Capitolul 9

Data mining – clustering

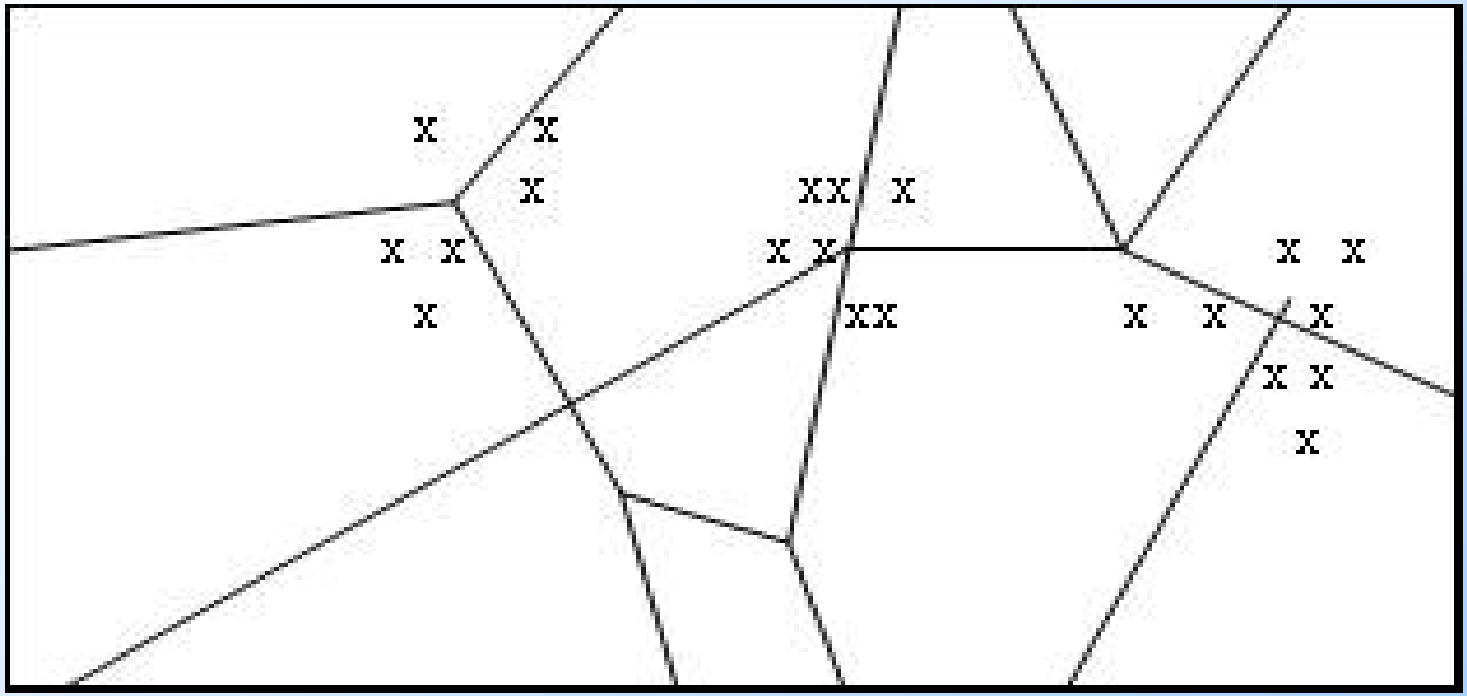
Problema

- Dându-se puncte într-un spaţiu oarecare – deseori un spaţiu cu foarte multe dimensiuni – grupează punctele într-un numar mic de *clustere*, fiecare cluster constând din puncte care sunt "apropiate" într-un anume sens.

Exemple de aplicatie

1. Cu mulţi ani în urmă, în timpul unei izbucniri a holerei în Londra, un medic a marcat localizarea cazurilor pe o hartă, obţinând un desen care arăta ca în fifura urmatoare:

Exemple de aplicatie



Exemple de aplicatie

- Vizualizate corespunzător, datele au indicat că apariţiile cazurilor se grupează în jurul unor intersecţii, unde existau puţuri infestate, arătând nu numai cauza holerei ci indicând şi ce e de făcut pentru rezolvarea problemei.
- Din păcate nu toate problemele de data mining sunt atât de simple, deseori deoarece clusterele sunt în atât de multe dimensiuni încât vizualizarea este foarte dificilă.

Exemple de aplicatie

2. *Skycat* a grupat în clustere 2 x 109 obiecte cereşti în stele, galaxii, quasari, etc.

- Fiecare obiect era un punct într-un spaţiu cu 7 dimensiuni, unde fiecare dimensiune reprezenta nivelul radiaţiei într-o bandă a spectrului.
- Proiectul Sloan Sky Survey este o încercare mult mai ambiţioasă de a cataloga şi grupa întregul univers vizibil.

