Sovellusohjelmointi Matlab-ympäristössä: Vertaisverkon koneiden klusterointi

28.4.2013

Annemari Auvinen
(annauvi@st.jyu.fi)

Anu Niemi
(anniemi@st.jyu.fi)

Sisällysluettelo

1 Johdanto

Työmme aiheena oli Chedar-vertaisverkossa olevien koneiden luokittelu käyttäen Kmeans-algoritmia ja lisäksi havaintoavaruuden dimension pienentäminen pääkomponenttianalyysilla luokittelun tulosten esittämiseksi graafisesti.

Chedar on resurssien etsintään kehitetty vertaisverkkoalusta, jossa verkon koneet kommunikoivat keskenään ilman keskitettyä pistettä. Chedar on Agora Centerin Cheese Factory –projektin tuotos ja tätä työtä voidaan mahdollisesti käyttää hyväksi projektin myöhemmissä tutkimuksissa.

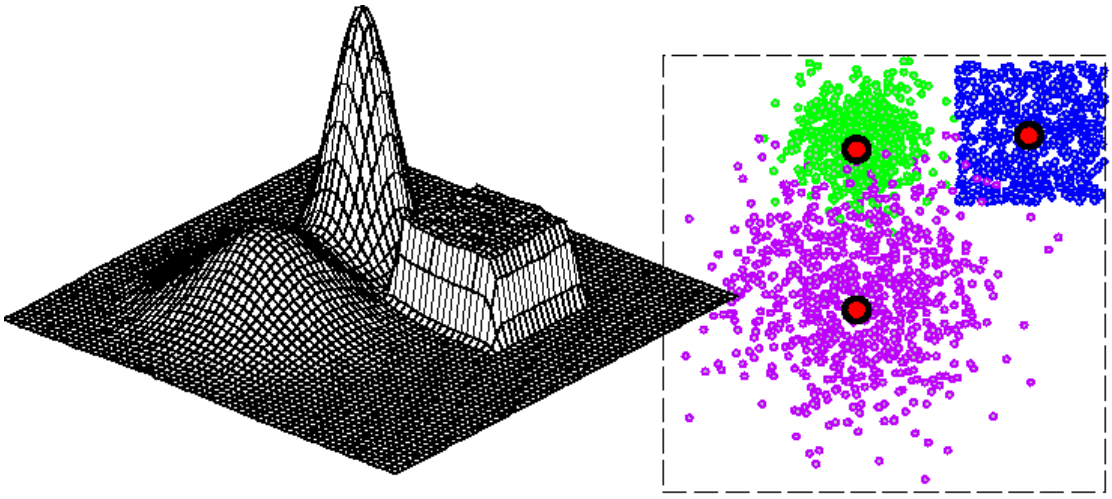
Kmeans-algoritmista tehtiin C-kielellä Matlabin mex-funktio. David Corneyn Matlab-toteutusta käytettiin mex-toteutuksen oikeellisuuden testaukseen [2]. Työssämme klusteroimme data-aineiston kummallakin toteutuksella ja vertailemme graafisesti näin saatuja tuloksia keskenään.

2 Käytetyt menetelmät

2.1 Kmeans-algoritmi

Kmeans tunnetaan myös yleistettynä Lloyd-algoritmina tai Linde-Buzo-Gray (LBD) -algoritmina.[1]

Kmeans-algoritmi jakaa havainnot eli data-aineiston luokkakeskipisteiden eli prototyyppien kesken siten, että kuhunkin luokkaan kuuluvat ne pisteet, jotka ovat sitä lähinnä. Kukin prototyyppi siis kaappaa osan havaintoavaruudesta itselleen, muodostaen nk. Voronoi-jaotuksen. [1]



Kuva 1: Kuvassa data-aineisto jakautuu kolmeen ryppääseen. Yleisesti jako on harvoin selkeä, koska kaikkia data-aineiston alkioita ei voida yksiselitteisesti määrätä kuuluvaksi johonkin ryppääseen.

