# Kategorisen datan ryhmittely

Antti Pöllänen
Opiskelijanumero: 151360

3.12.2009

Joensuun yliopisto
Tietojenkäsittelytieteen laitos
Erikoistyö

# Sisältö

# 1 Johdanto

Ryhmittelyn avulla pyritään löytämään tietoa datan rakentaan ja sitä voidaan hyödyntää useissa eri sovelluksissa. Ryhmiteltävät aineistot voivat sisältää monenlaista dataa. Yksi tärkeä osa-alue on kategorinen data. Kategorinen data tarkoittaa dataa, jossa arvojoukko on äärellinen ja etäisyyksiä eriarvojen välillä ei ole määritelty. Käytännössä kategoristen arvojen välillä voidaan tehdä vain yhtäsuuruus-vertailuja. Joissain tapauksissa myös järjestyksen vertaileminen on mahdollista. Operaatioiden rajallisuus tekee kategorisen datan ryhmittelystä haasteellista. Lisähaasteena on usein aineistojen suuri koko ja korkea ulotteisuus. Kuitenkin kategorisen datan ryhmittely on tärkeä tehtävä ja sitä voidaan hyödyntää erityisesti tiedonlouhinnassa, jossa kerätty data on usein kategorista.

Tämän työn tarkoituksena on luoda perustyökalut sekä joitain valmiita menetelmiä kategorisen datan käsittelyyn ja ryhmittelyyn. Valmiit menetelmät ovat aikaisempiin julkaisuihin perustuvia ja ovat sellaisenaan valmiita käytettäväksi. Lisäksi ohjelman lähdekoodi on vapaasti saatavilla ja se sisältää useita valmiita funktioita, joita tarvitaan usein kategorisen datan ryhmittelyssä. Näin ollen kuka tahansa voi jatkokehittää menetelmiä, jos valmiit menetelmät eivät anna toivottuja tuloksia.

Ohjelma on kirjoitettu C-kielellä ja perustuu Joensuun yliopiston SIPU (Speech and Image Processing Unit) -ryhmän aiempiin toteutuksiin.

# 2 Käyttöönotto

Helpoiten kaikki ohjelmat saa käännettyä ajamalla *bin* hakemistossa *create.sh* skriptin:

```
./create.sh
```

Suorituksen jälkeen hakemistossa on löytyy paketin mukana tulevat ohjelmat. On kuitenkin huomioitava, että kääntämistä varten tarvitset C-kääntäjän ja make-ohjelman.

Tarvittaessa lähdekoodi löytyy hakemistosta *src*. Jokainen ohjelma on sijoitettu omaan alihakemistoon. Alihakemistosta on ohjelman lähdekoodi ja makefile kääntämistä varten. Varsinainen kääntäminen tapahtuu suorittamalla ohjelman alihakemistossa komento:

```
make
```

Komennon jälkeen ohjelma on valmis käytettäväksi.

# 3 Käyttöohje

Tässä luvussa esitellään valmiiden menetelmien käyttöä peruskäyttäjän näkökulmasta. Ohjelmien käyttö ei edellytä ohjelmointiosaamista.

## 3.1 Ryhmittelymenetelmät

Paketti sisältää kolme eri moduulia, jotka sisältävät valmiita ryhmittelymenetelmiä: CBACE, CBCKM, CBROCK.

Kaikki ryhmittelymenetelmät ottavat parametrina aineiston tiedostonimen ja osituksen tiedostonimen. Lisäksi menetelmille voidaan määrittää yleisiä valitsimia, joita on esitelty tarkemmin luvussa 3.6. Lisäksi menetelmillä on joukko menetelmäkohtaisia valitsimia.

### 3.1.1 CBACE

*ACE* [2] on agglomeratiivinen ryhmittelymentelmä, jossa ryhmien yhdistämiskustannuksena käytetään entropian kasvua. ACE ryhmittelymenetelmän etuna on, ettei sille ole pakko määritellä ylimääräisiä valitsimia mahdollisten yleisten valitsimien lisäksi. Valitsimen $-Mn$ avulla saadaan tuotettua useampi ryhmittelyjä yhdellä kertaa. Valitsin $-Mn$ määrittää suurimman ryhmien lukumäärän ja yleinen valitsin $-Cn$ määrittää pienimmän ryhmien lukumäärän. ACE tuottaa kaikki eri kokoiset ryhmittelyt tältä väliltä. Ilman valitsinta $-Mn$ tuottaa vain yhden ryhmittelyn.