Exemple de aplicatie

3. Documentele pot fi percepute ca puncte într-un spaţiu multi-dimensional în care fiecare dimensiune corespunde unui cuvânt posibil.
4. Poziţia documentului într-o dimensiune este dată de numărul de ori în care cuvântul apare în document (sau doar 1 dacă apare, 0 dacă nu).
5. Clusterele de documente în acest spaţiu corespund deseori cu grupuri de documente din acelaşi domeniu.

Distanta

- Pentru a discuta dacă o mulţime de puncte sunt suficient de apropiate pentru a fi considerate un cluster avem nevoie de o *măsură a distanţei* D(x, y) care spune cât de depărtate sunt punctele x şi y.
- Nu orice functie poate fi utilizata ca functie de masurarea distantei.

Distanta

- Axiomele uzuale pentru o măsură a distanţei D sunt urmatoarele:

1. D(x, y) >= 0
2. D(*x*, *x*) = 0. Un punct este la distanţă 0 de el însuşi.
3. D(*x*, *y*) = D(*y*, *x*). Distanţa e simetrică.
4. D(*x*, *y*) ≤ D(*x*, *z*) + D(*z*, *y*). *Inegalitatea triunghiului*.

Distanta

- Deseori punctele pot fi percepute ca existând într-un spaţiu euclidian k-dimensional şi distanţa între orice două puncte:
- x = [x1, x2, ..., xk] şi
- y = [y1, y2, ..., yk]

este dată într-una din manierele uzuale:

- Distanţa comună ("norma L2")
- Distanţa *Manhattan* ("norma L1")
- Maximul pe o dimensiune ("norma L∞")

Distanta comuna

- Distanta comuna (sau norma L2) este data de formula cunoscuta:

$$\sqrt{\Sigma_{i=1}^{k}(x_i - y_i)^2}$$

Distanta Manhattan

- Distanta Manhattan (sau norma L1) este data de formula urmatoare:

$$\sum_{i=1}^{k} |x_i - y_i|$$

- Ea poate fi folosita de exemplu si pentru calculul distantei intre doua puncte pe o placheta cu circuite imprimate multistrat.

Maximul pe o dimensiune

- Este data de formula:

$$\max_{i=1}^{k} |x_i - y_i|$$

- Aceasta functie verifica toate cele 4 conditii pentru a fi functie de distanta.
- Poate fi folosita de exemplu pentru spatii euclidiene hiperdimensionale (numar de dimensiuni foarte mare)

Alte distante

- Unde nu există un spaţiu euclidian în care să plasăm punctele gruparea devine mult mai dificilă.
- Iată un exemplu în care are sens: o măsură a distanţei în lipsa unui spaţiu euclidian

Alte distante

- Şirurile de caractere, cum sunt secvenţele ADN, pot fi similare chiar şi dacă există unele inserări şi ştergeri precum şi modificări ale unor caractere.
- De exemplu, *abcde* şi *bcdxye* sunt destul de similare chiar dacă nu au nici o poziţie comună şi nu au nici chiar aceeaşi lungime.

Alte distante

- Astfel, în loc să construim un spaţiu euclidian cu câte o dimensiune pentru fiecare poziţie, putem defini funcţia distanţă:

$$D(x, y) = |x| + |y| - 2|LCS(x, y)|$$

unde LCS este cea mai lunga subsecvenţă comună lui *x* şi *y*.

- În exemplul nostru LCS(*abcde*, *bcdxye*) este *bcde* de lungime 4, deci D(*abcde*, *bcdxye*) = $5 + 6 - 2 \times 4 = 3$; i.e. şirurile sunt destul de apropiate.

Alte distante

- Aceasta functie de distanta arata cate caractere trebuie sterse sau adaugate unuia dintre siruri pentru a obtine celalalt sir.
- Intr-adevar, pentru a obtine de exemplu pe *abcde* din *bcdxye* trebuie sa:
  1. Adaugam un a in fata
  2. Stergem pe x
  3. Stergem pe y

Hiperdimensionalitatea

- O consecinţă mai puţin intuitivă a lucrului în spaţii hiperdimensionale este că aproape toate perechile de puncte sunt la o depărtare aproape egală cu media distanţelor între puncte.

Exemplu:

- Să presupunem că aruncăm aleator puncte într-un cub *k*-dimensional.
- Pentru *k*=2, ne aşteptăm ca punctele să fie răspândite în plan cu unele foarte apropiate între ele şi alte perechi aproape la distanţa maxim posibila.

Hiperdimensionalitatea

- Cu toate acestea, să presupunem *k* foarte mare, să zicem 100.000. Indiferent de norma folosită, L_2 , L_1 sau L_∞ , ştim că:

$$D(x, y) \geq \max_i |x_i - y_i|$$

pentru $x = [x1, x2, ...]$ şi $y = [y1, y2, ...]$.

