas desktop computer is not frequent. Uniform minimum support threshold (in a second picture) simplify search procedure but reduced support threshold does not.

33. Neuronova siet - Backpropagation

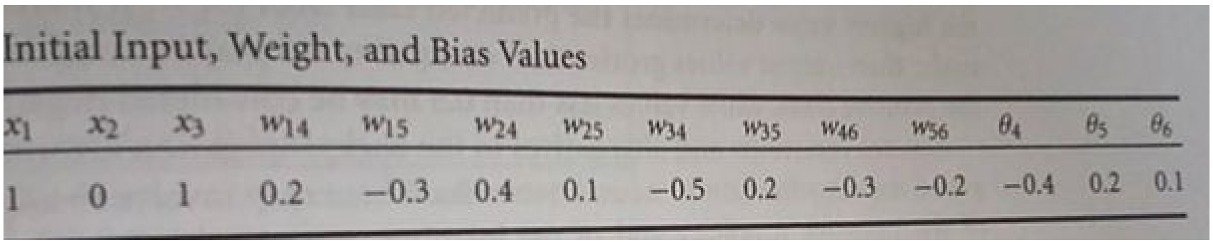
Classification by Backpropagation
Backpropagation is a neural network learning algorithm. The neural networks field was originally kindled by psychologists and neurobiologists who sought to develop and test computational analogs of neurons. Neural network is a set of connected input/output units in which each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label/value (such as buys_computer=yes where yes is a class value) of the input tuples/rows. Neural network learning is also referred to as **connectionist learning** due to the connections between units.

Backpropagation learns by iteratively processing a dataset of training tuples, comparing the network's prediction for each tuple with the actual known target value. The target value may be the known class label (yes for a class buys_computer) of the training tuple (for classification problems) or a continuous value (for numeric prediction). For each training tuple, the weights are modified so as to minimize the mean-squared error between network's prediction and the actual target value. These modifications are made in the backwards direction through each hidden layer down to the first hidden layer. Although it is not guaranteed, in general the weights will eventually converge, and the learning process stops.

Example:
Imagine this neural network:



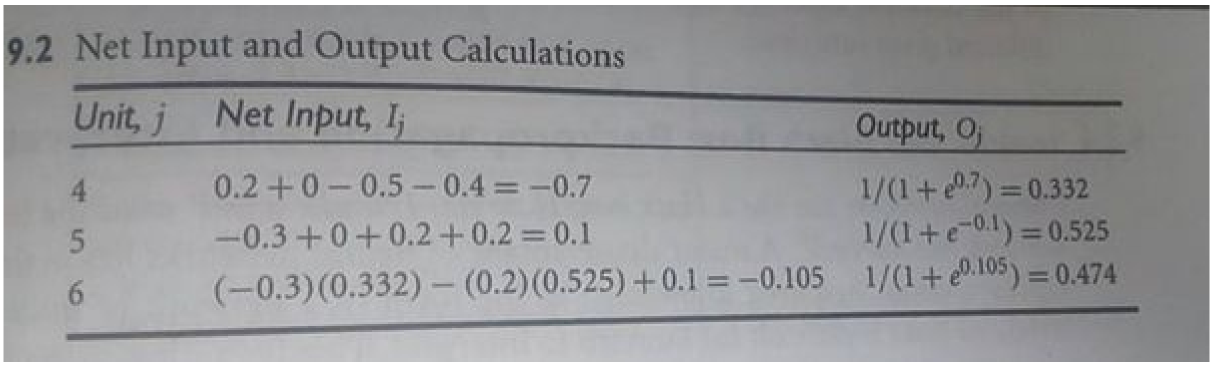
Let the learning rate be 0.9. Backpropagation function takes dataset, learning rate and network params. The initial weight and bias values of the network are given in this table:



Initial Input, Weight, and Bias Values

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

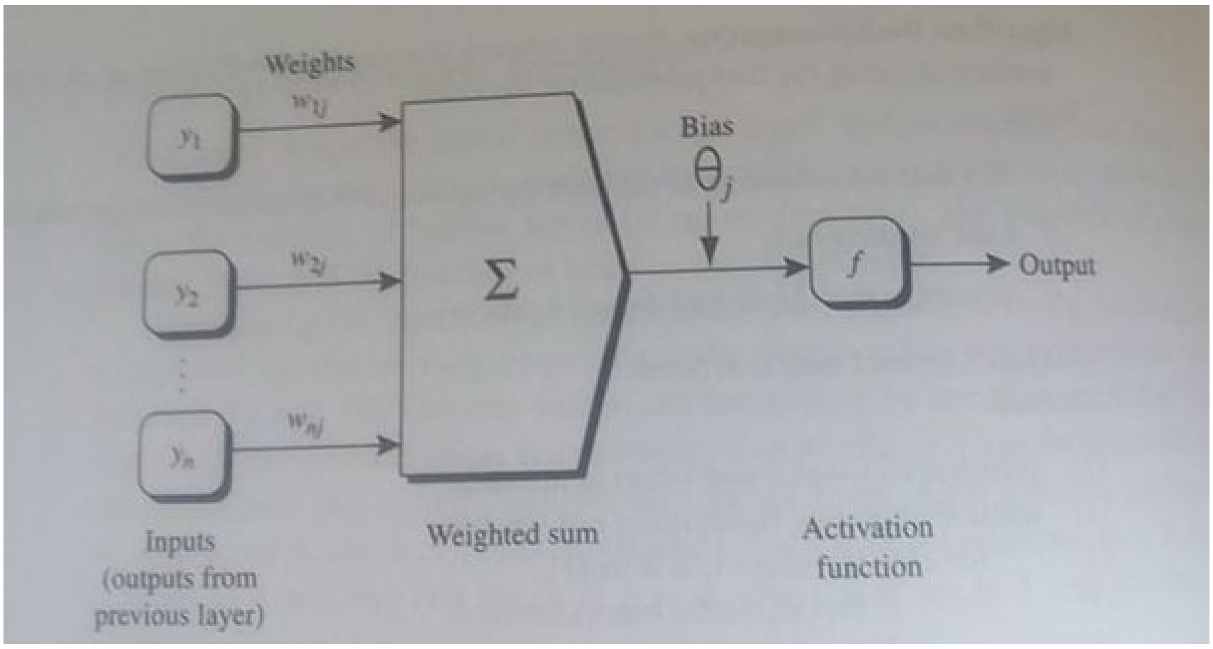
Initial weights and bias values are always small random numbers.
First training tuple/row is made of three attribute values X=(x1=1,x2=0,x3=1)
This tuple X is fed into the network, and the net input and output of each unit are computed. Results are in this table:



9.2 Net Input and Output Calculations

Unit, j	Net Input, I_j	Output, O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

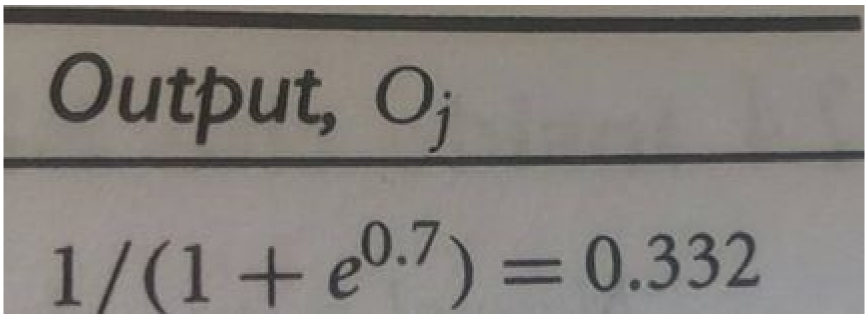
So how they were computed?
First training tuple X=(x1, x2, x3) is fed to the network's input layer. Please look at the network again and find this layer. It is first layer of three nodes from the left side. These inputs (nodes x1, x2 and x3) are passed through the unit „j“ to the next layer unchanged. What am I saying?
Unit „j“ is this:



This picture is a layer, hidden layer because we do not see it on the network's graph. It is also output layer, because we have three layers in the network's graph (please, look on it). This layer is called „unit j“ and has its own inputs which are outputs from the previous layer. So we have three layers in the network's graph and also have THREE unit js (nodes: 4, 5 and 6).
Inputs to this „unit j“ are multiplied by their corresponding weights. So for our network graph:

- From node 1 rise two branches w14 and w15 (to node 5 in a second layer) but I pay attention to the branch w14 which ends up in a node 4. It's because I want to calculate unit 4
- From node 2 in a first layer from left rises two branches but only one which ends up in a node 4. This branch is a w24.
- From node 3 in a first layer from the left rises only one branch which ends up in a node 4 (node 4 is our unit 4). This branch is w34

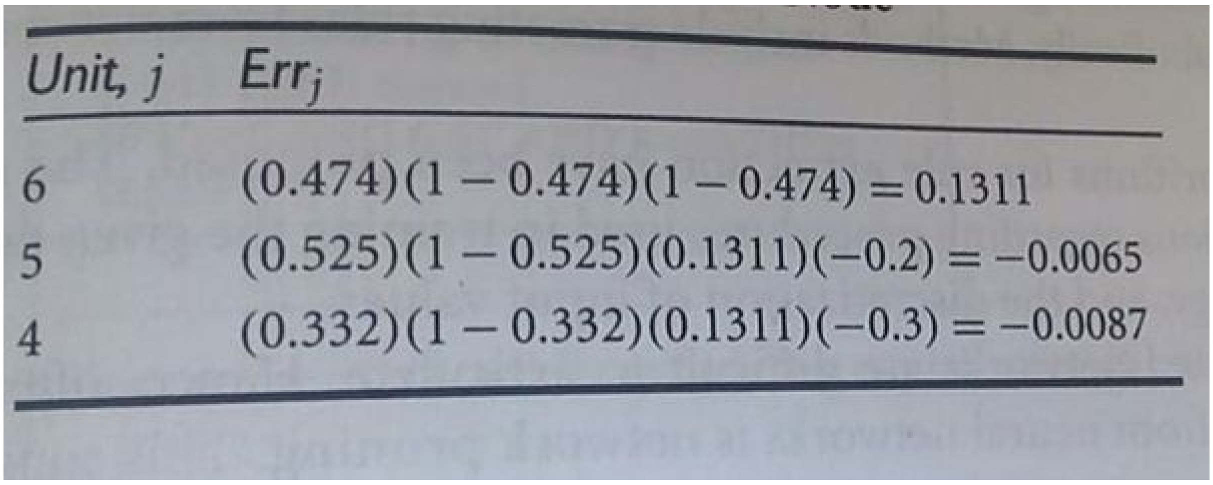
So net input of Unit 4 is:
(bias value for w14) + (bias value for w24) + (bias value for w34) = (0.2) + (-0.5) + (-0.4) = 0.2 – 0.5 – 0.4 = 0.2 – 0.9 = -0.7
Absolute value of the result |-0.7| = 0.7 will be an exponent to this formula:



Output, O_j
$1/(1+e^{0.7}) = 0.332$

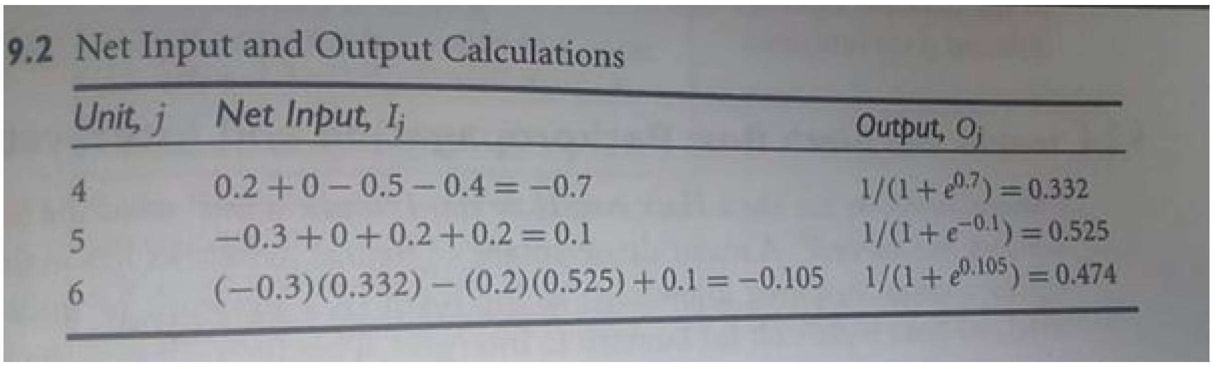
Result 0.332 is an Output of our hidden layer Unit 4. Unit 4 is hidden layer because I needed to calculate it and it's not part of our neural network.
Error of our node Unit 4 is as following:
Error_of_unit4=(output_we_calculated)*(1-output_we_calculated)*(target_value_of_our_tupleX-output_we_calculated)*(input_of_unit4_for_unit6)=
(0.332)*(1-0.332)*(0.1311)*(-0.3)=-0.0087

If you do this for all three Units you will get:



Unit, j	Err_j
6	$(0.474)(1-0.474)(1-0.474) = 0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2) = -0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3) = -0.0087$

How can we classify an unknown tuple using a trained network?
To classify an unknown tuple X we have to know that tuple X is input to the trained network, and compute the net input and output of each unit (we have unit 4,5 and 6). Results are here:



9.2 Net Input and Output Calculations

Unit, j	Net Input, I_j	Output, O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7}) = 0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1}) = 0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105}) = 0.474$

If there is one output node per class, then the output node with the highest value determines the predicted class label for X. If there is only one output node, then output values greater than or equal to 0.5 may be considered as belonging to the positive class, while values less than 0.5 may be considered negative.

34. Vyhody a nevyhody neuronovych sieti

Efficiency of Backpropagation, because its a method used in neural networks depends on the time spent training the network. Given |D| tuples and W weights, each epoch requires O(|D| x W) time. However, in the worst-case scenario, the number of epochs can be exponential in N, where N is the number of inputs. In practice, the time required for the networks to converge is highly variable. A number of techniques exist that help speed up the training time. For example, a technique known as **simulated annealing (simulované žíhanie)** can be used, which also ensures convergence to a global optimum.

35. Cistenie dat

Data Cleaning as a Process

The first step in data cleaning as a process is discrepancy detection. Discrepancies can be caused by several factors, including poorly designed data entry forms that have many optional fields, human error in data entry, deliberate errors (respondents not wanting to divulge information about themselves), and data decay (outdated addresses). Discrepancies may also arise from inconsistent data representations and inconsistent use of codes. Other sources of discrepancies include errors in instrumentation devices that record data and system errors. Errors can also occur when the data are (inadequately) used for purposes other than originally intended. There may also be inconsistencies due to data integration (where a given attribute can have different names in different databases).

As a starting point, use any knowledge you may already have regarding properties of the data. Such knowledge or data about data is referred to as metadata. Field overloading is another error source that typically results when developers squeeze new attribute definitions into unused (bit) proportions of already defined attributes (an unused bit of an attribute that has a value range that uses only, say, 31, out of 32 bits).
The data should also be examined regarding unique rules, consecutive rules, and null rules. A **unique rule** says that each value of the given attribute must be different from all other values for that attribute. A **consecutive rule** says that there can be no missing values between the lowest and highest values for the attribute, and that all values must also be

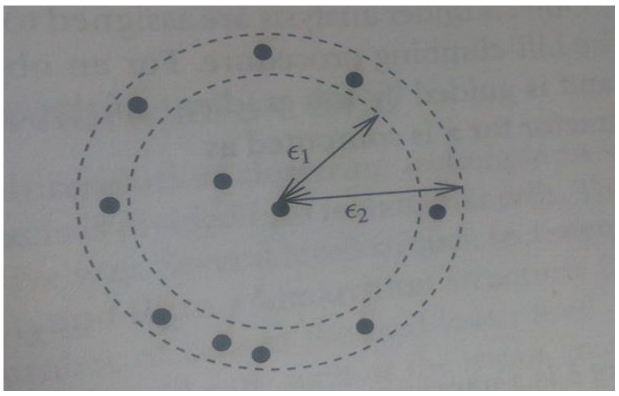
unique. A **null rule** specifies the use of blanks, question marks, special characters, or other strings that may indicate the null condition (where a value for a given attribute is not available) and how such values should be handled.

There are number of different commercial tools that can aid in the discrepancy detection step. **Data scrubbing tools** use simple domain knowledge to detect errors and make corrections in the data. **Data auditing tools** find discrepancies by analyzing the data to discover rules and relationships, and detecting data that violate such conditions. Commercial tools can assist in the data transformation step. **Data migration tools** allow simple transformations to be specified such as to replace the string „gender“ by „sex“.
ETL (extraction/transformation/loading) tools allow users to specify transforms through a graphical user interface.

36. DENCLUE: Clustering Based on Density Distribution Functions

Density estimation is a core issue in density-based clustering methods. DENCLUE is a clustering method based on a set of density distribution functions. In probability and statistics, density estimation is the estimation of an unobservable underlying probability density function based on a set of observed data. In the context of density-based clustering, the unobservable underlying probability density function is the true distribution of the population of all possible objects to be analyzed. The observed data set is regarded as a random sample from that population.

In DBSCAN and OPTICS, density is calculated by counting the number of objects in a neighborhood defined by a radius parameter E. Such density estimate can be highly sensitive to the radius value used. Let’s consider following example, where we increase radius parameter E by a small amount:



To overcome this problem, **kernel density estimation** can be used, which is a nonparametric density estimation approach from statistics. In this method we treat an observed object as an indicator of high-probability density in the surrounding region. The probability density at a point depends on the distances from this point to the observed objects.

Formally, let X=x1, ... , xn be an independent and identically distributed sample of a random variable F. The kernel density approximation of the probability density function is:

KDAPDS <- (1/(NROW(X)*smoothing_parameter))*∑kernel((point_x – xi)/smoothing_parameter)

*** to preto, lebo počítame hustotu rozdelenia pravdepodobnosti čo je funkcia a funkci počítame v niakom bode. Tým bodom je point_x a smoothing_parameter musí byť zadaný. Funkciu NROW(X) som prebral z R-ka, vracia počet riadkov vektora X ktorý môže byť stĺpec. Resp. použite funkciu length().***

Technické detaily DENCLUE:
DENCLUE uses a Gaussian kernel to estimate density based on the given set of objects to be clustered. It is because function Kernel is defined as follows:

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x - x_i)^2}{2h^2}}.$$

Why?
Because formula on its right side is a normal distribution function and on a left side is Kernel function. It implies that Kernel function is defined by a normal distribution applied on the Kernel function.

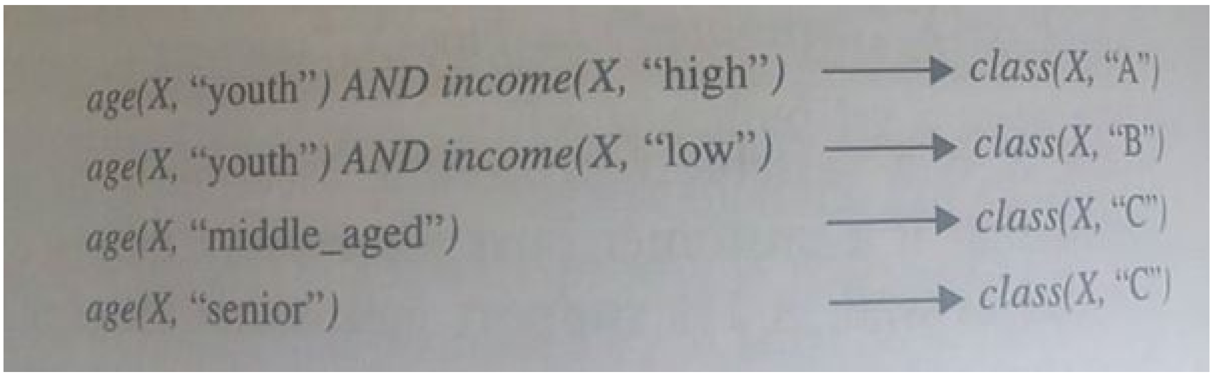
What is point_x?
point_x is a density attractor if it is a local maximum of the estimated density function.

What are advantages of the DENCLUE?
DENCLUE can be regarded as a generalization of several well-known clustering methods such as single-linkage approaches and DBSCAN. DBSCAN is invariant against noise. The kernel density estimation of DENCLUE can effectively reduce the influence of noise by uniformly distributing noise into the input data.

37. Rozhodovacie stromy

Decision trees
A decision tree is a flowchart-like tree structure, where each node denotes a test on an attribute value, each branch represents an outcome of the test, and tree leaves represent classes or class distributions. Decision trees can easily be converted to classification rules as following:

Suppose we have these rules:



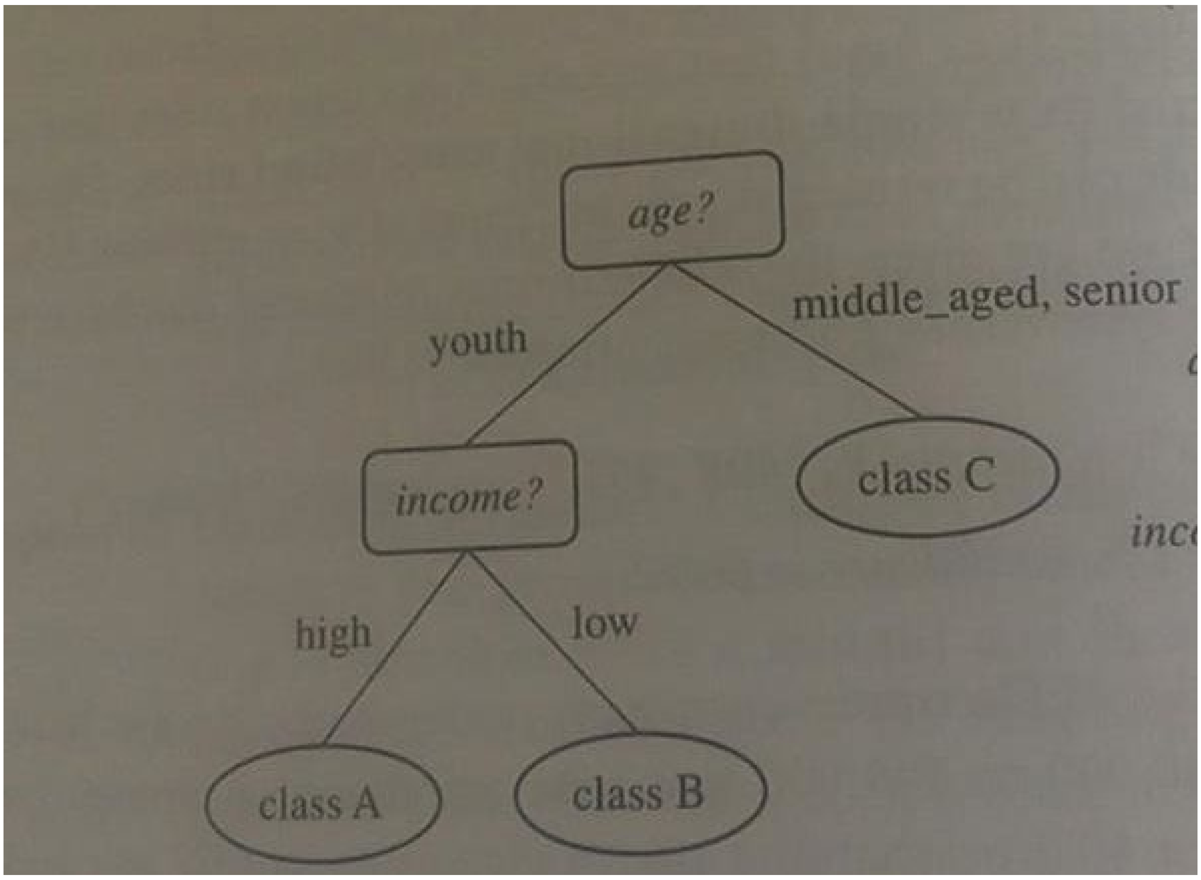
age(X, "youth"*) AND income(X,* "high"*)* $\longrightarrow$ *class(X,* "A"*)*