Esimerkki ACE menetelmän käytöstä:

```
./cbace mushroom.ts mushroom.pa
./cbace -Cn=1 -Mn=30 mushroom.ts mushroom.pa
```

### 3.1.2 CBCKM

*CBCKM* on sisältää joukon k-meansin perusajatusta hyödyntäviä ryhmittelymenetelmiä. Tämä perusajatus on korjata vuorotellen ositusta perustuen ryhmien edustajiin ja ryhmien edustajia perustuen ositukseen.

Ryhmittelymenetelmä valitaan valitsimella $-Mn$, jonka mahdollisia arvoja ovat:

- 0 = K-entropies (oletus)
- 1 = K-distributions [1]
- 2 = K-histograms [3, 6]
- 3 = K-medoids [7]
- 4 = K-modes [4]
- 5 = Delta k-entropies
- 6 = Hybrid k-entropies
- 7 = Summary table k-medoids
- 8 = Distance table k-medoids

Ryhmittelymenetelmän lisäksi on valittava ryhmien lukumäärä valitsimelle $-Cn$ ja alustusmenetelmä valitsimille $-In$.

Lisäksi on mahdollista valita käytettävien iteraatioiden lukumäärä valitsimella $-Kn$. Jos iteraatioiden lukumäärää ei valita, ryhmittelyä jatketaan kunnes muutoksia ei enää tapahdu. Saman voi ilmaista eksplisiittisesti valitsemalla iteraatioiden lukumääräksi 0.

### 3.1.3 CBROCK

*ROCK* [5] on agglomeratiivinen ryhmittelymentelmä, jossa ryhmien yhdistämiskustannus perustuu ryhmien välisiin linkkeihin. Yleisten valitsimien lisäksi ROCK menetelmälle täytyy määritellä parameterina $\theta$-arvo. Arvo on todellisuudessa väliltä $0 - 1$, mutta ohjelmassa määritellään valitsimelle Tn arvo väliltä $0 - 1000$. Annettu Tn-arvo jaetaan tuhannella, jolloin se skaalautuu välille $0 - 1$.

Esimerkki ROCK menetelmän käytöstä arvolla $\theta = 0.7$:

```
./cbrock -Tn=700 mushroom.ts mushroom.pa
```

ROCK menetelmä sisältää poikkeavien arvojen käsittelyn. Normaalisti menetelmä tulostaa poikkeamat sisältävän ryhmän, mutta poikkeamista voi myös kirjoittaa lisätietoa tiedostoon valitsimella $-Z$. Tällöin ROCK tuottaa tiedoston *outliers*.

Kuten ACE menetelmässä, ROCK menetelmässässa voidaan valitsimen $-Mn$ avulla saadaan tuotettua useampi ryhmittelyjä yhdellä kertaa. Valitsin $-Mn$ määrittää suurimman ryhmien lukumäärän ja yleinen valitsin $-Cn$ määrittää pienimmän ryhmien lukumäärän. ROCK tuottaa kaikki eri kokoiset ryhmittelyt tältä väliltä. Tässä tapauksessa ROCK tuottaa ryhmittelyjä, joissa on aina poikkeamat sisältävä ryhmä 1. Myös silloin kun tämä ryhmä on tyhjä.

## 3.2 Arviointimenetelmät

*CBCEVAL* ohjelma sisältää ryhmittelytulosten arviointimenetelmiä. Arviointimenetelmän valinta tapahtuu valitsimella $-Mn$. Arviointimenetelmät ottavat parametrina ryhmittelyyn käytetyn aineiston sekä ryhmittelytuloksen. Poikkeuksena on luokitteluvirhe, joka saa parametrinaan oikean luokkajaon sisältävän aineiston. Vektoreiden oikeat luokat on kirjoitettu ensimmäisen attribuutin arvoksi. Tämä ero perustuu siihen, että luokitteluvirhe on ulkoiseen kriteeriin perustuvat arviointimenetelmä. Tiedostomuoto on kuitenkin kaikissa tapauksissa sama. Taulukossa 1 on parametrien sisältö.

| Virhetyyppi | Valitsin | Ryhmittelytulos | Aineisto |
|---|---|---|---|
| Entropia (oletus) | -Mn=0 | Ositus | Alkuperäinen |
| Category utility | -Mn=1 | Ositus | Alkuperäinen |
| Luokitteluvirhe | -Mn=2 | Ositus | Oikea luokkajako |

Taulukko 1: Arviointimentelmien parametrit

Lisäksi arviointimenetelmät kykenevät jättämään yhden ryhmän kokonaan huomioimatta arvioinnissa. Tämä ominaisuus on poikkeamien käsittelyä varten. Huomiotta jätettävä ryhmä määritellään valitsimella $-On$, jonka arvoksi asetetaan ryhmän numero.