- Pentru *k* foarte mare, e foarte posibil să existe o dimensiune *i* astfel încât *x*i şi *y*i sunt diferite aproape de maximul posibil, chiar dacă *x* şi *y* sunt foarte apropiate în alte dimensiuni.
- Astfel $D(x, y)$ va fi foarte apropiată de 1.

Hiperdimensionalitatea

- O alta consecinţă interesantă a hiper-dimensionalităţii este că toţi vectorii,

  $x = [x1, x2, ...]$ şi $y = [y1, y2, ...]$

  sunt aproape ortogonali.
- Motivul este că dacă proiectăm *x* şi *y* pe oricare dintre cele plane formate de două dintre cele *k* axe va exista unul în care proiecţiile vectorilor sunt aproape ortogonale (probabilitatea sa existe creste cu numarul de dimensiuni)

Abordari clustering

- La nivel înalt, putem împărţi algoritmii de grupare în două mari clase:

1. Abordarea tip centroid: ‘ghicim’ centriozii sau punctele centrale pentru fiecare cluster şi asignăm punctele la clusterul având cel mai apropiat centroid.

Abordari clustering

2. Abordarea ierarhică:

- Începem prin a considera că fiecare punct formează un cluster.
- Comasăm repetat clusterele apropiate prin folosirea unei măsuri pentru apropierea a două clustere (e.g. distanţa dintre centroizii lor), sau pentru cât de bun va fi clusterul rezultat (e.g. distanţa medie de la punctele din cluster la noul centroid rezultat).

Algoritmul k-means

- Acest algoritm este un algoritm popular care *ţine datele în memoria centrală*
- Pe acest algoritm care se bazează alti algoritmi de clustering (ex.: BFR)
- *k*-means alege *k* centroizi de cluster şi asignează punctele la acestea alegând centroidul cel mai apropiat de punctul respectiv.
- Pe măsură ce punctele sunt asignate la un cluster, centroidul acestuia poate migra.

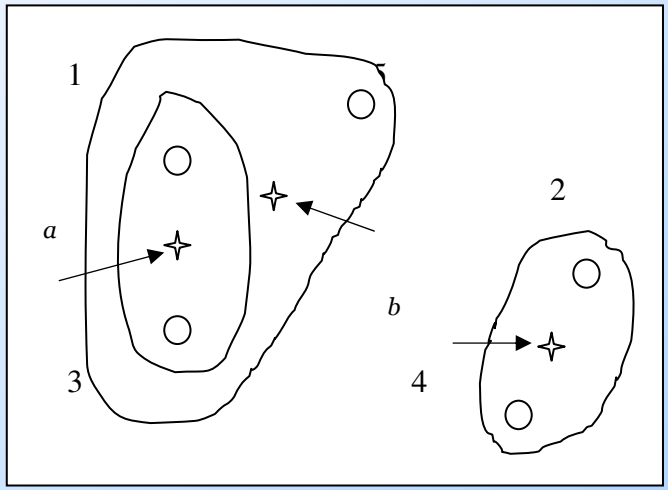
Observatie

- Centroidul unui cluster este 'centrul de greutate' al clusterului.
- El nu este de obicei unul din punctele care formeaza clusterul ci un punct intre ele in spatiul euclidian respectiv.
- Conceptul de centroid nu este valabil decat in cazul clusteringului in spatii euclidiene.
- Pentru cazurile in care nu avem un spatiu euclidian (punctele nu au coordonate) se foloseste alternativ conceptul de clustroid.

Exemplu

- Un exemplu foarte simplu cu cinci puncte în două dimensiuni.
- Presupunem că asignăm punctele 1, 2, 3, 4 şi 5 în această ordine, cu k=2.
- Atunci punctele 1 şi 2 sunt asignate celor două clustere şi devin centroidul lor pentru moment.

Exemplu



Exemplu

- Când considerăm punctul 3, să presupunem că este mai apropiat de 1, deci 3 se adaugă clusterului conţinând 1 iar centroidul acestuia se mută în punctul marcat ca *a*.
- Presupunem că atunci când asignăm 4 găsim că 4 este mai aproape de 2 decât de *a*, deci 4 se alătură lui 2 în clusterul acestuia iar centrul se mută în *b*.
- În final, 5 este mai aproape de *a* decât de *b*, deci el se adaugă la clusterul {1, 3} al cărui centroid se mută în *c*.

Aplicare k-means

- Putem iniţializa cei *k* centroizi alegând puncte suficient de depărtate de orice alt centroid până obţinem *k*.
- Pe măsură ce calculul progresează putem decide să spargem un cluster şi să unim două dintre ele pentru a păstra totalul de *k*. Testul pentru a decide asta poate fi să ne întrebam dacă făcând operaţia respectivă se reduce distanţa medie de la puncte la centroidul lor.