Käytimme työssämme seuraavaa Kmeans-algoritmia: [1]

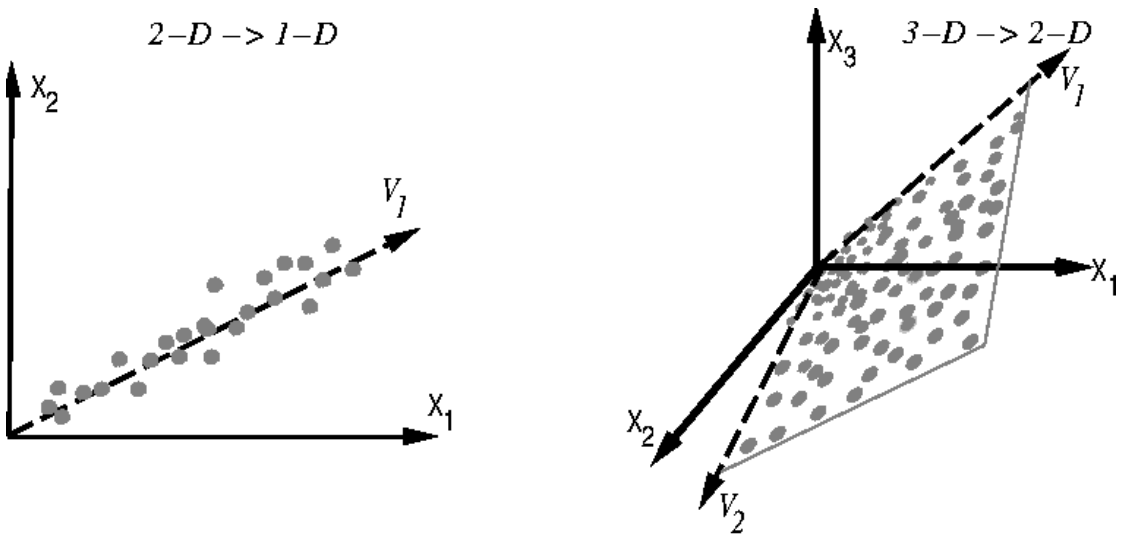
1. Alusta satunnaisesti (*t=1*)
   $w_i^{(t)} = X_j,\ j = rand(1,...,n), i = (1,...,k)$
2. Voronoi kuuluvuudet ryppäille
   $S_i = \{l \mid 1 <= l <= n, d(X_l, w_i^{(t)}) \le d(X_l, w_j^{(t)}) \forall j = 1,...,k,\ j \neq i\}$, *i=1, …, k*
3. Laske keskipisteet
   $$w_i^{(t+1)} = \frac{1}{\#S_i} \sum_{j=1}^{n} X_l, l \in S_{i,j}, i = (1,...,k)$$
4. Lopetetaan, jos ehto
   $\forall i \parallel w_i^{(t+1)} - w_i^{(t)} \parallel \le \varepsilon, i = (1,...,k)$
   täyttyy, muuten $t = t+1$ ja jatketaan kohdasta 2.

Algoritmissa *X* on data-aineiston sisältävä *n*m* matriisi, jonka riviin *j* viitataan algoritmissa X_j. *K* on haluttu klustereiden lukumäärä ja *w* on *k*m* matriisi, jossa on klustereiden keskipisteet. Alussa keskipisteiksi alustetaan satunnaisesti valitut data-aineiston alkiot. Tämän jälkeen data-aineiston alkiot jaetaan ryppäiksi niin, että kukin data-aineiston alkio kuuluu sitä lähinnä olevan keskipisteen identifioimaan ryppääseen. *K*n* kokoinen *S*-matriisi sisältää tiedon siitä, mihin ryppääseen kukin data-aineiston alkio kuuluu. *S*-matriisissa rivinumero kertoo klusterin keskipisteen rivin *w*-matriisissa ja alkion arvo ryppääseen kuuluvan data-alkion rivin *X*-matriisissa. Seuraavaksi lasketaan kullekin ryppäälle uusi keskipiste ja verrataan uusia keskipisteitä vanhoihin keskipisteisiin.