age(X, "youth"*) AND income(X,* "low"*)* $\longrightarrow$ *class(X,* "B"*)*

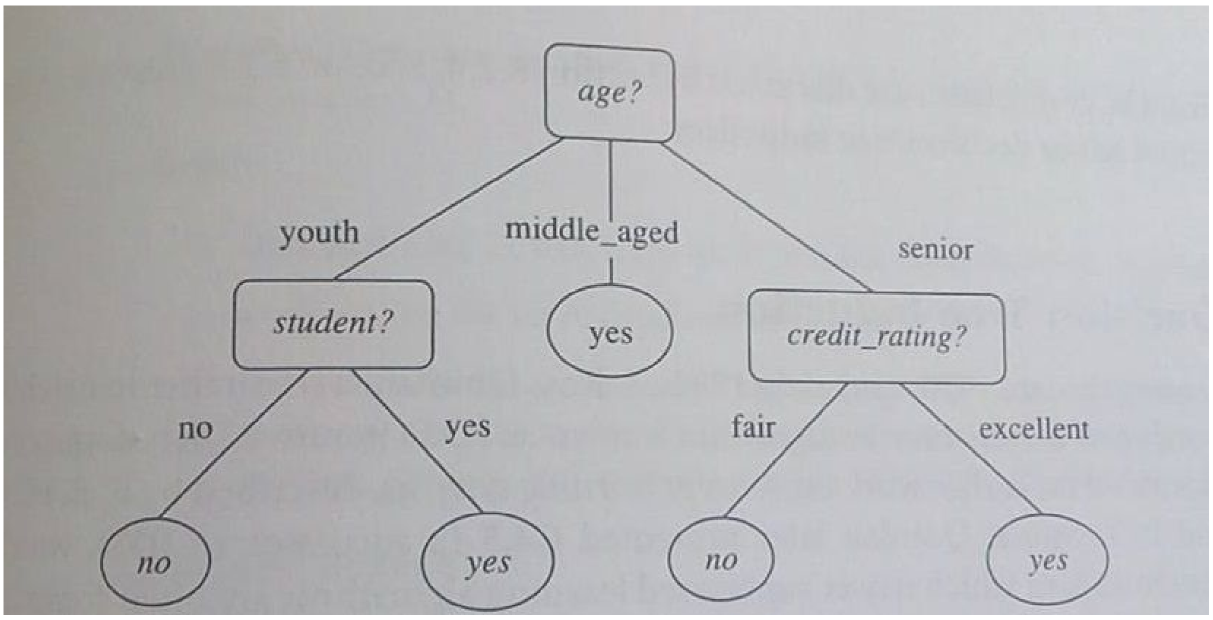
age(X, "middle_aged"*)* $\longrightarrow$ *class(X,* "C"*)*

age(X, "senior"*)* $\longrightarrow$ *class(X,* "C"*)*

They can be converted into a tree like:



A typical decision tree is shown in a next picture:



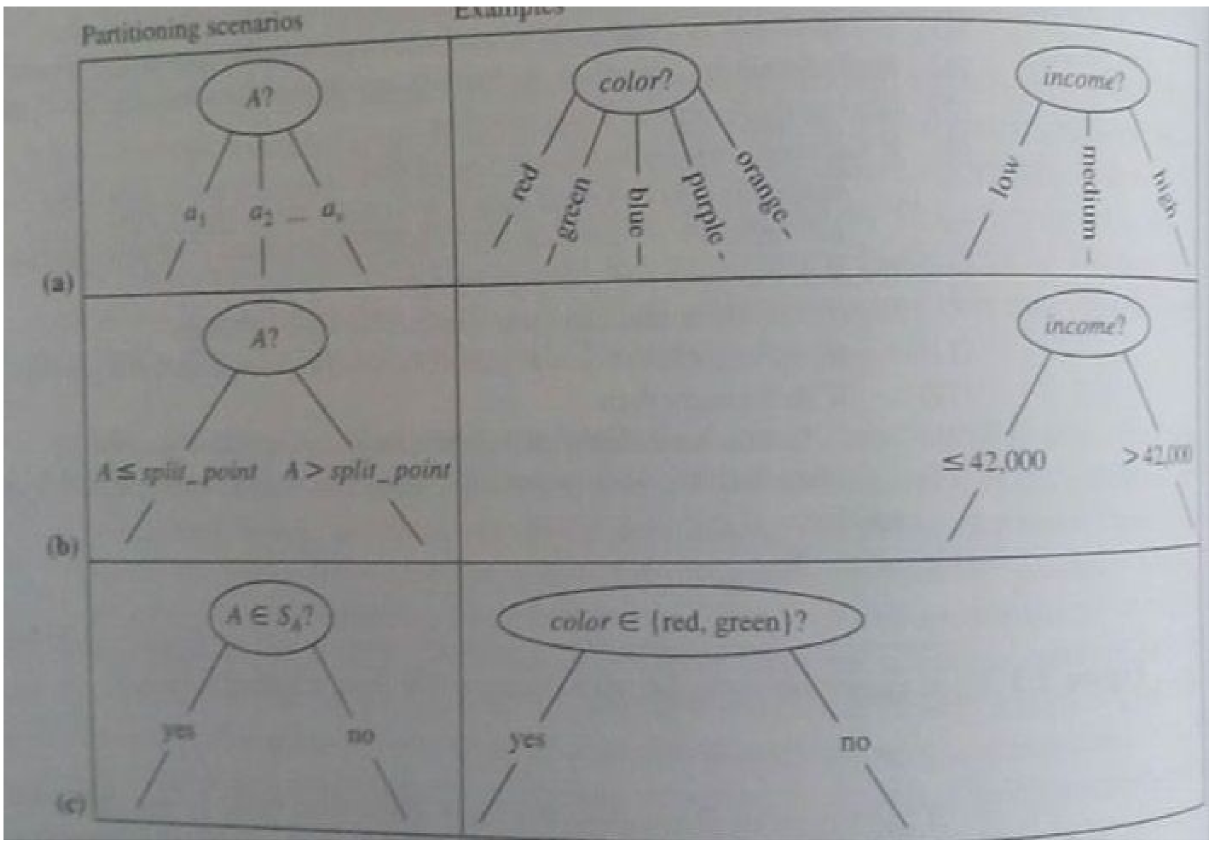
This tree represents the concept buys_computer, that is, it predicts whether a customer is likely to purchase a computer. Internal nodes are denoted by rectangles, and leaf nodes are denoted by ovals. Given a tuple X, for which the associated class label is unknown (there's no column for the classes in a dataset), the attribute values of the tuple (values for dataset features) are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple (yes means that customer will probably buy a computer). During the late 1970s J. Ross Quinlan developed a decision tree algorithm known as **ID3 (Iterative Dichotomiser).** Quinlan later presented **C4.5 (a successor of ID3).** In 1984 a group of statisticians published **CART (Classification and Regression Trees).** ID3, C4.5 and CART adopt a greedy (nonbacktracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner. Training set is recursively partitioned into smaller subsets as the tree is being built. Strategy of building a decision tree is as follows:

1. The tree starts as a single node N representing the training tuples in D. The partition of class-labeled training tuples (rows which have known class in some column) at node N is the set (množina) of tuples that follow a path from the root of the tree to node N when being processed by the tree. This set is sometimes referred as the family of tuples at node N. Rather than storing the actual tuples at a node, most implementations store pointers to these rows.
2. If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class. All terminating conditions of a function implementing building trees are usually wrote at the end of the function (bottom lines).
3. If the tuples in D are not of the same class, the algorithm calls **Attribute_selection_method** to determine the splitting criterion. The splitting criterion tells us which attribute to test at node N by determining the best way to separate or partition the tuples in D into individual classes. The splitting criterion also tells us which branches to grow from node N with respect to the outcomes of the chosen test. The splitting criterion indicates the splitting attribute and may also indicate either a split-point or a splitting subset.

4. The node N is labeled with the splitting criterion, which serves as a test at the node. A branch is grown from node N for each of the outcomes of the splitting criterion. The tuples in D are partitioned accordingly. There are 3 possible scenarios. Let A be the splitting attribute. A has V distinct values {a1, a2, …, av} based on the training data. So:

- **A is discrete-valued:** in this case the outcomes of the test at node N correspond directly to the known values of A. A branch is created for each known value **a**j of A and labeled with that value.
- **A could be continous-valued:** in this case, the test at node N has two possible outcomes, corresponding to the conditions A<=split_point and A>split_point respectively where split_point is the split-point returned by **Attribute_selection_method**.
- **A could be discrete-valued and a binary tree must be produced:** this is the case, which is usually dictated by our used function for building tree. The test at node N is of the form "**Is A part of Sa?**" where "**Sa**" is the splitting subset for A returned by **Attribute_selection_method** as part of the splitting criterion.

All three scenarios look like this:



5. The algorithm uses the same process recursively to form a decision tree for the tuples at each resulting partition of D
6. The recursive partitioning stops **only** when any one of the following terminating conditions is true:

- all the tuples in partition D belong to the same class
- **or** there are no remaining attributes on which the tuples may be further partitioned

- **or** there are no tuples for a given branch that is, a partition of D is empty.

The computational complexity of the algorithm given training set D is O(n * |D| * log(|D|)) where:

- **n is the number of attributes (not rows!)**
- **|D| is the number of training tuples (these are the rows!)**

38. Dolovanie asociačných pravidiel

Association rule mining consists of first finding frequent itemsets satisfying a minimum support threshold from which strong association rules in the form A => B are generated. These rules also satisfy a minimum confidence threshold (which is a prespecified probability of satisfying B under the condition that A is satisfied). Associations can be further analyzed to uncover correlation rules, which convey statistical correlations between itemsets A and B.