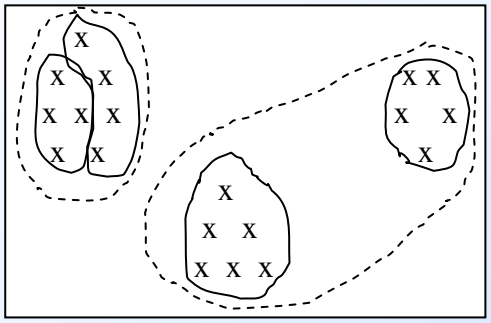
Aplicare k-means

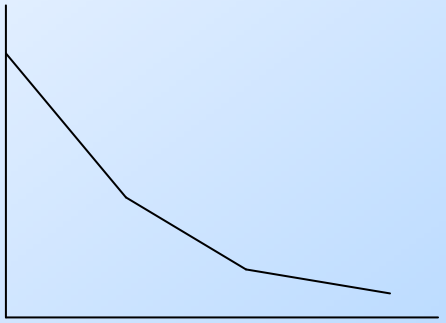
- După localizarea centroizilor celor *k* clustere putem reasigna toate punctele deoarece unele puncte care au fost asignate la început pot acum să fie mai aproape de un alt centroid care s-a mutat.
- Dacă nu suntem siguri de valoarea lui *k* putem încerca valori diferite pentru *k* până când găsim cel mai mic *k* astfel încât mărirea lui *k* nu micşorează prea mult distanţa medie a punctelor faţă de centroidul lor. Exemplul urmator ilustrează acest lucru.

Alt exemplu

- Să considerăm datele din figura urmatoare.
- În mod clar k=3 este numărul corect de clustere dar să presupunem ca întâi încercăm k=1.
- În acest caz toate punctele sunt într-un singur cluster şi distanţa medie la centroid va fi mare.

Alt exemplu - cont





- Presupunem că apoi încercăm *k*=2.
- Unul dintre cele trei clustere va fi un cluster iar celelalte două vor fi forţate să creeze un singur cluster, asa cum arată linia punctată.
- Distanţa medie a punctelor la centroid de va micşora astfel considerabil.

Alt exemplu - cont

- Daca luăm *k*=3 atunci fiecare dintre clusterele vizibile va forma un cluster iar distanţa medie de la puncte la centroizi se va micşora din nou, aşa cum arată graficul din Ffigura.
- Totuşi, dacă mărim *k* la 4 unul dintre adevăratele clustere va fi partiţionat artificial în două clustere apropiate, aşa cum arata liniile continui.

Alt exemplu - cont

- Distanţa medie la centroid va scădea puţin dar nu mult.
- Acest eşec de a merge mai departe ne arată că valoarea k=3 este corectă chiar dacă datele sunt în atât de multe dimensiuni încât nu putem vizualiza clusterele.
- In acest fel aflam valoarea corecta a lui k – numarul de clustere

Algoritmul BFR

- Bazat pe *k*-means, acest algoritm citeşte datele o singură dată în tranşe egale cu memoria centrală disponibilă la fiecare pas.
- Algoritmul lucrează cel mai bine dacă clusterele sunt normal distribuite în jurul unui punct central, eventual cu o deviaţie standard diferită în fiecare dimensiune.

Reprezentare clustere în BFR

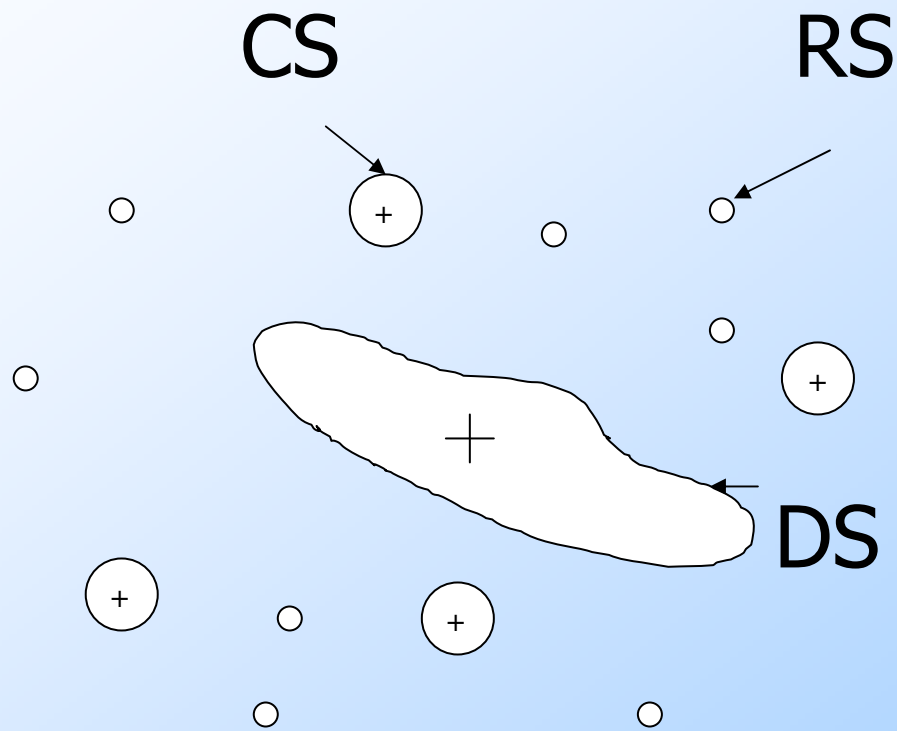
Cei care au creat acest algoritm şi-au reprezentat clusterele ca pe nişte galaxii.