Esimerkkejä arviointimenetelmien käytöstä:

```
./cbceval mushroom.pa mushroom.ts
./cbceval -Mn=1 -On=1 mushroom.pa mushroom.ts
./cbceval -Mn=2 mushroom.pa mushroom_labels.ts
```

Oletusarvoisesti kaikki arviointimenetelmät antavat tuloksena ainoastaan yhden luvun desimaaliluvun, joka kuvastaa ryhmittelyn laatua.

## 3.3 Testausmenetelmät

Testausta varten löytyy skriptejä hakemistosta *testenv*. Ryhmittelymenetelmistä poiketen testausympäristöä ei ole pyritty tekemään täysin kiinteäksi ja helppokäyttöiseksi, koska testausympäristö on ainoastaan ohjelmistokehittäjiä varten. Skriptit on tehty toimimaan vain Unix/Linux-ympäristössä ja on todennäköistä, että niitä joudutaan räätälöimään eri ympäristöihin ennen käyttöä.

Testauksessa käytetään asetustiedostoja, joissa määritellään testauksessa käytettävät parametrit. Parametrien avulla määritetään muun muassa käytettävä ryhmittelymenetelmä, ryhmien lukumäärä ja iteraatioiden lukumäärä. Asetustiedostoissa käytetään päätettä cp, joka viittaa sanoihin clustering properties. Menetelmää käytetään kirjoittamalla seuraava komento:

```
./run.sh mushroom.cp
```

Tarkempia tietoja asetuksista ja testauksesta saa tarkastelemalla esimerkkitiedostoa *mushroom.cp*.

## 3.4 Aineistojen luonti

Aineistojen luontia varten kannattaa käyttää Marko Tuonosen kehittämää CB2TXT-moduulia, joka löytyy *src/cb2txt* hakemistosta. Tällä ohjelmalla saadaan luotua CB-formaatin mukainen aineisto tekstitiedostosta. Ohjelman käyttöohjeet löytyvät sen omasta dokumentaatiosta.

CB-formaatin mukainen aineisto on sellaisenaan valmis käytettäväksi, mutta tämän lisäksi kannattaa skaalata aineiston arvot. Koska data on kategorista, ei etäisyyksiä ole määritelty. Esimerkiksi jos attribuutti saa arvoja 10, 20 ja 30, voidaan yhtä hyvin käyttää arvoja 1, 2, 3. Skaalaus nopeuttaa joitain ryhmittelymenetelmiä. Skaalausta varten on CBCTK-moduulissa *catscale*-ohjelma, jolle annetaan parametrina olemassa oleva aineisto sekä tiedostonimi, johon skaalattu aineisto kirjoitetaan. Esimerkiksi:

```
./catscale mushroom.ts mushroom_scaled.ts
```

## 3.5 Näytteistäminen

Erittäin suurien aineistojen ryhmittelyssä voidaan hyödyntää näytteistämistä. Toisin sanoen alkuperäisestä aineistosta valitaan pieni osajoukko eli näyte, joka ryhmitellään. Lopuksi koko aineisto sijoitetaan ryhmiin, jotka on saatu näytteen avulla. Näytteistämisessä on kolme vaihetta

- Näytteen ottaminen
- Näytteen ryhmittely
- Alkuperäisen aineiston sijoittaminen ryhmiin

Näytteen ottamista varten on CBCTK-moduulissa *catsample*-ohjelma. Sille annetaan parametrina aineisto ja näytteen koko. Tuloksena syntyy halutun kokoinen näyte. Esimerkiksi sieniaineistosta voitaisiin ottaa 100 vektorin näyte seuraavasti:

```
./catsample -Sn=100 mushroom.ts mushroom_sample.ts
```

Aineiston ryhmiin sijoittamista varten on CBCTK-moduulissa *catmap*-ohjelma. Ohjelma saa parametrina näytteen ja näytteen ryhmittelyn sekä alkuperäisen aineiston. Tuloksena syntyy näiden tietojen perusteella alkuperäisen aineiston ryhmittely. Esimerkiksi sieniaineiston näytteestä ja sen ryhmittelystä sekä alkuperäisestä sieniaineistosta saataisiin ryhmittely seuraavalla komennolla:

```
./catsample mushroom_sample.ts mushroom_sample.pa mushroom.ts mushroom.pa
```

