

Príprava na vnútrosemestrálku z IB031 (Úvod do strojového učenia)

Vypracované na základe slajdov z prednášok, vlastných poznámok, wikipédie, a materiálov zo stránok <https://www.fi.muni.cz/~popel/>

Obsah

Supervised Learning - Učenie s učiteľom

- Consistency

- Bias

- Conjunctive Hypothesis

- Generalisation/Specialization

- Hypothesis Space

 - Veľkosť priestoru hypotéz (všeobecne) - Complexity

- Version Spaces

 - Version Space Learning

Decision Trees - Rozhodovacie stromy

- Complexity:

- Entropy (disorder, impurity)

- IG

- Dealing with Missing Values

- ID3

 - Postup

 - Complexity

- C45 (C4.5)

 - Complexity

- Overfitting - Preučenie

- Overfitting Prevention - Pruning - Prevencia preučenia

 - Metódy určovania podstromov pre pruning

- Regression Trees

Vyhodnocovanie výsledkov

- Baseline - Ground Truth

- Confusion Matrix

- Evaluation Measures

 - Accuracy

 - Error Rate (misclassification rate/error)

 - Precision

 - Recall (sensitivity/true positive rate)

 - Specificity (true negative rate)

 - F1-measure (F1-miera)

- Vyhodnocovanie výsledkov regresných stromov (úloh)

Instance-Based Learning - Lenivé učenie

- Complexity

- Similarity/distance metrics

- Spojité dáta

- Diskrétné dáta

- Iné metódy - Edit Distance

- kNN - k nearest neighbours - k najbližších susedov

- Variácia kNN - kNN pre regresiu

- Metódy efektívneho indexovania

- kd-tree

- Váhy atribútov (Feature Relevance and Weighing)

Unsupervised Learning - Učenie bez učiteľa

- Zhlukovanie

- Hierarchické zhlukovanie

- Agglomerative clustering vs. Divisive clustering

- Direct Clustering Method

- HAC - Hierarchical Agglomerative Clustering

- Complexity

- Podobnosť zhukov - Cluster Similarity

- Nehierarchické zhlukovanie

- k-Means

- Complexity

- Buckshot

- Complexity

- Soft-Clustering

- EM - Expectation Maximization

- Complexity

Supervised Learning - Učenie s učiteľom

Consistency

Mám dvojicu $\langle x, c(x) \rangle$ takú, že x je inštancia, a $c(x)$ je jej správna kategória (resp. Je správne fungujúca neznáma kategorizačná funkcia), a tréningové príklady D .

Hypotetická kategorizačná funkcia $h(x)$ taká, že pre ňu platí, že

$$\forall \langle x, c(x) \rangle \in D : h(x) = (\text{najbližšie k } c(x))$$

Čiže $h(x)$ správne (čo najlepšie) kategorizovala inštanciu x , a teda je KONZISTENTNÁ s dátami, ktoré máme k dispozícii.

Bias

- Je každá podmienka, ktorá nám nejakým spôsobom ovplyvňuje proces učenia. Ak by sme sa pýtali, aký bias je použitý pri rozhodovacích stromoch, je to výber jednej zo všetkých možných ciest - do koreňa vyberieme jeden z atribútov, čím nám vzniknú 2 (alebo viac) možných uzlov pod koreňom, a toľko isto nových ciest. V ďalšom poschodí znova vyberám atribút, podľa ktorého sa môžem pohnúť ďalej. Sú v podstate tri typy:
 - prehľadávací (search)
 - s obmedzením na jazyk (že ma z 3 atribútov zaujímajú len 2, alebo zaujíma ma len cesta o dĺžke max 2)
 - alebo termination bias - kedy mám skončiť
- U **kNN** napr. ignorujem všetky iné prípady, než k najbližších susedov.
- Na bias vyberám ten z atribútov, ktorý má pre nás najvyšší **Information Gain**.