Un cluster constă din:

1. Un nucleu central numit **Discard set - DS**.
2. Mulţimea acestor puncte este considerată ca aparţinând în mod sigur clusterului.

- Toate punctele din această mulţime sunt înlocuite de nişte statistici simple, descrise in continuare.
- Notă: deşi numite puncte « de aruncat » acestea au de fapt un efect semnificativ pe parcursul execuţiei algoritmului de vreme ce determină colectiv unde este centroidul şi care este deviaţia standard a clusterului în fiecare dimensiune.

Reprezentare clustere în BFR



Reprezentare clustere în BFR

2. Galaxii înconjurătoare, numite colectiv **Compression set** – CS (*Mulţimea comprimată*).

- Fiecare subcluster din CS constă într-un grup de puncte care sunt suficient de apropiate unele de altele încât pot fi înlocuite cu statisticile lor, la fel ca şi DS.
- Totuşi, ele sunt suficient de departe de orice centroid de cluster încît nu suntem încă siguri de care cluster aparţin.

Reprezentare clustere în BFR

- Stele individuale care nu sunt parte a nici unei galaxii sau subgalaxii, *Mulţimea reţinută* (**Retained set – RS**).
- Aceste puncte nici nu pot fi asignate vreunui cluster nici grupate în vreun subcluster al CS.
- Ele sunt stocate în memoria centrală ca puncte individuale împreună cu statisticile pentru DS şi CS.

Reprezentare comprimata

- Statisticile utilizate pentru a reprezenta fiecare cluster din DS şi fiecare subcluster din CS sunt:

1. Contorul numărului de puncte *N*.
2. Vectorul sumelor coordonatelor punctelor în fiecare dimensiune. Vectorul este notat cu *SUM* iar componenta pentru dimensiunea *i* cu SUM_i
3. Vectorul sumelor pătratelor coordonatelor punctelor în fiecare dimensiune notat cu *SUMSQ.* Componenta pentru dimensiunea *i* cu $SUMSQ_i$

Reprezentare - cont

- De notat că aceste trei tipuri de informaţii, totalizând în cazul în care avem k dimensiuni $2k+1$ numere sunt suficiente pentru a calcula statistici importante pentru un cluster sau subcluster.
- Este mai convenabil de menţinut pe măsură ce punctele sunt adăugate la cluster decât, să spunem, media şi varianţa în fiecare dimensiune.

Reprezentare - cont

- Cordonata μ_i a centroidului clusterului în dimensiunea i este SUM_i /N
- Varianţa în dimensiunea *i* este:

$$\frac{SUMSQ_i}{N} - \left(\frac{SUM_i}{N}\right)^2$$

- iar deviaţia standard σ este rădăcina pătrată a acesteia.

Procesare

- La prima încărcare cu date a memoriei centrale, BFR selectează *k* centroizi de clustere utilizând un algoritm oarecare lucrând în memoria centrală, e.g. se ia un eşantion al datelor, se optimizează exact clusterele şi se aleg centroizii lor ca centroizi iniţiali.
- O memorie centrală de puncte este procesată la fel în toate încărcările cu date urmatoare după cum urmează:

1. Se determină care puncte sunt suficient de apropiate de un centroid curent astfel încât pot fi luate în DS iar statisticile lor (*N, SUM, SUMSQ*) combinate cu statisticile anterioare ale clusterului.

Procesare

2. În memoria centrală se încearcă gruparea punctelor care nu au fost încă plasate în DS, inclusiv puncte ale RS din paşii precedenţi.

- Dacă găsim un cluster de puncte a căror varianţă este sub un prag ales, atunci vom privi aceste puncte ca un subcluster, le înlocuim cu statisticile lor şi le considerăm parte a CS.
- Toate celelalte puncte vor fi plasate în RS.

Procesare

- Luăm în considerare unirea unui subcluster nou apărut cu un subcluster anterior din CS. Testul pentru a vedea dacă este de dorit să facem asta este ca mulţimea combinată de puncte să aibă o varianţă sub un anumit prag. De notat că statisticile ţinute pentru subclusterele din CS sunt suficiente pentru a calcula varianţa mulţimii combinate.