## 3.6 Komentovalitsimien käyttö

Valitsimien avulla määritellään ohjelman parametreja. Valitsimet sijoitetaan heti käytetyn komennon perään ennen muita parametereja. Valitsimien järjestyksellä ei ole merkitystä. Valitsin alkaa viivalla $(-)$, seuraavaksi määritellään valitsimen nimi esimerkiksi (esim. $Cn$) ja tämän jälkeen tulee tarvittaessa tarkentava lisämääre (esim. $= 10$). Totuusarvot eivät vaadi lisämääreitä, vaan valitsimen käyttö tai käyttämättä jättäminen antaa tarvittavan tiedon. Esimerkiksi ACE ryhmittely, jossa määritellään ryhmien lukumääräksi 10 ja ylikirjoitetaan ositustiedosto, jo se on jo olemassa:

```
./cbace -Cn=10 -O mushroom.ts mushroom.pa
```

Lähes jokaisesta ohjelmasta löytyy seuraavassa listauksessa 2 esitetyt valitsimet.

Kattavat listaukset ohjelmien valitsimista saa ajamalla ohjelmat ilman parametreja, jolloin tulosteena saa ohjelman käyttöohjeen.

| Valitsin | Selitys | Tyyppi | Käyttö |
|---|---|---|---|
| -Cn | Ryhmien lukumäärä | Kokonaisluku | -Cn=10 |
| -O | Ylikirjoita tulos, jos tiedosto on jo olemassa | Totuusarvo | -O |
| -Qn | Hiljaisuustaso | Kokonaisluku | -Qn=2 |

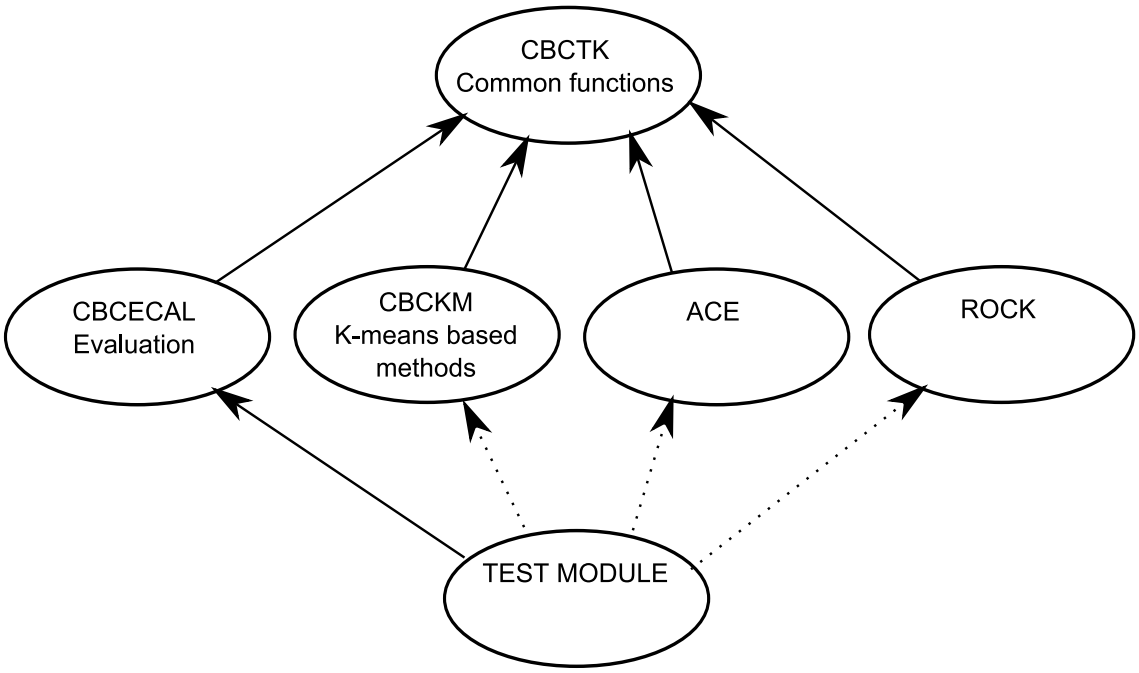
Taulukko 2: Yleiset komentovalitsimet

# 4 Ohjelmointirajapinta

Tässä luvussa kuvataan ohjelman tietorakenteita ja funktioita sovelluskehittäjän näkökulmasta.

## 4.1 Yleiskuvaus

Kuvassa 1 on esitetty eri moduulit ja niiden riippuvuussuhteet. Valinnaiset riippuvuudet on merkitty katkoviivalla.