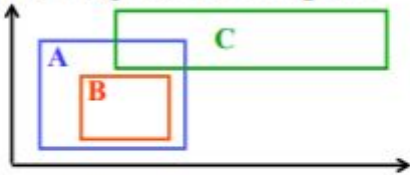
Conjunctive Hypothesis

- Konjunktívna hypotéza je taká, ktorá popisuje všetky spoločné vlastnosti všetkých pozitívnych príkladov.
- Napr. "red & circle" je konjunktívna hypotéza, "(red & circle) OR (blue & something)" je disjunktívna hypotéza
- Strom je sám o sebe DNF -> každá jeho cesta je konjunkcia

Generalisation/Specialization

- Ak máme dve hypotézy, h_1 and h_2 , h_1 je **všeobecnejšia alebo rovnako všeobecná** ako h_2 ($h_1 \geq h_2$) vtedy a práve vtedy, keď každá inštancia, ktorá uspokojí h_2 uspokojí aj h_1 .
- Ak máme dve hypotézy, h_1 and h_2 , h_1 je **striktne všeobecnejšia alebo rovnako všeobecná** ako h_2 ($h_1 > h_2$) vtedy a práve vtedy, keď platí, že $h_1 \geq h_2$, a zároveň neplatí, že $h_2 \geq h_1$.

Examples of Generality

- **Conjunctive feature vectors**
 - $\langle ?, \text{red}, ? \rangle$ is more general than $\langle ?, \text{red}, \text{circle} \rangle$
 - Neither of $\langle ?, \text{red}, ? \rangle$ and $\langle ?, ?, \text{circle} \rangle$ is more general than the other.
- **Axis-parallel rectangles in 2-d space**
 - A is more general than B
 - Neither of A and C are more general than the other.

- Hypotézy sa **musia** zovšeobecniť aby mohli správne klasifikovať inštancie neprítomné v tréningových dátach (len naučenie sa tréningových príkladov “naspamäť” je síce konzistentná hypotéza, ale negeneralizuje)
- Jednoduchá hypotéza pomáha zabezpečiť generalizáciu (Occam's razor)

Treba vedieť:

- Určiť, či je nejaká z dvoch hypotéz všeobecnejšia ako druhá, alebo či medzi nimi neplatí relácia generalizácie (“všeobecnejšia než”) neplatí (napr. “objekt je červený” a “objekt je trojuholník” nie sú hypotézy, medzi ktorými platí relácia generalizácie, ale medzi hypotézami “objekt je červený” a “objekt je červený trojuholník” platí relácia generalizácie)