Procesare

- Dacă este ultimul pas, i.e. nu mai sunt date, atunci putem asigna subclusterele din CS şi punctele din RS la cel mai apropiat cluster de ele chiar dacă ele vor fi destul de departe de orice centroid de cluster.
- In felul acesta obtinem clusterele finale produse de algoritm

Scalarea multidimensionala

- In multe cazuri nu avem un spatiu euclidian ci doar o multime de puncte si distanta intre oricare doua dintre acestea.
- Exemplu: un graf in care cunoastem dimensiunea fiecarui arc. Din acestea putem afla distanta intre oricare doua noduri ca fiind lungimea drumului minim intre ele

Scalarea multidimensionala

- Se poate demonstra ca avand N puncte si distantele intre oricare 2 dintre ele putem crea un spatiu cu N-1 dimensiuni
- In acest spatiu punctele sunt plasate plasate exact (distanta calculata din coordonate este aceeasi cu distanta de la care s-a plecat pentru orice pereche de puncte)

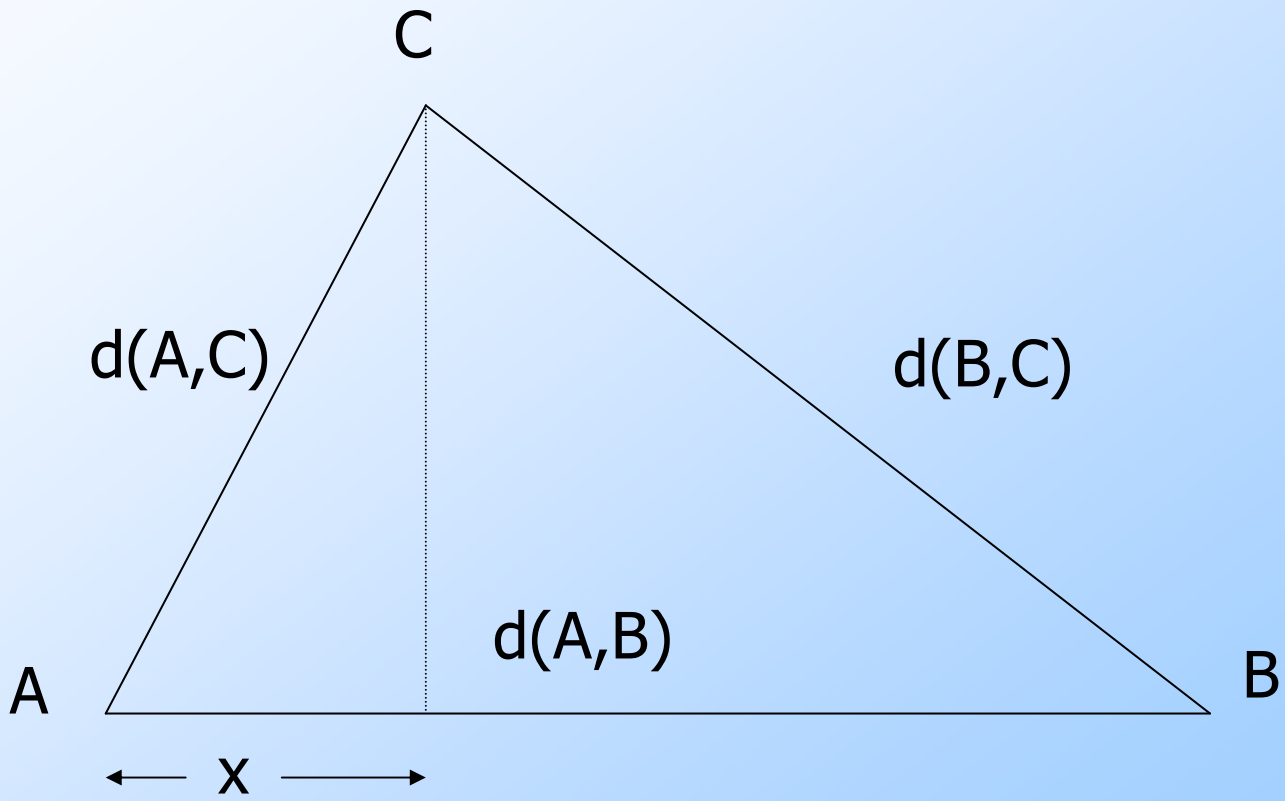
Scalarea multidimensionala

- Problema este ca in cazul unui numar mare de puncte rezulta un numar mare de dimensiuni (spatiu hiperdimensional)
- Ideal ar fi sa plasam cat mai exact cele N puncte intr-un spatiu avand K dimensiuni unde K << N.
- Acest proces se numeste scalare multidimensionala.
- Plasarea celor N puncte nu este 100% exacta (distantele calculate din coordonatele rezultate nu sunt total exacte cu cele de la care s-a pornit)