2.2 Pääkomponenttianalyysi (PCA)

Pääkomponenttianalyysissa (*Principal Components Analysis*) data-avaruuteen määritellään uudet muuttujat, ja data esitetään uusien muuttujien lineaarikombinaationa. PCA:n avulla selvitetään, mikä havaintoaineistosta on olennaista ja tärkeää informaatiota. Tavoitteena on siis vähentää havaintoavaruuden dimensiota niin, että mahdollisimman vähän informaatiota menetetään. [1], [3]

Analyysissa lasketaan kovarianssimatriisin ominaisarvot ja niitä vastaavat ominaisvektorit. Pääkomponentit on tapana järjestää merkittävimmästä (suurimmasta) vähiten merkittävään (pienimpään) ominaisarvojen mukaan. Suurinta ominaisarvoa edustava komponenttia sisältää muita enemmän informaatiota jakaumasta ja pienin taas vähiten. Pääkomponentit ovat aina keskenään ortogonaalisia. Valitsemalla joukko suurimpia ominaisarvoja ja niitä vastaavia ominaisvektoreita saadaan pääkomponenttien virittämä aliavaruus (eli pääaliavaruus). [1], [3]



Kuva 2: Vasemmanpuoleisessa kuvassa havaintoavaruuden dimensio on pudotettu kahdesta yksiulotteiseksi, oikeanpuoleisessa kolmesta kahteen.

Lasketuista ominaisvektoreista valitaan haluttu määrä pääkomponetteja, joista muodostetaan matriisi *V*. Merkitään datamatriisia *X*:llä. Datan projisointi kannan *V* suhteen saadaan laskemalla

$Y = X*V.$

3 Data-aineisto

Data-aineisto luetaan tekstimuotoisesta tiedostosta, jossa vertaisverkon koneiden tiedot on riveittäin ja arvot on eroteltu välilyönnein. Kustakin koneesta on tiedostoon kerätty seuraavat arvot:

- Naapurien lukumäärä,
- koneelta lähtevien kaikkien yhteyksien pienin viive millisekunteina,
- koneen kautta kulkevan resurssikyselyliikenteen määrä kilotavuina,
- resurssikyselyihin lähetettyjen vastausten lukumäärä sekä
- tarjolla olevien resurssien lukumäärä.

Esimerkiksi kymmenen koneen tiedot sisältävä tiedosto voisi olla seuraavanlainen.

```
2 448 3067 1 5
5 401 9201 25 16
2 409 1851 9 8
3 403 4210 2 10
2 401 2316 5 3
1 426 4210 6 12
2 403 6161 7 9
1 489 0 1 2
1 478 1890 15 11
```

4 Toteutuksesta

Työssämme toteutimme kolme funktiota: Kmeans, PCA sekä tulosten vertailussa käytetyn etäisyyksiä laskevan distance-funktion. Näiden lisäksi toteutimme myös käyttöliittymän.

4.1 Kmeans

```
[centres,classes]=kmeans(k,data)
```

Tekemällemme C-kieliselle Kmeans-funktiolle viedään parametreina klustereiden lukumäärä `k` sekä data-aineisto matriisissa `data`. Funktio on toteutettu luvussa 2.1 mainitun algoritmin mukaisesti. Se palauttaa klustereiden keskipisteet `centres`-matriisissa sekä matriisin `classes`, joka sisältää tiedon siitä, mihin luokkaan kukin data-alkio kuuluu. Algoritmin suorituksen lopetusehtona on epsilon, jonka arvoksi on määritetty 0.0001.