39. Dolovanie v prúdoch dát

Stream data refer to data that flow into a system in vast volumes, change dynamically, are possibly infinite, and contain multidimensional features. Such data cannot be stored in traditional database systems. Moreover, most systems may only be able to read the stream once in sequential order. This poses great challenges for the effective mining of stream data. Substantial research has led to progress in the development of efficient methods for mining data streams, in the areas of mining frequent and sequential patterns, multidimensional analysis (this is construction of data cubes), classification, clustering, outlier analysis, and the online detection of rare events in data streams. This includes collecting information about stream data in sliding windows or tilted time windows (where the most recent data are registered at the finest granularity and the more distant data are registered at a coarser granularity), and exploring techniques like microclustering, limited aggregation, and approximation. Many applications of stream data mining can be explored - for example, real-time detection of anomalies in computer network traffic, botnets, text streams, video streams, power-grid flows, web searches, sensor networks, and cyber-physical systems.

40. Popište 2 bioinformatické problémy a jejich řešení + uveďte dolovací použité metody.

Mining Sequence data and Biological Sequences:
Biological sequences generally refer to sequences of nucleotides or amino acids (amino-kyseliny) **Biological sequence analysis** compares, aligns, indexes, and analyzes biological sequences and thus plays a crucial role in bioinformatics and modern biology.

Sequence alignment is based on the fact that all living organisms are related by evolution. This implies that the nucleotide (DNA, RNA) and protein sequences of species that are closer to each other in evolution should exhibit more similarities. An alignment is the process of lining up sequences to achieve a maximal identity level, which also expresses the degree of similarity between sequences. Two sequences are **homologous** if they share a common ancestor. The degree of similarity obtained by sequence alignment can be useful in determining the possibility of homology between two sequences. Such an alignment also helps determine the relative positions of multiple species in an evolution tree, which is called **phylogenetic tree.**

*** Pozn. neviem či vás táto téma zaujíma, ale výbornú prednášku na matfyze mala na ňu:
https://www.youtube.com/watch?v=J5txnfnH9cs
V prednáške hovorí o sekvenovani genomu
***

The problem of alignment of biological sequences can be described as:

"given two or more input biological sequences, identify similar sequences with long conserved sub-sequences"

If the number of sequences to be aligned is exactly two, the problem is called **pairwise sequence alignment**. If the number of sequences is not two, it is **multiple sequence alignment.** We recognize two kinds of alignments in a nature:

- local alignments, this means that only portions of the sequences are aligned
- global alignments, this requires alignment over the entire length of the sequences

For either nucleotides or amino acids, insertions, deletions, and substitutions occur in nature with different probabilites. **Substitution matrices** are used to represent the probabilites of substitutions of nucleotides or amino acids (amino kyseliny) and probabilities of insertions and deletions.

Algorithms to find Biological Sequences
To represent the structure or statistical regularities of sequence classes, biologists construct various probabilistic models such as:

- Markov chains (pomocou Markovovych reťazcov viete predikovať aj slová v texte, aj kadečo), in them state depends only on that of the previous state.

*** v tomto je výhoda Markovových reťazcov, pretože ak by sme na predikciu ďalšieho slova v texte využili priveľa predošlých slov strašne by sme zaťažili pamäť. Markov model využíva len pár predošlých slov ***

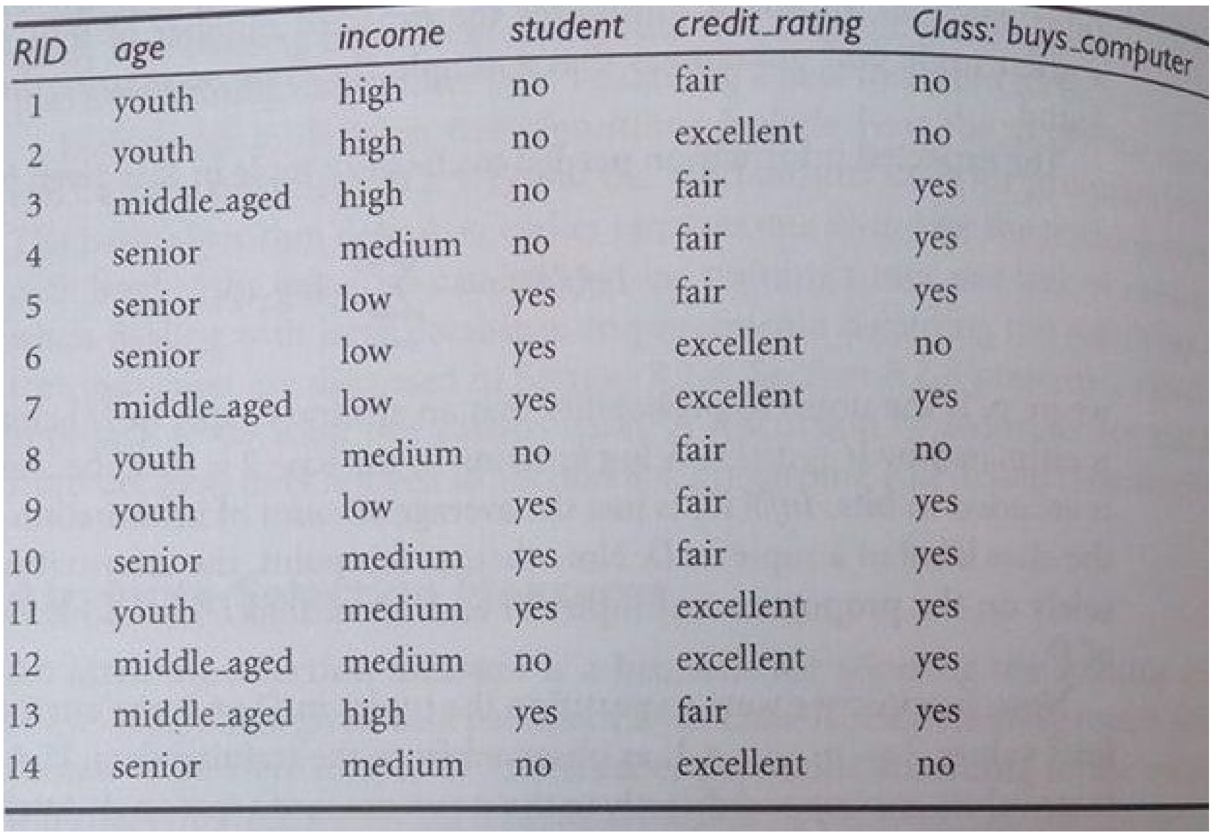
- hidden Markov chains

The most common methods for constructing hidden Markov models are:

- forward algorithm (hľadá pravdepodobnosť, že nájdeme x)
- Viterbi algorithm (hľadá najpravdepodobnejšiu cestu k cieľu v modeli)
- Baum-Welch algorithm (zvyšuje efektivitu a zlepšuje parametre predošlých modelov pri ich trénovaní)

41. Algoritmus k-means

Suppose a data set D which contains n objects in Euclidean space:

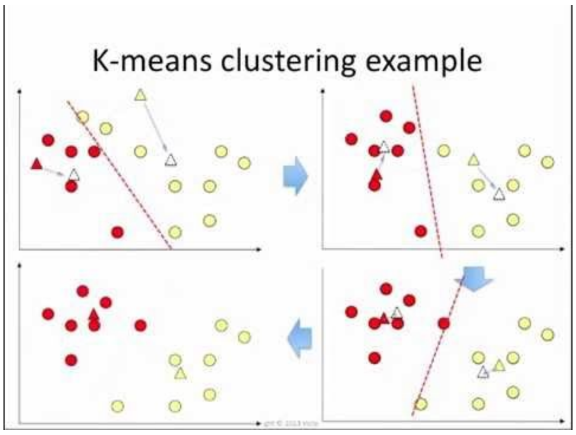


RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

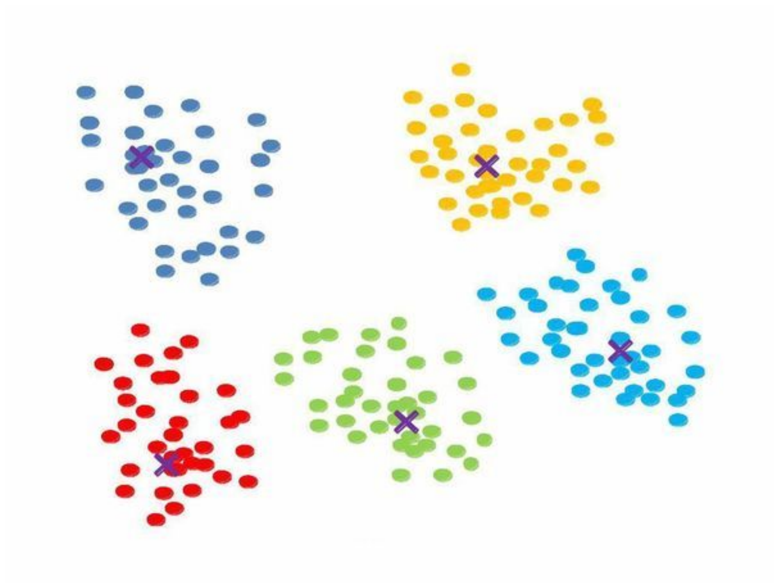
this dataset forms 6-dimensional Euclidean space. Now we use partitioning methods to distribute these objects (columns and their data) into K clusters. Every cluster will be subset of dataset D (above) and there will be no two clusters that would be equal. We can choose from these partitioning methods:

- k-means
- k-medoids

So I choose k-means. K-means has its own objective function used to assess the quality of partitioning so that objects within a cluster are similar to one another but dissimilar to objects in other clusters.



Because k-means is a centroid based, it computes centroids of clusters to represent those clusters. The centroid of a cluster is its center point and can be defined as the mean (priemer) or medoid of the objects (or points) assigned to the cluster.



Difference between some green dot and its centroid is measured by a **dist(green_dot, centroid)**, this distance is Euclidean distance between two points. The quality of green cluster can be measured by the within-cluster variation, which is the sum of squared error between all objects (all dots) in this green cluster and its centroid (x). Because we want to compute only one cluster's quality, this will be computed as follows:

E = ∑ (distance(dots, centroid))^2

Euclidean distance assumes that dots are numbers and centroid is also number. But this formula is only for one cluster, if we want quality of all clusters we write:

E = ∑ ∑ (distance(dots, centroid))^2

The k-means algorithm defines the centroid (centroids are denoted as x) of a cluster as the mean value of the points within the cluster. First, it randomly selects selects K objects from dataset D (number of columns). Because this method previously partitioned dataset, each cluster probably stands for different column. So it will choose all of the clusters we have computed. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the Euclidean distance between the object and the cluster mean. So if we would have value = 100 in a column called price_in_eur and in some cluster would be the prices 200, 300, 100, 500, and so on it will assign our price = 100 to this cluster. But we have changed the cluster, added a new item to it. K-means will compute new mean using the objects assigned to this cluster in the previous (this) iteration.

42. Rozdiel medzi klasifikáciou a predikciou

Classification is a form of data analysis that extracts models describing important data classes. Such models called classifiers predict categorical (discrete, unordered) class labels. For example, we can build a classification model to categorize bank loan applications as either safe or risky. Such analysis can help provide us with a better understanding of the data at large. Many classification methods have been proposed by researchers in machine learning, pattern recognition, and statistics. Most algorithms are memory resident, typically assuming a small data size. Recent data mining research has built on such work, developing scalable classification and prediction techniques capable of handling large amounts of disk-resident data. Classification has numerous applications, including fraud detection, target marketing, performance prediction, manufacturing, and medical diagnosis.