Scalarea multidimensionala

- Formula de baza folosita este cea prin care avand doua puncte putem afla distantele proiectiei unui al treilea punct pe segmentul format de primele doua puncte.
- Formula este obtinuta din teorema lui Pitagora generalizata (teorema cosinusului)

Proiectia lui C pe AB



$$x = (d^2(A,C) + d^2(A,B) - d^2(B,C))/(2d(A,B))$$

Fastmap

- Algoritmul Fastmap este unul dintre algoritmii de scalare multidimensionala.
- Acesta este un algoritm prin care se calculeaza succesiv coordonatele punctelor, cate o coordonata (o dimensiune) la fiecare pas.
- Pasul algoritmului este urmatorul:

Fastmap

1. Se aleg doua puncte aflate la distanta cat mai mare, a si b. Acestea devin o axa de coordonate (cu originea in a).
2. Pentru orice punct c din cele N se calculeaza coordonata pe aceasta axa conform formulei anterioare:

$$x = (D^2(a, c) + D^2(a, b) - D^2(b, c)) / (2 * D(a, b))$$

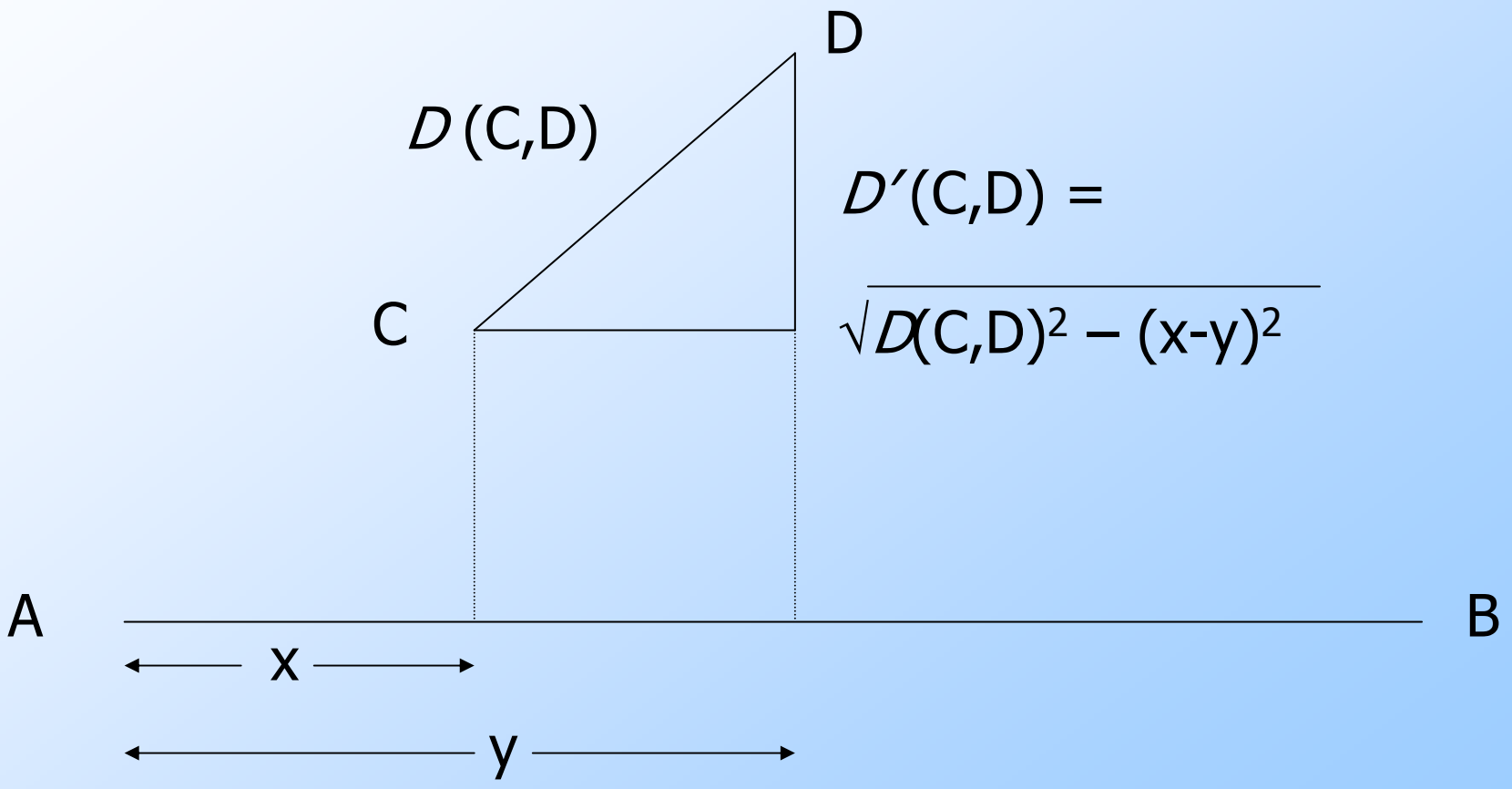
Fastmap

3. Pentru urmatoarele axe se vor folosi nu distantele initiale dintre puncte ci distante reportate in modul urmator:

$$D'^2 = D^2 - (x - y)^2$$

4. Procesul se sfarseste dupa calculul numarului dorit de coordonate sau cand nu mai pot fi alese noi axe de coordonate.

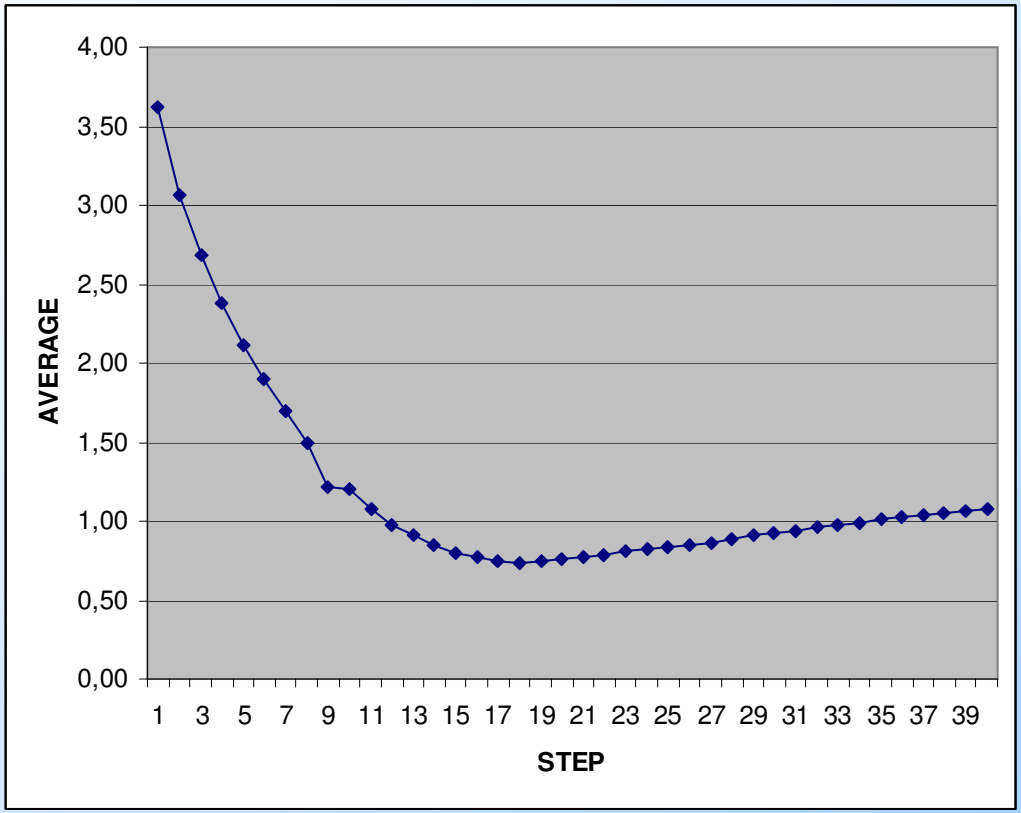