Kuva 1: Moduulit ja niiden väliset riippuvuussuhteet

CBCTK on kategorisen ryhmittelyn kirjastomoduuli. Se sisältää suuren joukon hyödyllisiä tietorakenteita ja funktioita, jotka on esitelty tarkemmin luvuissa 4.2 ja 4.4. Moduulit CBCEVAL, CBCKM, CBACE ja CBROCK on toteutettu käyttämällä kirjastomoduulia kuten kuvasta 1 voidaan päätellä. CBCEVAL sisältää arviointimenetelmät, CBCKM sisältää K-means pohjaiset menetelmät, CBACE sisältää ACE menetelmän ja CBROCK sisältää ROCK menetelmän. On syytä huomata, että nämä moduulit tarvitsevat kirjastomoduulia ainoastaan käännösaikana. Kun moduulit on käännetty, ne ovat käytettävissä itsenäisesti.

Testausmoduuli hyödyntää arviointimenetelmiä sekä valinnaisesti ryhmittelymenetelmiä. Ryhmittelymenetelmien on oltava saatavilla ajonaikana.

Ainoastaan kirjastomoduulin ohjelmointirajapintaa on esitelty tässä dokumentissa. Kirjastomoduuli on ainoa, joka on varsinaisesti tarkoitettu uudelleen käytettäväksi.

## 4.2 Tietorakenteet

Kategorista dataa varten on määritelty omia tietorakenteita CBCTK-kirjastomoduulissa. Tietorakenteita kannattaa muokata aina käyttämällä funktioita, jotka on kuvattu luvussa 4.4. Tietorakenteita voi kuitenkin joutua muokkaamaan silloin kun käytettävät funktiot eivät mahdollista haluttua toteutusta tai jos halutaan laajentaa kirjastomoduulia.

### 4.2.1 CATCODEBOOK ja sisältyvät tietorakenteet

**CATCODEBOOK** on tietorakenne kategorisen datan ryhmän edustajille.

```
typedef struct
  {
  CODEBOOK *pCodebook;
  CENTROIDINFO *CentroidInformation;
  int *Categories;
  } CATCODEBOOK;
```

pCodebook - Osoitin kategorista edustajistoa vastaavaan tavalliseen edustajisto.
CentroidInformation - Taulukko yksittäisien ryhmien edustajista.
Categories - Taulukko eri attribuuttien kategorioiden lukumääristä.

**CENTROIDINFO** sisältää tiedot ryhmän edustajasta

```
typedef struct
  {
  DIVISION *Divisions;
  int Count;
  } CENTROIDINFO;
```

Divisions - Taulukko eri attribuuttien arvojen jakaumista.
Count - Ryhmään kuuluvien vektoreiden lukumäärä.

**DIVISION** sisältää attribuuttien arvojen lukumäärän.

```
typedef struct
  {
  int *Instances;
  } DIVISION;
```

Instances - yksittäisen attribuutin kategoriodien lukumäärätauluko.

### 4.2.2 UTM

UTM (Upper Triangular Matrix) on tietorakenne, jonka avulla voidaan helposti hallita pareja matriisimaisesti. Tietorakenteen avulla säästetään myös muistia, koska ainoastaan matriisin toinen puoli on käytössä.

```
typedef struct
  {
  int N;
  int Size;
  void** Data;
  } UTM;
```

N - matriisin sivun koko.
Size - matriisissa olevien elementtien lukumäärä $\frac{N^2-N}{2}$.
Data - Osoitin matriisi varsinaiseent tietoon.

## 4.3 Luettelotyypit

**CATINITMETHOD** luetteloi erilaiset alustusvaihtoehdot.

- RANDOMINIT - Aineiston vektorien jakaminen satunnaisesti ryhmiin.
- SIMPLEINIT - Aineisto vektorien jakaminen järjestyksessä eri ryhmiin (deterministinen).

**CATDISTANCETYPE** luetteloi erilaiset etäisyydet.

- ENTROPYDISTANCE - Vektorin siirrosta aiheutuva entropian muutos.
- JOINTPROBABILITYDISTANCE - Ryhmässä esiintyvien kategorioiden prosentuaalisten osuuksien tulo.
- LINEARDISTANCE - Ryhmässä esiintyvien muiden kategorioiden prosentuaalisten osuuksien yhteenlaskettu summa.
- SIMPLEMATCHINGDISTANCE - Yksinkertainen yhteensopivuus kerroin eli poikkeavat attribuutin arvot ryhmän edustajan ja vektorin välillä.