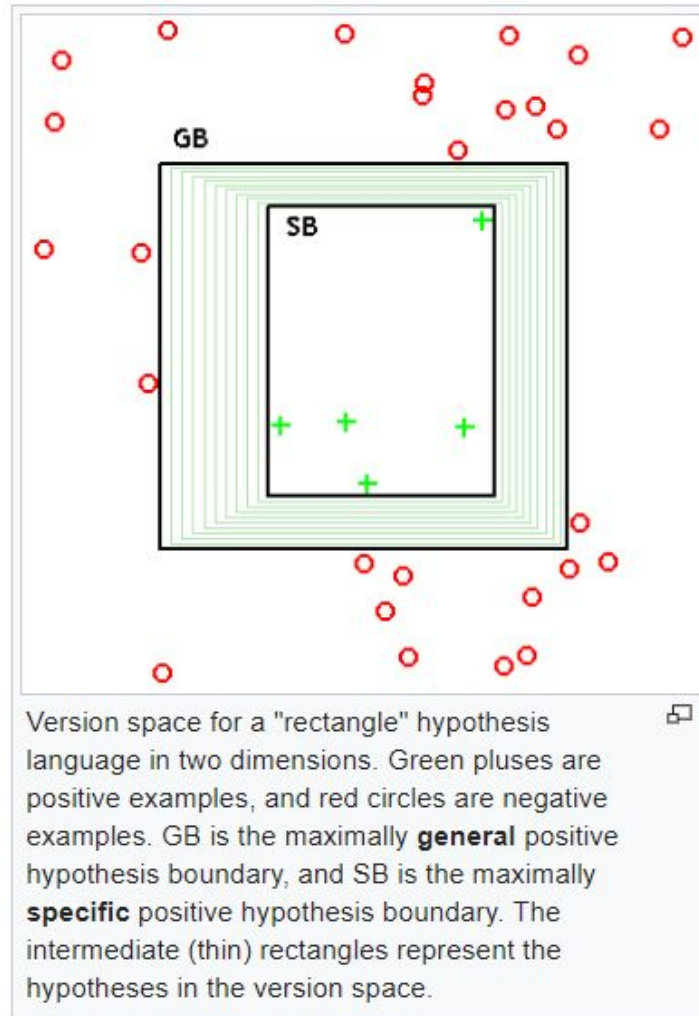
Hypothesis Space

- Vopred obmedzuje naučené funkcie na množinu priestoru hypotéz H , ktorá obsahuje všetky funkcie $h(x)$, ktoré môže byť uvažované ako definície $c(x)$
 - For learning concepts on instances described by n discrete-valued features, consider the space of conjunctive hypotheses represented by a vector of n constraints $\langle c_1, c_2, \dots, c_n \rangle$ where each c_i is either:
 - X , a variable indicating no constraint on the i th feature
 - A specific value from the domain of the i th feature
 - \emptyset indicating no value is acceptable
 - Sample conjunctive hypotheses are
 - $\langle \text{big, red, Z} \rangle$
 - $\langle X, Y, Z \rangle$ (most general hypothesis)
 - $\langle \emptyset, \emptyset, \emptyset \rangle$ (most specific hypothesis)

Veľkosť priestoru hypotéz (všeobecne) - Complexity

- Počet (sémanticky rôznych) hypotéz s binárnymi atribútmi (viď. Sample Generalization Lattice) je $3^n + 1$, kde n je počet bitov na vstupe (počet atribútov)
- Chcená binárna kategorizačná funkcia by mohla byť hocijaká z možných 2^{2^n} možných funkcií pre n vstupných bitov (atribútov)

Version Spaces



Version Space Learning

- Je logický prístup k ML, konkrétne k binárnej klasifikácii
- Takéto algoritmy prehľadávajú vopred určený priestor hypotéz, ktorý vnímajú ako množinu logických viet
- Formálne je priestor hypotéz disjunkcia
- Takýto algoritmus získa príklady (pozitívne a negatívne), podľa ktorých postupne obmedzuje svoj priestor hypotéz: pre každý príklad x odstráni z priestoru hypotéz tie hypotézy, ktoré sú nekonzistentné s x - takýto iteratívny postup sa volá "candidate elimination algorithm", a priestor hypotéz, ktorý si v sebe udržiava, sa volá "version space"

Decision Trees - Rozhodovacie stromy

- Poznáme **Klasifikačné stromy** (v uzloch sa rozhoduje podľa diskretných hodnôt) a **Regresné stromy** (v uzloch sa rozhoduje podľa toho, či je spojitá hodnota atribútu vyššia alebo nižšia než tá v uzle)
- V koreni je feature (atribút), a z neho vychádzajú hrany, ktoré majú návěští odpovedajúce tomu atribútu (ak je diskretný; ak je spojitý, tak ten interval toho atribútu zobrazuje tak, že máme hrany nazvané " $\leq h_i$ " alebo " $> h_i$ ")
- Výhoda: vidíme príklady, na základe ktorých sa algoritmus učil (na rozdiel od napr. Neurónovej siete)
- Nevýhoda: výpočetne náročné; nepovoľuje nám vrátiť sa späť (pevne určená heuristika)

Complexity:

- pesimistický predpoklad, že na každej z ciest sa vyskytne každý príklad. V praxi je však naučený strom málokedy kompletný - väčšinou je lineárny v počte atribútov aj príkladov
 - At each level, i , in the tree, must examine the remaining $m-i$ features for each instance at the level to calculate info gains.

$$\sum_{i=1}^m i \cdot n = O(nm^2)$$

Entropy (disorder, impurity)

- Entropia je spôsob merania náhodnosti binárnej klasifikácie spracovávaných dát - čím vyššia je entropia, tým je ťažšie získať z dát akýkoľvek záver
- Funkcia $Entropy(S)$ = očakávaný počet bitov potrebný na zakódovanie kategórie (+) alebo (-) (pozitívnych alebo negatívnych inštancií) náhodne vybraných členov S (s použitím dátovej kompresie, ako je napr. Huffmanovo kódovanie)
- Počíta sa ako:

$$Entropy(S) \equiv -p_{+} \log_2 p_{+} - p_{-} \log_2 p_{-}$$

Kde S je množina príkladov, p_{+} je pomer pozitívnych príkladov v S a p_{-} je pomer negatívnych príkladov v S .