Suppose these examples:

- bank loans officer needs analysis of her data to learn which loan applicants are safe and which are risky for the bank.
- marketing manager needs data analysis to help guess whether a customer with a given profile will buy a new computer

In each of these examples the data analysis task is classification, where a model or classifier is constructed to predict class labels such as “safe” or “risky” for the loan application data, “yes” or “no” for the marketing data. These categories can be represented by a discrete values, where the ordering among values has no meaning.

Suppose that the marketing manager wants to predit how much a given customer will spend during a sale at out shop. This data analysis task is an example of numeric prediction, where the model constructed predicts a continuous-valued function, or ordered value as opposed to a class label where ordering has no meaning. This model is a predictor.

Fázy klasifikácie a predikcie
In the first step of a classification, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step, where a classification algorithm builds the classifier by analyzing or learning from a training set made up of database tuples and their associated class labels.

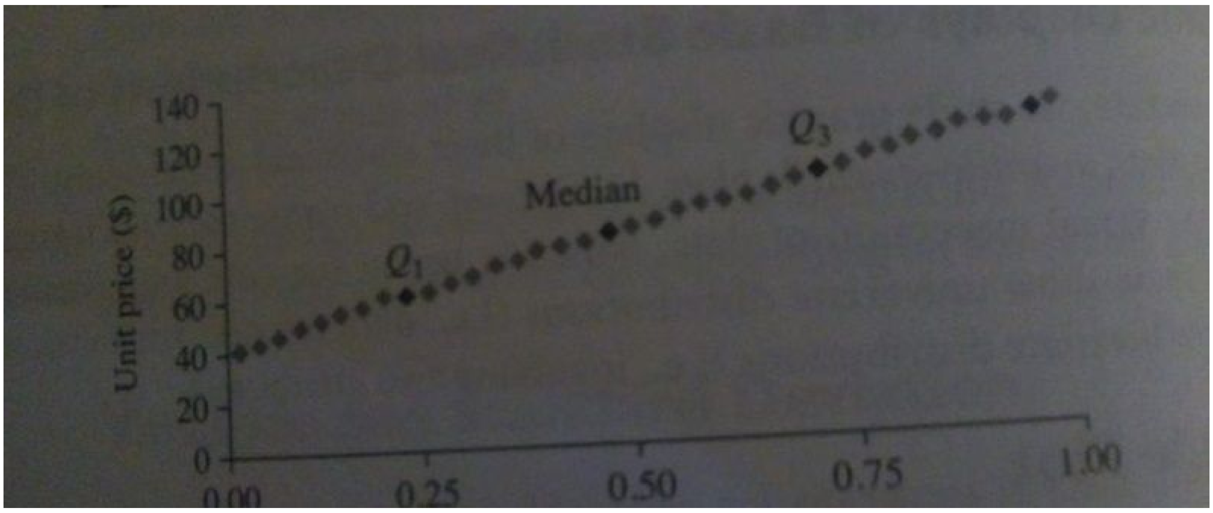
The data classification process consists of:

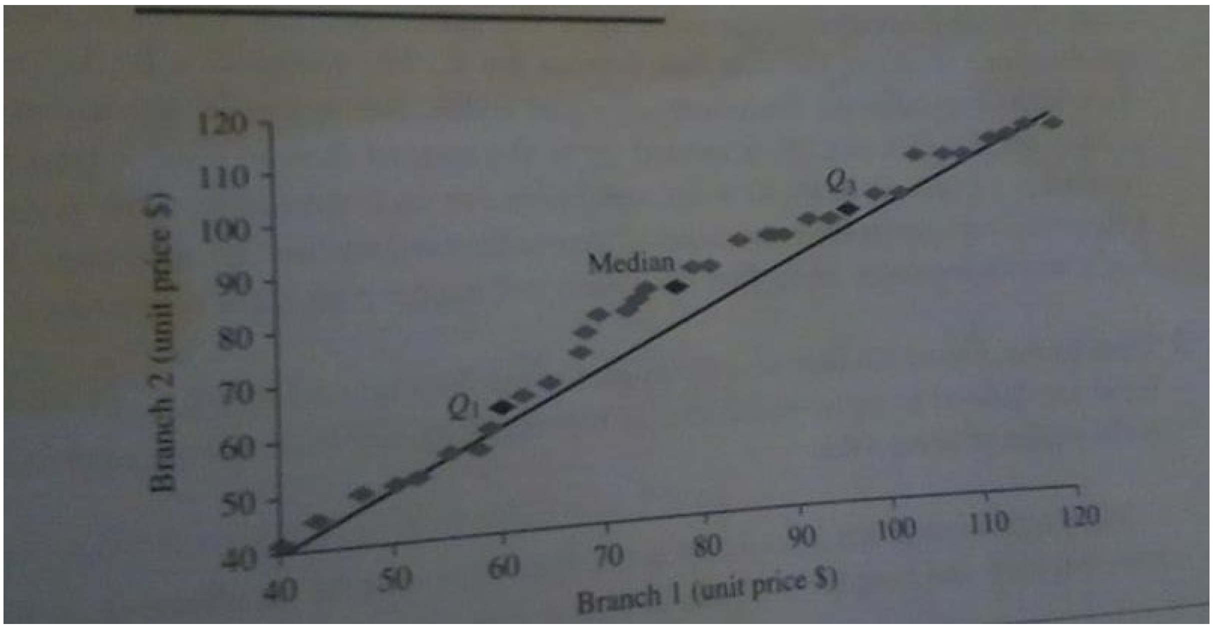
- Learning: training data are analyzed by a classification algorithm
- Classification: test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples/rows.

43. Grafy na základnú štatistickú analýzu

A quantile plot is a simple and effective way to have a first look at a univariate data distribution. First, it displays all of the data for the given attribute. Second it plots quantile information. Let xi for i = 1 to N be the data sorted in increasing order so that x1 is the smallest observation and xn is the largest for some ordinal or numeric attribute X. Each observation xi is paired with a percentage Fi, which indicates that approximately Fi x 100 % of the data are below the value xi. We say approximately because there may not be a value with exactly a fraction Fi of the data below xi.

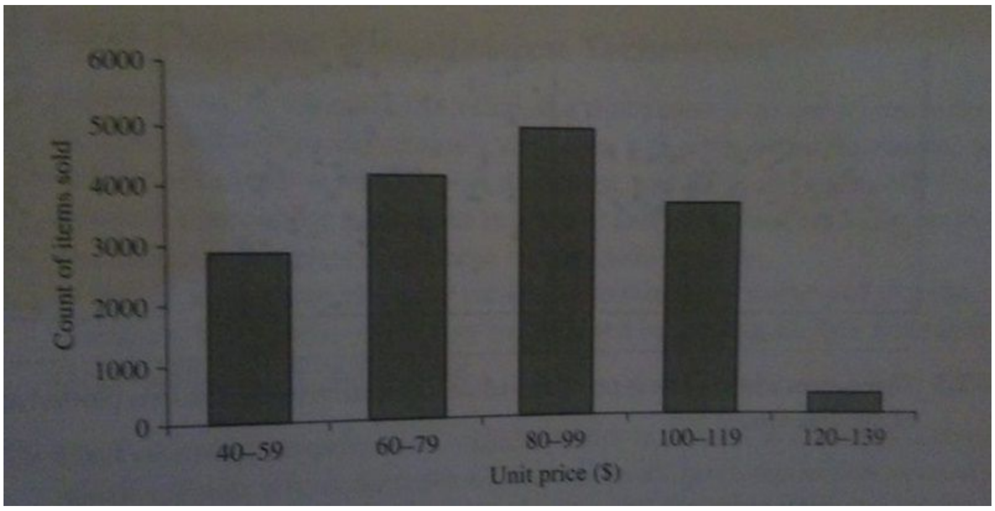
A quantile-quantile plot (Q-Q plot) grapsh the quantiles of one univariate distribution against the corresponding quantiles of another. It is a powerful visualization tool in that it allows the user to view whether there is a shift in going from one distribution to another. Suppose that we have two sets of observations for the attribute or variable unit_price taken from two different branch locations. Let x1,...,xn be the data from the first branch and let y1,...,ym be the data from the second location, where each data set is sorted in increasing order. If their length is equal (so M=N), then we simply plot yi against xi where yi and xi are both (i-0.5)/N quantiles of their respective data sets. If M < N (the second branch has fewer observations than the first) there can be only M points on the q-q plot. This is the Q-Qplot



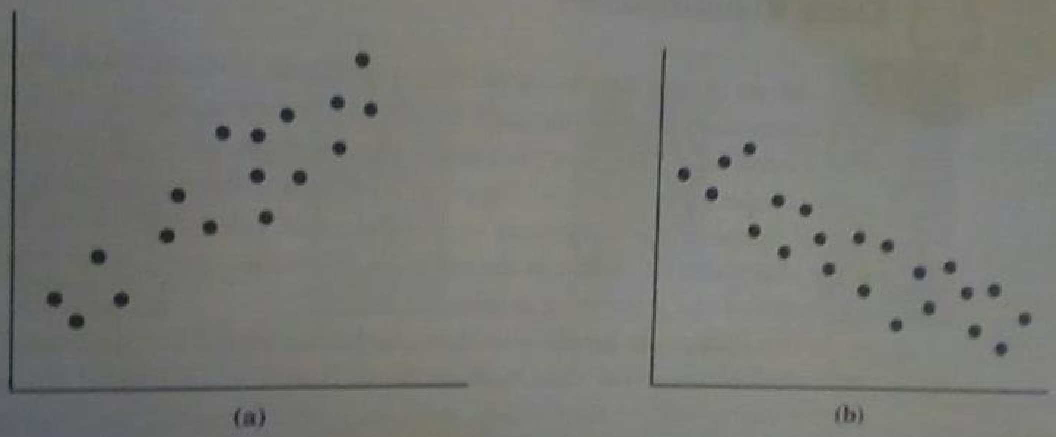


Picture above this text shows a quantile-quantile plot for unit price data of items sold at two branches of some e-shop during a given time period. Each point corresponds to the same quantile for each data set and shows the unit price of items sold at branch 1 versus branch 2 (branch 2 is on axis Y and branch 1 is on the axis X). To aid in comparison the straight line represents the case where, for each given quantile, the unit price at each branch is the same. The darker points correspond to the data for Q1, the median, and Q3 respectively(Q1 is first quartile and Q3 last we always compute).

Histograms (or frequency histograms) are at least a century old and are widely used. “Histos” mean pole or mast, and “gram” means chart, so a histogram is a chart of poles. Plotting histograms is a graphical method for summarizing the distribution of a given attribute X. If X is nominal, such as automobile_model or item_type then a pole or vertical bar is drawn for each known value of X. The height of the bar indicates the frequency. If X is numeric, the term histogram is preferred. The range of values for numeric X is partitioned into disjoint consecutive subranges. The subranges referred to as buckets or bins are disjoint subsets of the data distribution X. Histogram:



A scatter plot is one of the most effective graphical methods for determining if there appears to be a relationship, pattern, or trend between two numeric attrributes. To construct a scatter plot each pair of values is treated as a pair of coordinates in an algebraic sense and plotted as points in the plane. Two attributes X and Y are correlated, if one attribute implies the other. Correlations can be positive, negative or null (uncorrelated). This picture shows those examples (on the left is positive correlation an on the right side is negative correlation):



ure 2.8 Scatter plots can be used to find (a) positive or (b) negative correlations between attributes.