Fastmap



Probleme in cazuri reale

- In cazurile reale (matricea nu este euclidiana) se poate intampla ca patratul lui D’ calculat cu formula anterioara sa dea un numar negativ.
- In astfel de cazuri pentru a putea continua se poate lua D’ = 0.
- Aceasta alegere duce insa la erori care se propaga.
- Iata un exemplu de rulare pentru algoritm in cazul in care sunt considerate 2000 de noduri.

Probleme in cazuri reale



Probleme in cazuri reale

- Figura arata media diferentei intre Dreal si Dcalculat unde:
- Dreal este distanta intre puncte de la care s-a pornit (cunoscuta prin ipoteza problemei)
- Dcalculat este distanta dintre puncte calculata pe baza coordonatelor obtinute pana la pasul respectiv.
- Se observa ca exista un minim dupa 18 pasi (spatiu optim are deci pentru acest exemplu 18 dimensiuni, k=18)

Probleme in cazuri reale

- In mod normal graficul ar trebui sa tindda asimptotic catre 0.
- Faptul ca nu se intampla asa e datorat erorii induse de considerarea lui D’ = 0 in cazul in care patratul este negativ.
- Erorile acumulate fac ca dupa al 18-lea pas graficul sa inceapa sa creasca.

Bibliografie

- J.D.Ullman - CS345 --- Lecture Notes, Clustering I, II

  http://infolab.stanford.edu/~ullman/cs345-notes.html

Sfârşitul capitolului 9