- Ak sú všetky príklady v jednej kategórii, entropia je 0 (minimum)
- Ak sú príklady v kategóriách 50:50, entropia je 1 (maximum)

IG

- Je vlastnosť featury F (atribútu), ktorá popisuje redukciu v entropii, ktorú očakávame pri použití tejto featury v uzle stromu na rozdelenie inštancií

$$Gain(S, F) = Entropy(S) - \sum_{v \in \text{values}(F)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where S_v is the subset of S having value v for feature F .

Dealing with Missing Values

Napr. môžeme:

- doplniť chýbajúcu hodnotu priemernou hodnotou z dát (median impute). Ak mám napr. 2 triedy, kde má jedna trieda priemernú hodnotu 1 a druhá 1000, je lepšie dosadiť priemernú hodnotu tej triedy, do ktorej je daná inštancia klasifikovaná
- Odstrániť túto inštanciu
- iné

ID3

- Ross Quinlan
- Algoritmus, ktorý rozdeľuje priestor v decision tree tak, aby sme mohli izolovať negatívne hodnoty atribútov od pozitívnych
- Preferuje kratšie stromy s atribútmi s vysokým IG blízko koreňa
- Bias je preferencia pre niektoré hypotézy, nie reštrikcia na priestor hypotéz
- Occam's razor - najkratšia (najjednoduchšia) hypotéza konzistentná s dátami

Postup

- Začína s pôvodnou množinou S v koreni
- V každej iterácii iteruje cez každý atribút množiny S a počíta entropiu alebo IG daného atribútu
- Vyberie atribút s najnižšou entropiou (najvyšším IG) a podľa neho rozdelí S na podmnožiny
- Algoritmus rekurzívne pokračuje na každej podmnožine (každým uzle), ale uvažuje len atribúty, ktoré ešte predtým nevybral
- Rekúzia na podmnožine sa zastaví ak:
 - Všetky inštancie v danej podmnožine patria do tej istej triedy - uzol je listom a je nazvaný podľa danej triedy
 - Už nemá žiadne atribúty, z ktorých by vyberal, avšak všetky inštancie podmnožiny ešte nepatria do jednej triedy. Vtedy je vytvorený list podľa najčastejšie vyskytujúcej sa triedy inštancií

- V podmnožine už neostali žiadne inštalácie (napr. Absencia osoby v populácii vo veku nad 100 rokov). Vtedy je vytvorený list podľa najčastejšie vyskytujúcej sa triedy inštalácií jeho rodiča

Complexity

- Nech n je počet atribútov, a m počet inštalácií, potom je časová náročnosť stromovej indukcie $O(n*m*\log(m))$ (platí aj pre ID3)

C45 (C4.5)

- Ross Quinlan
- Vylepšená verzia ID3
- Využíva ho väčšina komerčných aj nekomerčných data mining nástrojov
- Dá sa nastaviť na binárne stromy

Scheme of C4.5 algorithm:

Run several time and choose the best tree

Inner: Take $L\%$ of learning data randomly

Call ID3 (pre-pruning, see $-m$ parameter)

Prune the tree (post-pruning, $-cf$)

Take $T\%$ of unseen learning data for validation

If validation criterion holds, exit

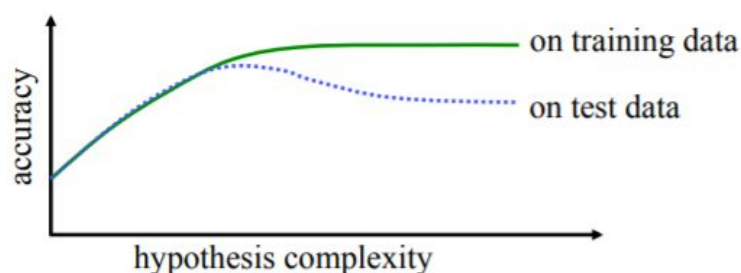
Otherwise add $L_{crement}$ to L and go to Inner