44. Dolovanie v texte

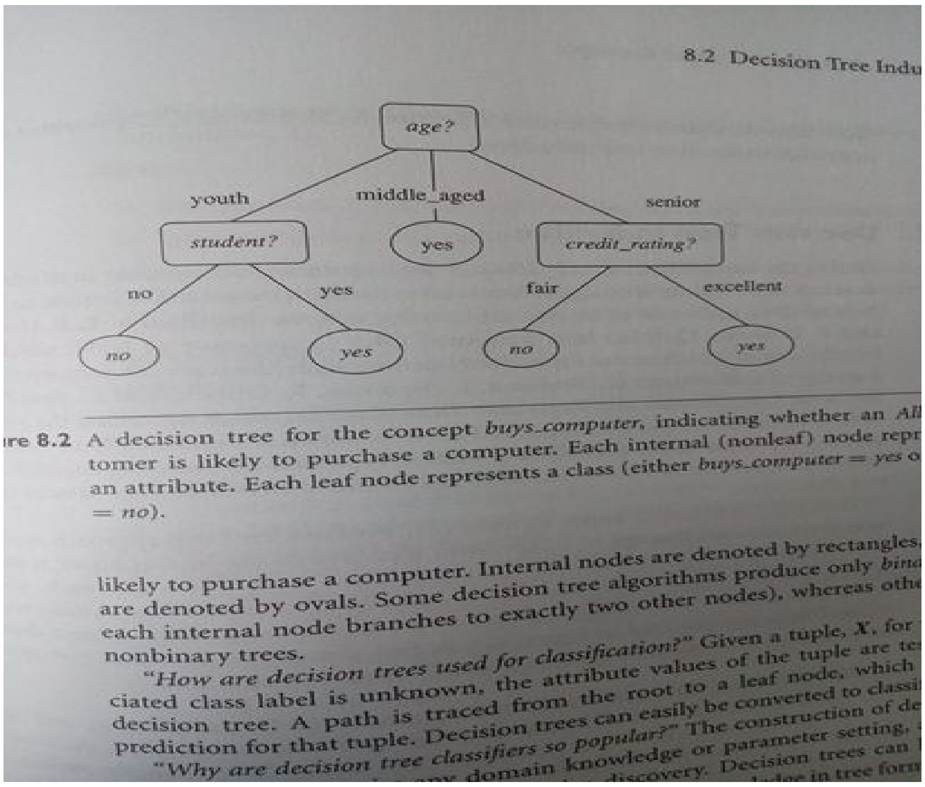
Text mining is an interdisciplinary field that draws on information retrieval, data mining, learning, statistics, and computational linguistics. A substantial portion of information is stored as text such as news, articles, technical papers, books, digital libraries, email messages, blogs and web pages. Hence, research in text mining has been very active. An important goal is to derive high-quality information from text. This is typically done through the discovery of patterns and trends by means such as statistical pattern learning, topic modelling, and statistical language modelling. Text mining usually requires structuring the input text. This is followed by deriving patterns within the structured data, and evaluation and interpretation of the output. High quality in text mining usually refers to a combination of relevance, novelty, and interestingness.

Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity-relation modeling. Other examples include multilingual data mining, multidimensional text analysis, contextual text mining, and trust and evolution analysis in text data, as well as text mining applications in security, biomedical literature analysis, online media analysis, and analytical customer relationship management. For these tasks we can use WordNet or SemanticWeb.

45. Extrakcia pravidla z rozhodovacieho stromu

Decision tree classifiers are a popular method of classification. Decision trees can become large and difficult to interpret. In comparison with a decision tree, the IF-THEN rules may be easier for humans to understand, particularly if the decision tree is very large. To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each splitting criterion along a given path is logically ANDed to form the rule antecedent ("IF" part). The leaf node holds the class prediction, forming the rule consequent ("THEN" part).

Example
Following decision tree


8.2 Decision Tree Indu
age?
youth
middle_aged
senior
student?
yes
credit_rating?
no
yes
fair
excellent
no
yes
no
yes

ıre 8.2 A decision tree for the concept *buys_computer*, indicating whether an *Al*
tomer is likely to purchase a computer. Each internal (nonleaf) node repr
an attribute. Each leaf node represents a class (either *buys_computer* = *yes* o
= *no*).

likely to purchase a computer. Internal nodes are denoted by rectangles,
are denoted by ovals. Some decision tree algorithms produce only *bina*
each internal node branches to exactly two other nodes), whereas othe
nonbinary trees.
"*How are decision trees used for classification?*" Given a tuple, *X*, for
ciated class label is unknown, the attribute values of the tuple are te
decision tree. A path is traced from the root to a leaf node, which
prediction for that tuple. Decision trees can easily be converted to classi
"*Why are decision tree classifiers so popular?*" The construction of de
domain knowledge or parameter setting,
discovery. Decision trees can
in tree form

can be converted to classification IF-THEN rules by tracing the path from the root node to each leaf node in the tree. The rules extracted from this graph are as follows:
Rule1: IF age = youth AND student=no THEN buys_computer=no
Rule2: IF age = youth AND student = yes THEN buys_computer = yes
Rule3: IF age = middle_aged THEN buys_computer=yes
Rule4: IF age = senior AND credit_rating = excellent THEN buys_computer=yes
Rule5: IF age = senior AND credit_rating = fair THEN buys_computer=no

A disjunction (OR) is implied between each of the extracted rules:
Rule1 OR Rule2 OR Rule3 OR Rule4 OR Rule5.
Because the rules are extracted directly from the tree, they are mutually exclusive and exhaustive.

Mutually exclusive means that we cannot have rule conflicts here because no two rules will be triggered for the same tuple. It’s because we have one rule per leaf, and any tuple/row can map to only one leaf).
Exhaustive means there is one rule for each possible attribute-value combination, so that this set of rules does not require a default rule. Therefore, the order of the rules does not matter - they are unordered. They are exhaustive, because these rules were extracted from this table (look, they have all the attributes):

Age	Student	Credit Rating
Youth	Yes	Fair
Middle aged	Yes	Fair
Senior	no	Excellent

But we have one problem. Since we end up with one rule per leaf, the set of extracted rules is not much simpler than the corresponding decision tree. The extracted rules may be even more difficult to interpret than the original trees in some cases. Although it is easy to extract rules from a decision tree, we may need to do some more work by pruning the resulting rule set. For a given antecedent any condition that does not improve the estimated accuracy of the rule can be pruned (pruned means removed), thereby generalizing the rule. **C4.5 is a method** which extracts rules from an unpruned tree, and then prunes the rules using a pessimistic approach similar to its tree pruning method. The training tuples and their associated class labels (values of the classes, so buys_computer is a class and yes is its value) are used to estimate rule accuracy. However, because this would result in an optimistic estimate, alternatively, the estimate is adjusted to compensate for the bias, resulting in a pessimistic estimate. In addition, any rule that does not contribute to the overall accuracy of the entire rule set can also be pruned.

46. AdaBoost

Suppose that we are a patients and have certain symptoms. Instead of consulting one doctor, we choose to consult several. Suppose we assign weights to the values or worth of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made. The final diagnosis for us is then a combination of the weighted diagnoses.

In boosting, weights are also assigned to each training tuple/row. A series of K classifiers (models) is iteratively learned. After a classifier M with some index is learned, the weights are updated to allow the subsequent classifier M with index (i+1) to "pay more attention" to the training tuples that were misclassified by previous M. The final boosted classifier M* combines the votes of each individual classifier (model we prepared for a classification task), where the weights of each classifier's vote is a function of its accuracy.

AdaBoost (short for Adaptive Boosting) is an algorithm which works as follows:
suppose we want to boost the accuracy of a learning method. We are given a dataset **D** which contains of ***d*** class-labeled tuples. This means, that each row in a dataset has assigned column with a class (value) that is not empty. We have tuples X and its associated class values:
(X1, y1), (X2, y2), …, (X***d***,y***d***) where "y" is the class value of tuple X
Initially AdaBoost assigns each training tuple X an equal weight of (1/***d***). Generating K classifiers for the ensemble requires K rounds through the rest of the algorithm. In round ***i*** the tuples from dataset **D** are sampled to form a training set **D**i of size ***d*** . Sampling with replacement is used - the same tuple may be selected more than once. Each tuple's chance of being selected is based on its weight (1/***d***). A classifier model ***Mi*** is derived from the training tuples of **D**i. Its error is then calculated using **D**i as a test set. The weights of the training tuples are then adjusted according to how they were classified. If a tuple was incorrectly classified, its weight is increased. If a tuple was correctly classified, its weight is decreased. A tuple's weight reflects how difficult it is to classify - **the higher the weight, the more often it has been misclassified.** These weights will be used to generate the training samples for the classifier of the next round (i+1). **The basic idea is that when we build a classifier, we want it to focus more on the misclassified tuples of the previous round.** Some classifiers may be better at classifying some difficult tuples than others.

Calculating AdaBoost Step by Step

- to compute the error rate of model ***Mi*** we sum the weights of each of the tuples in **D**i that ***Mi*** misclassified. So:

error(of_model_Mi) = ∑ weight***err(X**j**)**
where **err(X**j**)** is the misclassification error of tuple **X**j

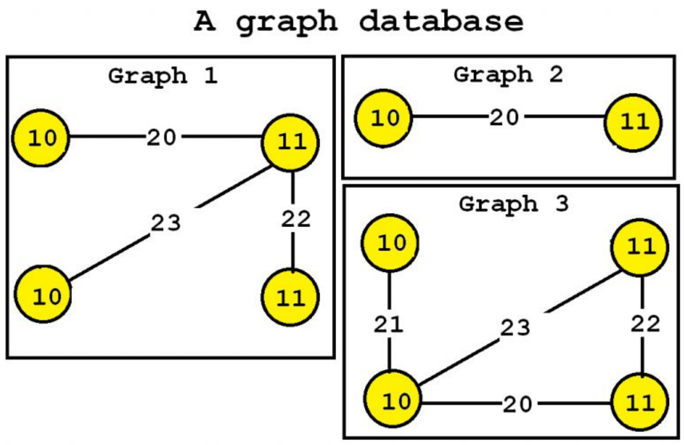
- to compute the weight of classifier we have to know basic principle of AdaBoost. Unlike bagging (it is a method for improving accuracy) where each classifier was assigned an equal vote, boosting assigns a weight to each classifier's vote, based on how well the classifier performed. The lower a classifier's error rate, the more accurate it is, therefore the higher its weight for voting should be. The weight of classifier ***Mi***'s vote is:
  log ((1-error(of_model_Mi))/error(of_model_Mi))

47. Dolovanie z grafov

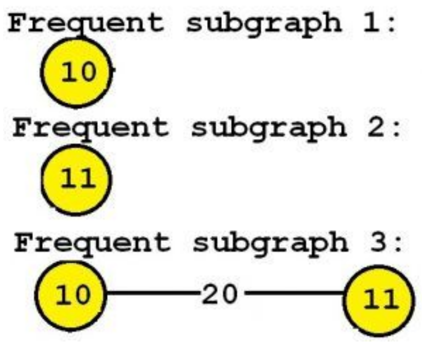
Graph represents a more general class of structures than sets (sets su v anglictine mnoziny), sequences, lattices (to sú také tabuľky podobné maticiam v matike) and trees. Graph pattern mining is the mining of frequent subgraphs in one or a set of graphs. Methods for mining graph patterns can be categorized into these approaches:

1. Apriori-based
2. pattern growth-based

Alternatively, we can mine the set of closed graphs where a graph **g** is closed if there exists no proper supergraph **g'** that carries the same support count as **g.**
In order to find frequent subgraph we need to have graph database:



Task is as follows:
find all subgraphs from a given graph database D that appear at least in 3 graphs.
Solution:



Because the definition of closed graphs says, that we need to compute support count of graph **g**, our three subgraphs have these support counts (or frequencies):

- first frequent subgraph has frequency/support count equal to 3 because node with a label 10 appears in all of the graphs in our dataset
- second frequent subgraph has support count equal to 3, because node with a label 11 appears in all three graphs of our dataset
- third frequent subgraph has support count equal to 3, because its sequence of nodes appears identically in all of three graphs in a dataset

Definition:
"a graph **g** is closed if there exists no proper supergraph **g'** that carries the same support count as **g"**

So in our example we have NO closed graphs, because all three subgraphs have the same support count.