**CATCENTROIDTYPE** luetteloi erilaiset ryhmän edustajat.

- DISTRIBUTIONREPR - Eri kategorioiden jakaumat.
- MEDOIDREPR - Medoidi eli ryhmän vektori, jonka yhteenlaskettu etäisyys muihin ryhmän vektoreihin on pienin mahdollinen.
- MODEREPR - Moodi eli attribuuttien useimmiten esiintyvät kategoriat.
- STMEDOIDREPR - Medoidi kuten MEDOIDREPR, mutta etsintä suoritetaan yhteenvetotaulukon avulla.

**CATMESSAGETYPE** luetteloi viestien mahdolliset tasot. Arvoja käytetään CATPrint ja CATPrintV makroissa määrittämään missä tapauksissa tulostus suoritetaan.

- ERRORMESSAGE - Varmat virhetilanteet. Tulostetaan ellei tulostusta ole erikseen kielletty.
- WARNINGMESSAGE - Mahdolliset virhetilanteet. Tulostetaan ellei tulostusta ole erikseen kielletty.
- INFOMESSAGE - Lisätietoja suorituksesta. Tulostetaan käyttäjän pyynnöstä.
- DEBUGMESSAGE - Lisätietoja testausta varten. Tulostetaan käyttäjän pyynnöstä.

## 4.4 Kirjastofunktiot

Tässä luvussa on esitelty CBCTK sisältämät funktiot sekä makrot. Funktiot on jaettu alilukuihin aihepiireittäin.

### 4.4.1 Makrot

Kirjastoon on määritelty muutamia makroja käytön helpottamiseksi. Makrojen avulla voidaan kapseloida tietorakenteita, mutta tietueiden jäseniä voidaan käsitellä edelleen normaalien muuttujien tapaan. Vastaava ei onnistuisi funktioiden avulla. Lisäksi kirjastoon on määritelty tulostusmakroja tehokkuuden vuoksi.

`CategoryInstances(pCCB,prot,attr,cat)`

Viittaus ryhmässä oleviin kategorioiden lukumäärää ilmaisevaan muuttujaan.

`Categories(pCCB,attr)`

Viittaus attribuutilla eri kategorioiden lukumäärää ilmaisevaan muuttujaan.

`Count(pCCB,prot)`

Viittaus ryhmän jäsenten määrää ilmaisevaan muuttujaan.

`CATPrint(quietLevel, msgLevel, format)`

Tulostaa viestin jos viestin taso on enintään yhtä korkea kuin ohjelman hiljaisuustaso. Makro sallii ainoastaan staattisia merkkijonoja.

`CATPrintV(quietLevel, msgLevel, format, ...)`

Tulostaa viestin jos viestin taso on enintään yhtä korkea kuin ohjelman hiljaisuustaso. Makro sallii dynaamisia viestejä kuten printf-funktio.

### 4.4.2 CATBODEBOOK funktiot

```
void CATScaleTrainingSet (TRAININGSET* in, TRAININGSET* out);
```

Funktion nimeää kategoriat uudelleen käyttäen mahdollisimman pieniä positiivisia kokonaislukuja. Funktio ottaa parametrina kaksi aineistoja, joiden täytyy olla samankokoiset. Ensimmäistä aineistoa käytetään syötteenä ja toiseen aineistoon kirjoitetaan saatu tulos. Funktio ei palauta arvoa.

```
int  CATCreateNewCodebook(CATCODEBOOK *pCCB,
                          CODEBOOK *pCB,
                          TRAININGSET *pTS);
```

Funktio varaa tilan ja alustaa uuden kategorisen edustajiston. Parametrien pCB ja pTS täytyy olla alustettu oikeilla arvoilla. Funktio palauttaa arvon onnistumisesta. Nolla tarkoittaa onnistumista. Muut arvot tarkoittavat virhettä.

```
void CATFreeCodebook     (CATCODEBOOK* pCCB);
```

Funktio vapauttaa kategoriselle edustajistolle varatun tilan. Funktio ei palauta arvoa.

```
void CATClearDivisions   (CATCODEBOOK *pCCB);
```

Funktio alustaa kaikkien ryhmien edustajien kaikkien attribuuttien kaikki kategoriat nolliksi. Funktio ei palauta arvoa.