Complexity

- Nech m je počet inštalácií a n počet atribútov, potom je zložitosť C4.5 $O(m * n^2)$

Overfitting - Preučenie

- Naučenie sa stromu, ktorý dokonale klasifikuje tréningové dáta, nemusí viesť k najlepšej generalizácii pre nepoznané dáta
 - V tréningových dátach môže byť ruch, ktorý strom nesprávne splňa
 - Algoritmus môže blízko pri listoch robiť nesprávne rozhodnutia na základe príliš málo dát, čo nemusí zodpovedať skutočným spoľahlivým trendom v strome
- Hypotéza h spravila "overfit" (prečila sa) na tréningových dátach, ak existuje nejaké iná hypotéza h' taká, že h má menšiu chybu na tréningových dátach, než h' , ale vyššiu chybu na nezávislých testovacích dátach



Overfitting Prevention - Pruning - Prevencia preučenia

Dva základné prístupy:

1. **Prepruning** - prestať narastanie stromu počas top-down konštrukcie vtedy, keď nemáme dostatočné dáta na robenie spoľahlivých rozhodnutí
2. **Postpruning** - nechať narásť celý strom, a dodatočne odstrániť také podstromy, ktoré majú nedostatok dôkazov (nedostatok spoľahlivých dát)

Metódy určovania podstromov pre pruning

1. **Cross-validation** - ponechaj si trochu tréningových dát ako "hold-out set" (validačná množina, ladiaca množina) na evaluáciu užitočnosti podstromu
2. **Štatistický test** - použi štatistický test na tréningových dátach na určenie, či môže byť nejaká z pozorovaných pravidielností môže byť zanedbaná ako pravdepodobná náhoda
3. **Minimum description length (MDL)** - urči, či je dodatočná zložitosť hypotézy nižšia, než explicitné zapamätanie si hocijakých výnimiek vytvorených pruningom

Regression Trees

- Povoľujú real-valued atribúty - spojité hodnoty
- V uzloch sa ďalej rozdeľujú podľa toho či je hodnota menšia alebo väčšia než tá v uzle

Vyhodnocovanie výsledkov

Baseline - Ground Truth

- Nejaký štandard, podľa ktorého môžem porovnávať iné príklady
- Vo väčšine úloh nemám

Confusion Matrix

- Špeciálna tabuľka (matica), ktorá umožňuje vizualizáciu údajov a výkonnosti algoritmu, typicky Supervised Learning algoritmu
- Každý riadok reprezentuje inštancie predikovanej triedy, zatiaľ čo každý stĺpec reprezentuje inštancie skutočnej triedy
- Meno podľa toho, že vďaka tejto vizualizácii je ľahké vidieť, či si algoritmus zamieňa (confuses) tieto triedy
- Využíva učenie s cenami
- True Positive, True Negative, False Positive, False Negative

Evaluation Measures

Accuracy

- Celková správnosť - ako často sa algoritmus trafil do správnej odpovede

$$Acc = \frac{TP+TN}{TP+TN+FP+FN}$$

Error Rate (misclassification rate/error)

- Chyba

$$Err = 1 - Acc = \frac{w_{FP}*FP + w_{FN}*FN}{TP+TN+FP+FN}$$

- Weighted error

w_{FP}, w_{FN} ... weight of FP and FN errors
default $w_{FP}, w_{FN} = 1$

Precision

- Percento výsledkov, ktoré sú relevantné

$$\frac{TP}{TP+FP}$$

Recall (sensitivity/true positive rate)

- Percento všetkých výsledkov, ktoré algoritmus správne klasifikoval ako pozitívne

$$\frac{TP}{TP+FN}$$

Specificity (true negative rate)

- Percento všetkých výsledkov, ktoré algoritmus správne klasifikoval ako negatívne

$$\frac{TN}{TN+FP}$$

F1-measure (F1-miera)

- Miera celkovej správnosti testu
- Počíta s precision a recall

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

$$F_\beta = \frac{(1 + \beta^2) precision * recall}{\beta^2 * precision + recall}$$