48. Všetky otázky ohľadom OLAP

Please, do not confuse OLAP and OLTP. Because most people are familiar with commercial relational database systems, it is easy to understand what a data warehouse is by comparing OLTP and OLAP.

The major task of online operational database systems is to perform online transaction and query processing. These systems are called **online transaction processing - OLTP.** They cover most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting.

On the other hand, data warehouse systems serve users or knowledge workers (po slovensky povedané znalostní inžinieri, mali sme to na Umelej inteligencii) in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accomodate the diverse needs of different users. These systems are known as **online analytical processing - OLAP**.

The major distinguishing features between **OLAP and OLTP** are:

- **users and system orientation:** an OLTP system is customer-oriented and is used for transaction and query processing by clerks, clients, and information technology professionals. An OLAP system is market-oriented and is used for data analysis by knowledge workers, including managers, executives, and analysts.
- **data contents:** An OLTP system manages current data that typically are too detailed to be easily used for decision making. An OLAP system manages large amounts of historic data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity.
- **database design:** An OLTP system usually adopts an entity-relationship data model and an application-oriented database design. An OLAP system typically adopts either a star or snowflake model and a subject-oriented database design.
- **view:** An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historic data or data in different organizations. In contrast an OLAP system often spans multiple versions of a database schema, due to the evolutionary process of an organization.
- **access patterns:** The access patterns of an OLTP system consist mainly of short, atomic transactions. However, accesses to OLAP systems are mostly read-only operations although many could be complex queries.

Differences between OLAP, ROLAP, MOLAP and HOLAP

Relational OLAP servers are the intermediate servers that stand in between a relational back-end server and client front-end tools. They use a relational or extended-relational DBMS to store and manage warehouse data and OLAP middleware to support missing pieces. ROLAP servers are optimized for each back-end, they are scalable. Example of ROLAP is Microstrategy: https://www.microstrategy.com/us

Multidimensional OLAP (MOLAP) servers support multidimensional data views through array-based multidimensional storage engines. They map multidimensional views directly to data cube array structures. The advantage of using a data cube is that it allows fast indexing to precomputed summarized data.

Hybrid OLAP (HOLAP) servers combines ROLAP and MOLAP technologies, benefiting from the greater scalability of ROLAP and the faster computation of MOLAP. Examples of hybrid OLAP are: https://www.microsoft.com/en-us/download/details.aspx?id=22661

49. Otázka týkajúca sa metódy hierarchického zhlukovania

While partitioning methods (k-means, k-medoids) meet the basic clustering requirement of organizing a set of objects into a number of exclusive groups, in some situations we may want to partition our data into groups at different levels such as in a hierarchy. A hierarchical clustering method works by grouping data objects into a hierarchy or tree of clusters.

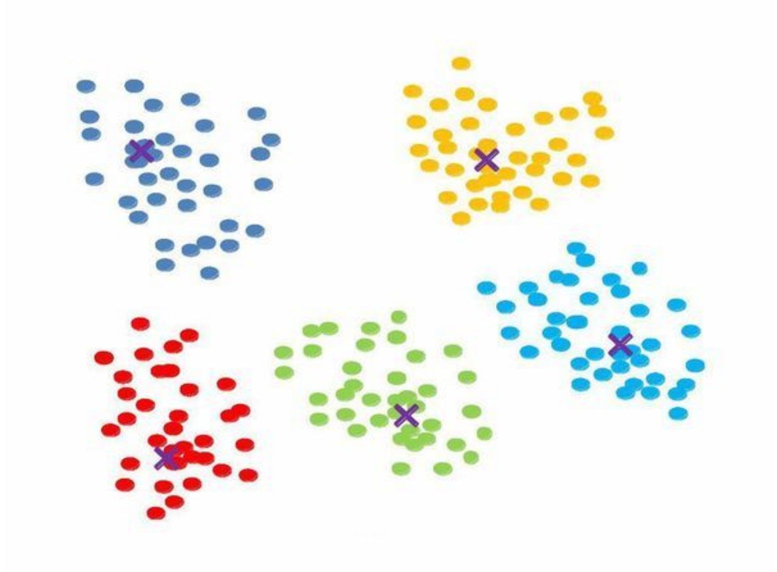
Examples of using hierarchical clustering method:

- as the manager of human resources at some shop we may organize our employees into major groups such as executives, managers, and staff. We can further partition these groups into smaller subgroups. For instance, the general group of staff can be further divided into subgroups of senior officers, officers, and trainees. All these groups form a hierarchy. We can easily summarize or characterize the data that are organized into a hierarchy, which can be used to find, say, the average salary of managers and of officers
- consider handwritten character recognition as another example. A set of handwriting samples may be first partitioned into general groups where each group corresponds to a unique character. Some groups can be further partitioned into subgroups since a character may be written in multiple substantially different ways. If necessary, the hierarchical partitioning can be continued recursively until a desired granularity is reached.

Computing BIRCH, Chameleon and Probabilistic Hierarchical Clustering as hierarchical methods

1. BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

is designed for clustering a large amount of numeric data by integrating hierarchical clustering and other clustering methods such as iterative partitioning. It overcomes the two difficulties in agglomerative clustering methods, for first **scalability** and for second **the inability to undo what was done in the previous step.** BIRCH uses for speeding up clustering feature (to summarize its clusters) and clustering feature tree (CF-tree) to represent a cluster hierarchy. BIRCH can be also used for streaming data and offers good speed and scalability especially in large datasets. Consider these clusters and pay attention to green cluster:



BIRCH assume that green cluster is made of N data objects where each is D-dimensional. So N is a number of rows in a dataset, each row is an object and each row has so much dimensions as it has columns. **Clustering feature of green cluster** is a 3-dimensional vector summarizing information about this cluster and its objects. It is defined as:

CF = <number_of_rows, linear_sum_of_green_points, square_sum_of_green_points)
where:
number_of_rows -> in a dataset
linear_sum_of_green_points -> ∑ (green_point)
square_sum_of_green_points -> ∑ (green_point)^2

Centroid (depicted as x) of green cluster is computed as:
centroid=∑ (green_point) /NROW(green_cluster) = linear_sum_of_green_points / NROW(green_cluster)

Radius of green cluster is computed as:
radius = sqrt(∑ (green_point - centroid_of_green_cluster) / NROW(green_cluster))

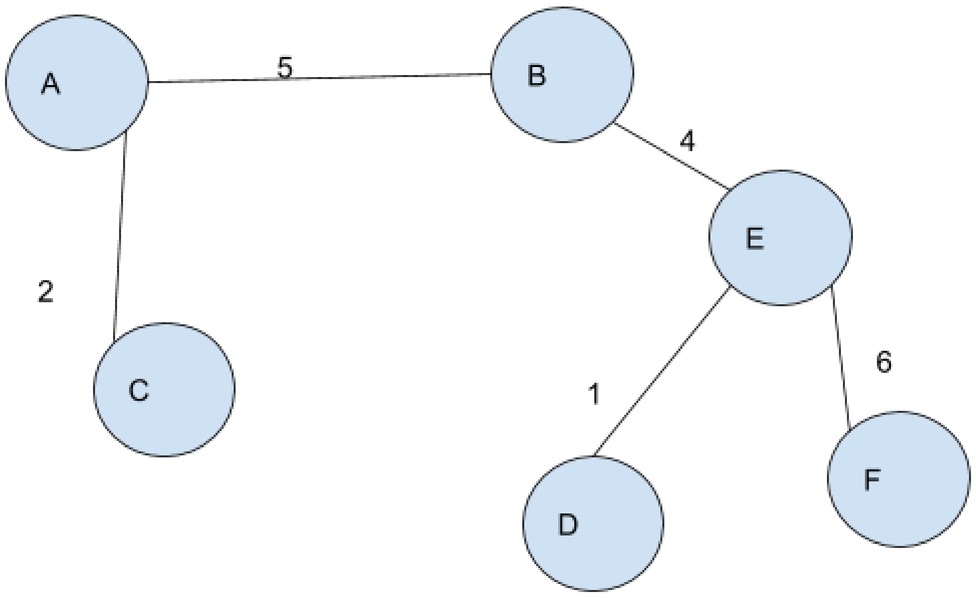
Average pairwise distance within a green cluster called diameter D is computed:
diameter = sqrt(((2*NROW(green_cluster)*((square_sum_of_green_points)^2)) - 2*(linear_sum_of_green_points)*((square_sum_of_green_points)^2)) / (NROW(green_cluster)*(NROW(green_cluster)-1))

Why is all of this computed? Because it avoids storing the detailed information about individual objects or points of our green cluster. **This is the key to BIRCH efficiency in allocating space.**

2. Chameleon: Multiphase Hierarchical Clustering Using Dynamic Modeling
Chameleon is a hierarchical clustering algorithm that uses dynamic modeling to determine the similarity between pairs of clusters. In Chameleon, cluster similarity is assessed based on:

- how well connected objects are within a cluster
- the proximity of clusters

Chameleon merges two clusters only if their interconnectivity which is computed as:
suppose this graph



then you have to know what is edge cut. Edge cut is the weight of the edges (numbers above them) which wants Chameleon minimize. It does so, because he wants to partition this graph into smaller graphs. If our graph would be cluster, then this cluster would be cut to relatively small subclusters. Chameleon uses graph partitioning algorithm and agglomerative hierarchical clustering algorithm to merge subclusters back based on their similarity.

Example
Chameleon wants to cut off the cluster (as a whole) in a above picture. Suppose that this cluster is made of two sub-clasters (subgraphs) (A,B,C) and (E,D,F). To cut them off he must cut edge between B and E. This edge has weight = 4. Absolute value of 4 is also 4. Because Chameleon wants to cut also these two subgraphs it does so by cutting edges between A and C, and cutting edges between E and D. Relative interconnectivity of the resulted graph would be:
RI = |4| / ((1/2)*(|2| + |1|))
but Chameleon wants to know also relative closeness so it will compute it:
-average weight of the edges that connect vertices between (A, B, C) and (E, D, F) is equal to weight between nodes B and E (it is 4/1=4)
- average weight of the edges that had to be cut in order to split 2 subgraphs are: for (A, B, C) it is 2 and for (E, D, F) it is 1
- numbers from the previous step will be multiplied by **|weight=2| / (|weight=2| + |weight=2|)**
and **|weight=1| / (|weight=1| + |weight=1|)** respectively
so:
RC= 4 / ((0,5*2) + (0,5*2)) = 2

*** ďalej to vysvetlené nie je, neviem čo potom robí***

50. Vysvetlenie dolovania frekventovaných vzorov a asociácií medzi nimi

Frequent patterns are patterns that appear frequently in a data set. They are mined from transactional data, transactional data is made of transactions. These transactions are rows, each row consist of some value in some column. If for example a set of items such as milk and bread appear frequently together in some rows in a transactional data set, then it is probably frequent itemset. A subsequence of frequent pattern is called frequent sequential pattern. We recognize these substructures of frequent patterns:

- subgraphs
- subtrees
- sublattices

If a substructure occurs frequently, it is called a frequent structured pattern.