### 4.4.3 Yleisiä ryhmittelyfunktioita

```
double CATVectorDistance(CATCODEBOOK *pCCB,
                         PARTITIONING *pP,
                         TRAININGSET *pTS,
                         int centroid,
                         int vector,
                         CATDISTANCETYPE disttype);
```

Funktio laskee ryhmän ja vektorin etäisyyden parametrina määritetyllä etäisyysmitalla. Funktio palauttaa etäisyyden.

```
int CATFindNearestVector(CATCODEBOOK *pCCB,
                         PARTITIONING *pP,
                         TRAININGSET *pTS,
                         int vector,
                         CATDISTANCETYPE disttype);
```

Funktio etsii ja palauttaa lähimmän prototyypin indeksin parametrina määritetyn etäisyysmitan mukaan. Funktio palauttaa lähimmän prototyypin indeksin.

```
void CATChangePartition (CATCODEBOOK *pCCB,
                         TRAININGSET *pTS,
                         PARTITIONING *pP,
                         int from,
                         int to,
                         int vector);
```

Funktio siirtää vektorin ryhmästä toiseen. Funktio ei palauta arvoa.

```
void CATMergeClusters   (CATCODEBOOK *pCCB,
                         TRAININGSET *pTS,
                         PARTITIONING *pP,
                         int cluster1,
                         int cluster);
```

Funktio yhdistää kaksi ryhmää yhdeksi ryhmäksi ja pienentää osituksen kokoa. Funktio ei palauta arvoa.

```
void CATGenerateInitialSolution(
                         PARTITIONING *pP,
```

```
                        CATCODEBOOK *pCCB,
                        TRAININGSET *pTS,
                        CATINITMETHOD initMethod);
```

Funktio generoi osituksen parametrina määritetyn alustusmenetelmän mukaisesti. Funktio ei palauta arvoa.

```
void CATGenerateOptimalCodebook(
                        PARTITIONING *pP,
                        TRAININGSET *pTS,
                        CATCODEBOOK *pCCB,
                        CATCENTROIDTYPE representativeType);
```

Funktio generoi optimaaliset ryhmien edustajat ositukseen perustuen. Funktio ei palauta arvoa.

```
void CATGenerateMedoids( PARTITIONING *pP,
                        TRAININGSET *pTS,
                        CATCODEBOOK *pCCB,
                        UTM* distances);
```

Funktio generoi ryhmälle optimaaliset medoidit. Funktio suorittaa funktion CATGenerateOptimalCodebook erikoistapauksen. UTM* distances sisältää etäisyystaulukon, jossa puuttuvat arvot on merkattu negatiiisilla luvuilla. Funktio ei palauta arvoa.

```
int CATGenerateOptimalPartition(
                        CATCODEBOOK *pCCB,
                        TRAININGSET *pTS,
                        PARTITIONING *pP,
                        CATDISTANCETYPE disttype);
```

Funktio generoi optimaalisen osituksen koodikirjastoon perustuen. Funktio palauttaa ositukseen tehtyjen muutosten lukumäärän.

```
double CATClusterEntropy(CATCODEBOOK* pCCB, int i);
```

Funktio laskee ja palauttaa ryhmän i entropian. Entropia on valmiiksi suhteutettu ryhmän kokoon nähden.

```
double CATDatasetEntropy(CATCODEBOOK* pCCB);
```

Funktio laskee ja palauttaa koko aineiston entropian. Entropiaa ei ole suhteutettu aineiston koon. Laskutapa ei huomioi ryhmittelyä ollenkaan vaan laskennan perustana on aineistossa esiintyvien kategorioiden lukumäärät.

```
double CATExpectedEntropy(CATCODEBOOK* pCCB, TRAININGSET* pTS)
```

Funktio laskee ja palauttaa ratkaisun odotetun entropian.

### 4.4.4 Ryhmittelymenetelmät

```
int PerformCKM          (PARTITIONING *pP,
                         CATCODEBOOK *pCCB,
                         TRAININGSET *pTS,
                         int kmIter,
                         int quietLevel,
                         CATDISTANCETYPE distanceMetric,
                         CATCENTROIDTYPE representativeType,
                         CATINITMETHOD initMethod);
```

Funktio suorittaa K-means iterointia parametrina annetuilla etäisyysmetriikalla, ryhmän edustajalla ja alustusmenetelmällä. Käsiteltävä data määräytyy parametrien kautta. Nämä ovat osoitin ositukseen pP, osoitin kategoriseen koodikirjaan pCCB ja osoitin aineistoon pTS. Parametri kmIter määrää suoritettavien iteraatioiden maksimimäärän. Nolla vastaa ääretöntä eli iterointia niin kauan, kunnes muutoksia ei tule.

Funktio palauttaa totuusarvon suorituksen aikana tapahtuneista virheistä. Epätosi eli nolla tarkoittaa onnistunutta suoritusta.