β ... a non-negative real number

Vyhodnocovanie výsledkov regresných stromov (úloh)

MSE - Mean-squared error

- == mean((reality - prediction)^2) (eg. mean((CO2\$uptake - prediction)^2))

$$\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}$$

RMSE - Root mean-squared error

- == sqrt(mean((reality - prediction)^2))

$$\sqrt{\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}}$$

MAE - Mean-absolute error

- == mean(abs(reality - prediction))

$$\frac{|p_1 - a_1| + \dots + |p_n - a_n|}{n}$$

Viac vid'. [Evaluation Measures](#)

Instance-Based Learning - Lenivé učenie

- Aj hypotézy typu “pozitívne inštancie sú všetky inštancie, ktoré sú pozitívne” majú zmysel -> key idea: “ulož si všetky tréningové príklady také, že $\langle x, c(x) \rangle$ ”
- Takýto algoritmus namiesto robenia explicitných generalizácií porovnáva nové inštancie s tými, ktoré už videl v tréningových dátach (ktoré si ukladá)
- Schopnosť prispôbiť svoj model predtým nepoznaným dátam

Complexity

- Takýto algoritmus konštruuje hypotézy priamo z tréningových dát - preto môže jeho zložitosť narastať úmerne s dátami
 - V najhoršom prípade je hypotéza zoznam n tréningových inštancií, a teda je aj zložitosť klasifikovania novej inštancie $O(n)$.

Similarity/distance metrics

Instance-based metódy využívajú nejakú funkciu na určenie vzdialenosti/rovnakosti medzi dvoma inštanciami.

Spojité dáta

- Počítame pomocou Euklidovskej vzdialenosti

$$d(x_i, x_j) = \sqrt{\sum_{p=1}^n (a_p(x_i) - a_p(x_j))^2}$$

Where $a_p(x)$ is the value of the p th feature of instance x .

Diskrétné dáta

- Využijeme princíp Hammingovej vzdialenosti - vzdialenosť je 0, ak je to tá istá inštancia, 1 ak sú rôzne
- Aby sme vykompenzovali rozdiely v jednotkách medzi features, všetky nascaleujeme do intervalu $[0, 1]$

Iné metódy - Edit Distance

- Napr. medzi stringami “abcd” a “bacd” je ich edit distance počet operácií potrebných na úpravu jedného na druhý

kNN - k nearest neighbours - k najbližších susedov

- Vypočítaj vzdialenosť medzi testovanou inštanciou a všetkými tréningovými inštanciami
- Vyber k najbližších inštancií a testovanej inštancii prirad' tú kategóriu, ktorá je medzi nimi najčastejšia
- Použi radšej nepárne k aby nedošlo k

Variácia kNN - kNN pre regresiu

- Môže byť použitá na odhadnutie hodnoty spojitej funkcie (funkcia s reálnymi číslami)
 - regresia - použitím priemernej hodnoty funkcie v k najbližších bodoch
- Všetky tréningové inštancie môžu byť použité na zlepšenie klasifikácie testovanej inštancie tým, že každej tréningovej inštancii bude priradená váha (weighted vote) podľa inverzie druhej mocniny jej vzdialenosti od testovanej inštancie ("All training examples can be used to help classify a test instance by giving every training example a vote that is weighted by the inverse square of its distance from the test instance.")

Metódy efektívneho indexovania

- Lineárne prehľadávanie na nájdenie najbližšieho suseda nie je efektívne na veľkých dátových setoch, preto môžeme pre zrýchlenie testovania vytvoriť indexované štruktúry
- Pre euklidovskú vzdialenosť môžeme vytvoriť **kd-tree**, ktorý redukuje očakávaný čas potrebný na nájdenie najbližšieho suseda na $O(\log(n))$

kd-tree

- Binárny strom, v ktorom je každý uzol k -dimenzionálny bod
- Každý nelistový uzol generuje rozdeľovaciu rovinu, ktorá rozdelí priestor do dvoch podpriestorov
- Body vľavo od tejto roviny reprezentujú ľavý podstrom tohto uzla, body vpravo pravý podstrom
- Smerovanie roviny je vybrané nasledovne: každý uzol rozdelený na podstromy je asociovaný s jednou z k -dimenzií tak, že rovina je kolmá na k -tý vektor