Suppose an example of a computer store. It has a set of items and each item has a Boolean variable representing the presence or absence of that item. So each customer basket can be represented by a Boolean vector of values assigned to these variables. Association rules would be Boolean vectors, which contain those items which are purchased together.

Rule has a:

- support
- confidence

So if we have next rule:
computer => antivirus_software[support=2%,confidence=60%]
then support=2% means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together. A confidence=60% means that 60% of the customers who purchased a computer also bought the software. Minimum support threshold is a minimum frequency of some pattern which is usually set by a user or domain expert.

Mining association rules is two-step process:

- finding all frequent itemsets which occur at least as frequently as a predetermined minimum support count
- generating strong association rules from the frequent itemsets which must satisfy minimum support and minimum confidence

TID	Item_IDs
T100	I1,I2,I5,I2,I2,I2,I2,I4
T200	I2,I4,I1,I1,I1,I1,I4,I4,I3

T300	I2,I3,I1,I3,I3,I3,I3,I1,I3

1. step of Apriori algorithm - scan table above for count of each candidate

Itemset	Support count
I1	7
I2	7
I3	7
I4	4
I5	1

2. Compare those candidate's support count (above) with minimum support count. Domain expert says, that minimum support count is 2. We get:

Itemset	Support count
I1	7
I2	7
I3	7
I4	4

because I5 has support count only 1. Now generate next candidates from the first column (from left) in a table above. Also scan database for count of each candidate. You will get:

Itemset	Support count
I1,I2	3
I1,I3	3
I1,I4	4
I2,I3	2
I2,I4	2
I3,I4	1

But {I3, I4} has support count only 1 then:

Itemset	Support count
I1,I2	3
I1,I3	3
I1,I4	4
I2,I3	2
I2,I4	2

Do first_column x first_column and you will get:

Itemset	Support count
I1,I2,I3	2
I1,I2,I4	2

all the rows remains and we will generate next candidates:

Itemset	Support count
I1,I2,I3,I4	1!!!!

The join results in R={I1, I2, I3, I4} so pruning starts:

1. we need to find length(R)-1 subsets of previous set:

they are {I1, I2,I3}, {I2,I3,I4}, {I2,I4,I1} and they have frequencies:
{I1, I2, I3} -> after looking at the table of 3-itemsets I found exactly 1 match
{I2,I3,I4}-> after looking at the table of 3-itemsets I have no found any match
{I2,I4,I1} -> one match
so we have found that at least one subset of the subsets above is not frequent and Apriori terminates, having found all of the frequent itemsets.

Because I previously generated all of the possible subsets of {I1,I2,I3,I4} I will generate for them association rules:
{I1, I2, I3} => I4 and its confidence is ¼ = 0.25*100 = 25
{I2,I3,I4} => I1 and its confidence is 1/7 = 0.1428*100 =14,28
{I2,I4,I1} => I3 and its confidence is 1/7 = 0.1428*100 = 14,28

Domain expert says that minimum confidence threshold is 20 % so only first rule satisfies. This rule is strong rule.

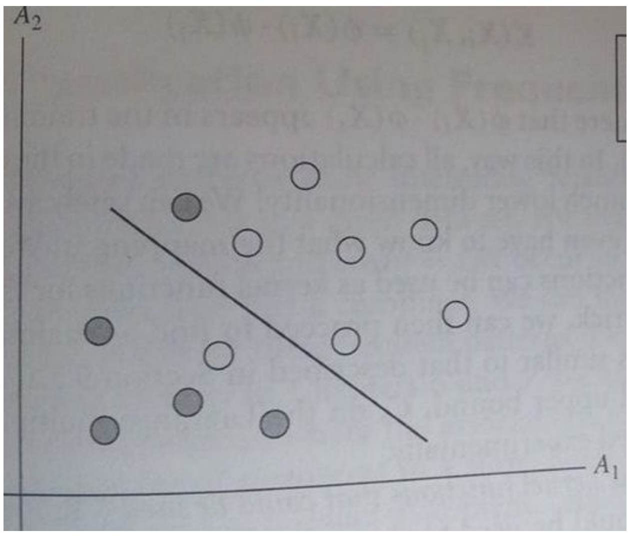
51. Nelineárne SVM

Nonlinear data are data which could not be linearly separable by hyperplane. I have to go through these steps:

- transform the original input data into a higher dimensional space using a nonlinear mapping
- search for a linear separating hyperplane in the new space

Table of Linearly Inseparable Data			
M	X	R	Class Labels Y
M1 = 1	X1 = 2	R1=4	Y1 = +1 **or buys_computer = yes**
M2 = 3	X2 = 4	R2=5	Y2 = -1 **or buys_computer = no**

Non-linear data:

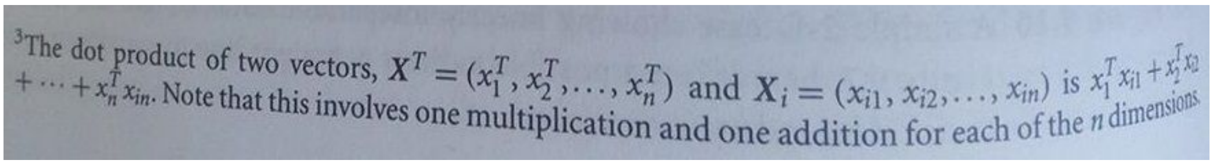


We have 3-D input vector A = (M, X, R) which will be mapped to 2*(3-D)=6-D space Z using these functions:

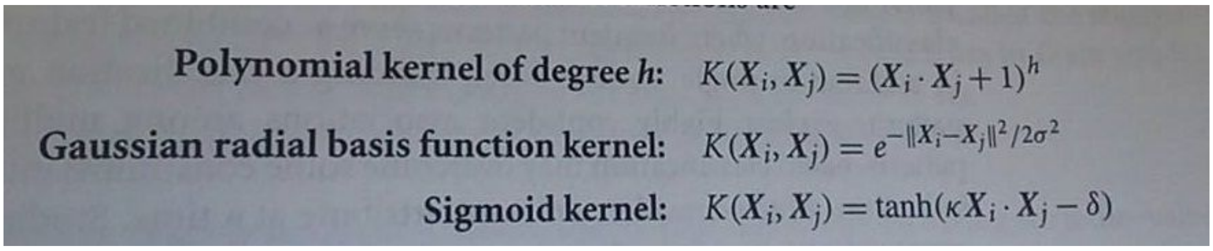
- *map_1(A) = M*
- *map_2(A) = X*
- *map_3(A)= R*
- *map_4(A) = (M)^2*
- *map_5(A) = M*X*
- *map_6(A)=M*R*

hyperplane in a 6-D space:
weight vector W=(1,1,1,1,1,1) = (w1,w2,w3,w4,w5,w6)
d(A) = w1*M + w2*X + w3*R + w4(M)^2+w5*M*X+W6*M*R + b **because**
d(A) =
w1*(map_1(A))+w2*(map_2(A))+w3*(map_3(A))+w4*(map_4(A))+w5*(map_5(A))+w6*(map_6(A)) + b

Replace above formula by Kernel function only where is var*var so:
w5*M*X=w5*(map_1(A)*map_2(A))
dot product = (map_1(A)*map_2(A)) because


[3]The dot product of two vectors, $X^T = (x_1^T, x_2^T, \ldots, x_n^T)$ and $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ is $x_1^T x_{i1} + x_2^T x_{i2} + \cdots + x_n^T x_{in}$. Note that this involves one multiplication and one addition for each of the n dimensions.

replace dot product by Kernel function. I chose first:


Polynomial kernel of degree h: $K(X_i, X_j) = (X_i \cdot X_j + 1)^h$

Gaussian radial basis function kernel: $K(X_i, X_j) = e^{-\|X_i - X_j\|^2/2\sigma^2}$

Sigmoid kernel: $K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$

so Kernel function for w5*M*X will be (1*2 + 1)^6 where 6 is a degree. Do this also for W6*M*R.

52. K-means

Suppose this dataset:

Human	Height	Weight
A	185	80
B	170	74
C	168	60
D	171	75

Randomly choose two rows as a two mean vectors. Those mean vectors represent dots in a 2-D space, they are centroids for the future clusters.

Human	Height	Weight
A	185	80
B	170	74

Mean Vector A -> A=(185,80) #centroid
Mean Vector B -> B=(170,74) #centroid

Compute Euclidean distances between each of the mean vectors and all points other that I chose:

Human	1	2
A	((185-185)^2 + (80-80)^2)^(½) = 0	((170-185)^2 + (74-80)^2)^(½) = 16,155
B	((185-170)^2 + (80-74)^2)^(½) = 16,155	((170-170)^2 + (74-74)^2)^(½) = 0
C	((185-168)^2 + (80-60)^2)^(½) = 26,248	((170-168)^2 + (74-60)^2)^(½) = 14,142
D	((185-171)^2 + (80-75)^2)^(½) = 14,866	((170-171)^2 + (74-75)^2)^(½) = 1,414

Human	1	2	Cluster
A	((185-185)^2 + (80-80)^2)^(½) = 0	((185-170)^2 + (80-74)^2)^(½) = 16,155	**1. cluster,** because 0 < 16,155 and 0 is in column 1
B	((185-170)^2 + (80-74)^2)^(½) = 16,155	((170-170)^2 + (74-74)^2)^(½) = 0	**2. cluster**, because 0 < 16,155 and 0 is in column 2
C	((185-168)^2 + (80-60)^2)^(½) = 26,248	((170-168)^2 + (74-60)^2)^(½) = 14,142	**2. cluster**, because 14,142 < 26,248 and 14,142 is in column 2
D	((185-171)^2 + (80-75)^2)^(½) = 14,866	((171-170)^2 + (75-74)^2)^(½) = 1,414	**2. cluster**, because 1,414 < 14,866 and 1,414 is in column 2

Now calculate new centroids, because we filled our clusters:

Centroid for Cluster 1	{(1/1)*(185),(1/1)*(80)}	c(185,80)
Centroid for Cluster 2	{(1/3)*(170+168+171),(1/3)*(74+60+75)}	c(169,666;69,666)

Human	1	2	Cluster
A	((185-185)^2 + (80-80)^2)^(½) = 0	((185-169,666)^2 + (80-69,666)^2)^(½) = 18,491	**1. cluster** because 0 < 18,491 and 0 is in 1. column
B	((185-170)^2 + (80-74)^2)^(½) = 16,155	((170-169,666)^2 + (74-69,666)^2)^(½) = 4,346	**2. cluster** because 4,346 < 16,155 and 4,346 is in 2. column
C	((185-168)^2 + (80-60)^2)^(½) = 26,248	((169,666-168)^2 + (69,666-60)^2)^(½) = 9,808	**2. cluster** because 9,808 < 26,248 and 9,808 is in 2. column
D	((185-171)^2 + (80-75)^2)^(½) = 14,866	((171-169,666)^2 + (75-69,666)^2)^(½) = 5,498	**2. cluster** because 5,498 < 14,866 and 5,498 is in 2. column

53. Porterov algoritmus na identifikovanie koreňov slov pri dolovaní dát z webu