Funktiomme toteuttaa mex-rajapinnan:

```
void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[])
```

Parametri nlhs kertoo ulostulevien parametrien lukumäärän. *plhs[] on osoitin ulostuleviin parametreihin. Parametri nrhs kertoo sisääntulevien parametrien lukumäärän ja *prhs[] on osoitin näihin sisääntuleviin parametreihin.

```
[classes,centres,finalDistance]=dcKMeans(data,k,initCentres,maxIters)
```

Corneyn funktio tarvitsee parametreina `data`-matriisin sekä klustereiden lukumäärän `k`. Lisäksi sille voidaan tarvittaessa viedä matriisi `initCentres`, jossa on alkuarvaus keskipisteistä sekä iteraatioiden maksimilukumäärä `maxIters`. Funktio palauttaa `classes`-matriisissa tiedon siitä, mihin luokkaan kukin data-aineiston alkio kuuluu, `centres`-matriisissa luokkien keskipisteet sekä `finalDistance`-matriisissa kunkin alkion etäisyyden keskipisteestä. Corneyn funktion lopetusehtona on iteraatioiden maksimimäärä, joka on oletuksena alustettu 500. [2]

4.2 PCA

```
[z]=PCA(data)
```

PCA-funktion parametrina on käsiteltävä datamatriisi `data`, jossa on oltava vähintään kolme saraketta. Mikäli sarakkeita on vähemmän, tulostuu virheteksti ja funktion suoritus lopetetaan. Datasta lasketaan ensin kovarianssimatriisi, josta lasketaan ominaisarvot ja niitä vastaavat ominaisvektorit. Kolmesta suurimmasta pääkomponentista muodostetaan oma matriisi `z`, jonka funktio palauttaa.

4.3 Distance-funktio

```
[dist]=distance(a,b)
```

Distance-funktiolle viedään parametreina keskipisteet sisältävät matriisit `a` ja `b`. Funktio järjestää matriisien rivit suuruusjärjestykseen käyttäen Matlabin `sortrows`-komentoa ja tämän jälkeen laskee keskipistematriisin kunkin alkion etäisyyden toisen matriisin vastaavasta alkiosta ja palauttaa tuloksen matriisissa `dist`.

4.4 UI

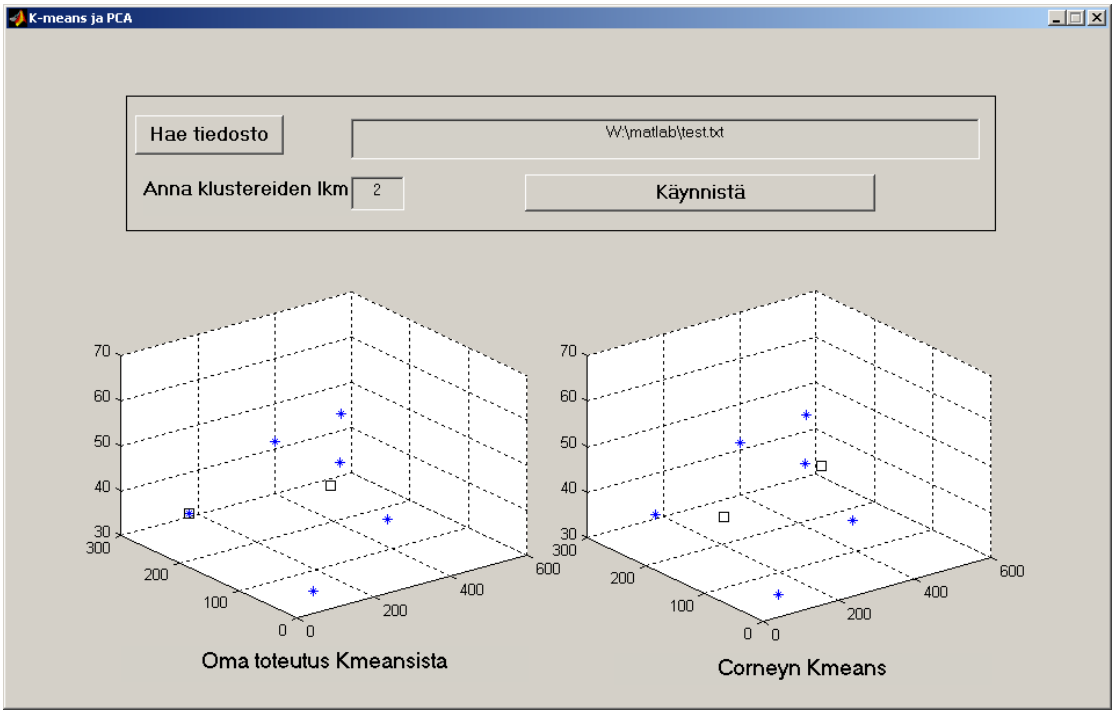
Teimme työtämme varten yksinkertaisen graafisen käyttöliittymän. Käyttöliittymässä käyttäjä saa valita tiedoston, josta data-aineisto luetaan sekä määritellä klustereiden lukumäärän.