Váhy atribútov (Feature Relevance and Weighing)

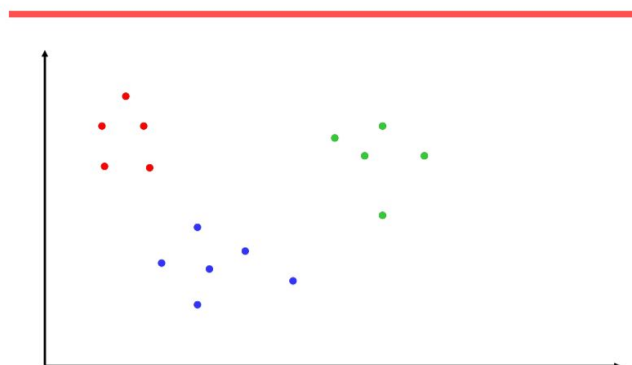
- Štandardne má každá feature rovnakú váhu - čo je ale problematické ak je mnoho z features irelevantných, čo by spôsobilo nesprávnu klasifikáciu
- Na ohodnotenie (nastavenie váhy) featurey môže byť použitý napr. IG danej featurey

Unsupervised Learning - Učenie bez učiteľa

Zhlukovanie

- Hľadám zobernení popis skupiny tak, aby inštancie v clustery boli veľmi podobné
- Hľadám také rozdelenie dát, aby každá inštancia bola v nejakom clustery, a tie clustery boli medzi sebou ostro disjunktné
- Objavujeme nové kategórie bez supervízie (no sample category labels)

Clustering Example



Hierarchické zhlukovanie

- Ako rozdelenie zvieracej ríše - tree-based hierarchická taxonómia (dendrogram) z množiny neoznačených príkladov
- Rekurzívna aplikácia štandardného clustering algoritmu môže vytvoriť hierarchický clustering

Agglomerative clustering vs. Divisive clustering

Agglomerative - bottom-up - začnú s každou inštanciou vo svojom vlastnom zhluke, a iteratívne ich kombinujú aby vytvorili väčšie a väčšie zhluky

Divisive - partitional, top-down - ihneď rozdelí všetky inštancie do zhlukov

Direct Clustering Method

- Požadujú presne určený počet zhlukov (k)
- **Clustering evaluation function** je funkcia, ktorá priradí zhluke kvalitatívnu mieru v podobe reálneho čísla
- (Ideálny) počet zhlukov môže byť automaticky určený explicitným generovaním viacerých zhľukovaní s rôznymi hodnotami k , aplikovaním clustering evaluation function na každý z rôznych zhľukovaní, a nájdením najlepšieho výsledku

HAC - Hierarchical Agglomerative Clustering

- Využíva funkciu podobnosti na určenie podobnosti dvoch inštancií
- Začne so všetkými inštanciami v separátnych zhluchoch a potom opakovane spája dva zhluhy, ktoré si sú najviac podobné, až pokým nezostane len jeden zhluh
- História spájania tvorí binárny strom alebo hierarchiu

HAC Algorithm

Start with all instances in their own cluster.

Until there is only one cluster:

Among the current clusters, determine the two clusters, c_i and c_j , that are most similar.

Replace c_i and c_j with a single cluster $c_i \cup c_j$

Complexity

$O(n^2)$

Podobnosť zhluhov - Cluster Similarity

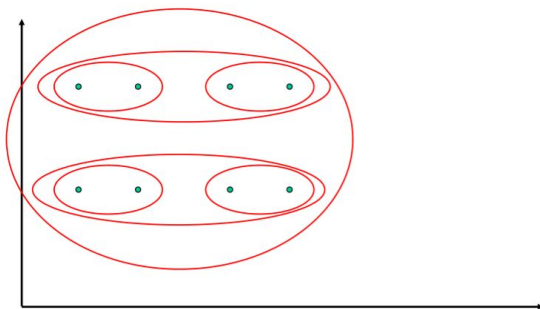
- Predpokladajme funkciu priradzujúcu podobnosť dvom inštanciam (Euclidean/Mahalanobis, Hamming, Cosine similarity, Pearson r, ...)

Simple Link - Podobnosť je určená dvojicou najpodobnejších inštancií, takých, že jedna je z jedného zhluhu, druhá z druhého.

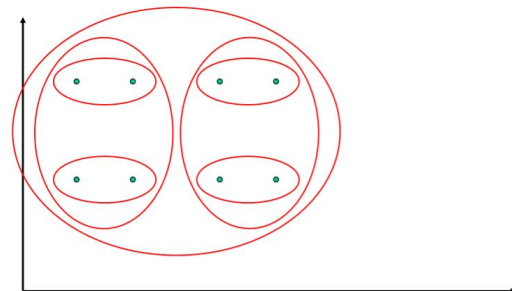
Complete Link - Podobnosť je určená dvojicou najmenej podobných inštancií, takých, že jedna je z jedného zhluhu, druhá z druhého.

Group Average - Podobnosť je určená priemernou podobnosťou medzi členmi zhluhov.

Single Link Example



Complete Link Example



Nehierarchické zhlukovanie

- Náhodne vyberiem k bodov (seeds), o ktorých predpokladám, že sú reprezentujú zhluky
- Podľa nich vytvorím zhluky - to ale môže dopadnúť zle (ak boli zle vybrané seeds)
- Tak prepočítam seeds a premiestnim body do nových zhlukov
- Ak toto zopakujem a k žiadnej zmene nedôjde, našli sme optimálne riešenie, inak opakujeme znova

k-Means

- Predpokladá, že inštancie sú real-valued vektory
- Zhluky sú založené na centroidoch (body v priestore reprezentujúce ťažiská) alebo priemere bodov v zhluku (podľa týchto vypočítaných bodov vyberiem seeds)
- Premiestnenie inštancie do iného zhluku je založené na vzdialenosti k ťažisku momentálneho zhluku
- Cieľom je zmenšiť totálny súčet umocnených vzdialeností každého bodu (inštancie) od ťažiska jeho zhluku
- NP

K-Means Algorithm

Let d be the distance measure between instances.

Select k random instances $\{s_1, s_2, \dots, s_k\}$ as seeds.

Until clustering converges or other stopping criterion:

For each instance x_i :

Assign x_i to the cluster c_j such that $d(x_i, s_j)$ is minimal.

(Update the seeds to the centroid of each cluster)

For each cluster c_j

$$s_j = \mu(c_j)$$

Complexity

$O(i*k*n*m)$, kde i je počet iterácií, m je rozmer vektorov, a $k*n$ je počítanie vzdialenosti

Buckshot

- Kombinuje HAC a K-Means clustering
- Najprv náhodne vyberieme z inštancií o veľkosti \sqrt{n}
- Spustíme group-average HAC na tejto vzorke, čo zaberie $O(n)$
- Použij výsledky HAC ako initial seeds pre K-Means
- Vyhyba sa problému bad seed selection
- Nezaručuje že si vytvoríme optimálne cluster
- Ale je to výhodné

Complexity

Celkový algoritmus má zložitosť $O(n)$.

Soft-Clustering

- Flexibilnejšie zhľukovanie
- Typický clustering nedovolí neistotu v tom, do akého zhľuku inštancia patrí, či to, aby inštancia patrila do viac než jedného zhľuku
- V soft-clusteringu má každá inštancia priradenú pravdepodobnosť, že patrí do množiny zhľukov

EM - Expectation Maximization

- Pravdepodobnostná metóda pre soft-clustering
- Priama metóda ktorá predpokladá k zhľukov
- Soft-verzia k-means
- Iteratívna metóda na učenie pravdepodobnostného modelu z nesledovaných dát
 - Initially assume random assignment of examples to categories.
 - Learn an initial probabilistic model by estimating model parameters θ from this randomly labeled data.
 - Iterate following two steps until convergence:
 - **Expectation (E-step):** Compute $P(c_i | E)$ for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates.
 - **Maximization (M-step):** Re-estimate the model parameters, θ , from the probabilistically re-labeled data.

Complexity

$O(n*k*i)$, kde n je počet inštancií, k je počet zhľukov, a i je počet iterácií