Klusteroinnin ja pääkomponenttianalyysin suorittamisen jälkeen käyttöliittymään piirtyy vierekkäin kummankin Kmeans-funktion antamat klustereiden keskipisteet ja data projisoituina pääkomponenttiakseleille.

5 Tulosten esittely ja johtopäätökset

Corneyn Kmeans-funktiosta löysimme virheen. Se ei ottanut huomioon, että on mahdollista, että klusteriin kuuluu vain yksi piste. Funktio laski tässä tapauksessa klusterin keskipisteen väärin. Ohjelmassamme käytimme toteutusta, josta tämä virhe korjattiin. Lisäksi mielestämme funktiossa oli se epäkohta, että iteraatioiden maksimilukumäärää ei voinut asettaa, jos ei vienyt myös alkuarvausta. Kokeilimme ohjelmaa myös asettamalla koodissa olevan lukumäärän suuremmaksi.

Sekä meidän että Corneyn Kmeans-funktiot antoivat samat arvot, jos meidän funktion antamat keskipisteet vietiin alkuarvauksena Corneyn funktiolle. Mikäli Corneyn funktiolle ei viety alkuarvauskeskipisteitä, funktiot saattoivat antaa vastauksena myös eri keskipisteet. Suorittamalla ohjelma useampia kertoja peräkkäin oli havaittavissa keskipisteiden ”heiluntaa” myös saman funktion sisällä. Heilunnalla tarkoitamme sitä, että keskipisteille saattoi tulla suorituskerroilla esimerkiksi kolme erilaista vaihtoehtoa k_1, k_2 ja k_3. Niissä tilanteissa, joissa Corneyn ja meidän toteutuksemme antoivat eri vastaukset, keskipisteet kuuluivat kuitenkin näiden vaihtoehtojen joukkoon. Esimerkiksi ensimmäisellä kerralla meidän funktio antoi vaihtoehdon k_1 ja Corneyn funktio k_3 ja toisella ajokerralla meidän funktio antoi vaihtoehdon k_3 ja Corneyn funktio k_1. Tämän aiheuttaa se, että yleensä pisteet eivät jakaudu selvästi ryppäisiin, vaan on pisteitä, joiden kuuluvuutta johonkin ryppääseen ei voida yksiselitteisesti määrätä. Tällöin pisteen vaihtuminen toiseen ryhmään kuuluvaksi vaikuttaa keskipisteisiin. Ongelma ilmenee kuvassa 3, jossa omassa toteutuksessamme toisen klusterin muodostaa vain yksittäinen piste, kun taas Corneyn toteutuksessa molempiin klustereihin kuuluu useampia pisteitä.



Kuva 3: Kuva eri tuloksen antavista klusteroinneista. Klustereiden keskipisteet on kuvassa merkitty neliöillä ja data-alkiot tähdillä.

Lähteet

[1] Pasi Koikkalainen, Tilastollisen hahmontunnistuksen perusteet 2000 - luentomoniste, saatavilla WWW-muodossa osoitteessa <URL : http://erin.mit.jyu.fi/pako/kurssit/th2000/ >, 17.2.2003

[2] David Corney, Clustering with Matlab , saatavilla WWW-muodossa osoitteessa <URL: http://www.cs.ucl.ac.uk/staff/D.Corney/ClusteringMatlab.html >, 17.2.2003

[3] Petri Nokelainen, BayMiner-esitys, saatavilla WWW-muodossa osoitteessa <URL: http://www.uta.fi/laitokset/aktk/l40/luennot/bayes/bayminer/sld003.htm >, 19.2.2003