### 4.4.5 UTM funktiot

```
int   UTMIndex          (UTM* pUTM, int x, int y);
```

Funktio laskee ja palauttaa oikean indeksin data-taulukossa. Laskenta perustuu tietorakenteen kokoon sekä x ja y koordinaatteihin. Huomioitavaa on, että parametrien x ja y järjestyksellä ei ole merkitystä.

```
void  UTMCreate         (UTM* pUTM, int N);
```

Funktio varaa tarvittavan tilan tietorakenteelle ja alustaa tietueen, johon parametrina tullut osoitin viittaa. Kun tietorakennetta ei enää käytetä, se tulee vapauttaa funktiolla UTMFree. Funktio ei palauta arvoa.

```
void* UTMGet            (UTM* pUTM, int x, int y);
```

Funktio palauttaa tietorakenteen osoitin-taulukossa olevan osoittimen. Taulukon indeksi määräytyy UTMIndex-funktion kutsulla, jossa parametreina välitetään x ja y.

```
void  UTMSet            (UTM* pUTM, int x, int y, void* data);
```

Funktio asettaa parametrina saadun osoittimen tietorakenteen sisältämään osoitin-taulukkoon. Taulukon indeksi määräytyy UTMIndex-funktion kutsulla, jossa parametreina välitetään x ja y. Funktio ei palauta arvoa.

```
void  UTMFree           (UTM* pUTM);
```

Funktio vapauttaa tietorakenteen käyttämän muistin. Funktio ei palauta arvoa.

### 4.4.6 Tulostusfunktiot

```
void CATPrintCluster    (CATCODEBOOK *pCCB, int cluster);
```

Funktio tulostaa ryhmän arvojakauman. Funktio ei palauta arvoa.

```
void CATPrintCodebook   (CATCODEBOOK *pCCB);
```

Funktio tulostaa ryhmittelyn kaikkien ryhmien arvojakaumat. Funktio ei palauta arvoa.

```
void CATPrintIterationInfo
                        (int quietLevel,
                        int iteration,
                        int changes);
```

Funktio tulostaa iteraatiokierroksen numeron ja muutoksien lukumäärän. Funktio ei palauta arvoa.

### 4.4.7 Sekalaiset

```
void CATPermutation     (int* permutation, int size, int p);
```

Funktio luo permutaation, joka sijoitetaan parametrina saatuun taulukkoon. Muita parametreja ovat taulukon koko sekä permutaation numero. Permutaatio on erilainen kaikilla ei-negatiivisilla p arvoilla, jotka ovat pienempiä kuin $size!$. Permutaatiot eivät ole loogisessa järjestyksessä. Funktio ei palauta arvoa.

```
int CATFactorial        (int i);
```

Funktio laskee ja palauttaa parametrina saadun luvun kertoman.

```
void InitRandom         ();
```

Funktio alustaa satunnaislukugeneraattorin. Funktio ei palauta arvoa.

# 5 Testaus

## 5.1 Arviointimenetelmät

Arviointimenetelmien testausta varten on tehty hyvin yksinkertaisia osituksia ja aineistoja. Aineistot löytyvät hakemistosta *datasets/test*. Aineistojen yksinkertaisuuden ansiosta oikeat tulokset voidaan laskea manuaalisti ja verrata näin saatuja arvoja ohjelman antamiin tuloksiin. Alla listaus luokitteluvirheen, entropian ja category utilityn testauksen tuloksista.

| Luokittelu | Ositus | Odotettu tulos | Saatu tulos |
|---|---|---|---|
| classes.ts | correct1.pa | 0 | 0.000000 |
| classes.ts | correct2.pa | 0 | 0.000000 |
| classes.ts | wrong1.pa | 0.125 | 0.125000 |
| classes.ts | wrong2.pa | 0.5 | 0.500000 |

Taulukko 3: Luokitteluvirheen testaus

| Aineisto | Ositus | Odotettu tulos | Saatu tulos |
|---|---|---|---|
| attr_simple.ts | correct1.pa | 0.693147 | 0.693147 |
| attr_simple.ts | correct2.pa | 0.693147 | 0.693147 |
| attr_simple.ts | wrong1.pa | 0.380396 | 0.380396 |
| attr_simple.ts | wrong2.pa | 0.0 | 0.000000 |

Taulukko 4: Entropian testaus

| Aineisto | Ositus | Odotettu tulos | Saatu tulos |
|---|---|---|---|
| attr_simple.ts | correct1.pa | 0.5 | 0.500000 |
| attr_simple.ts | correct2.pa | 0.5 | 0.500000 |
| attr_simple.ts | wrong1.pa | 0.3 | 0.300000 |
| attr_simple.ts | wrong2.pa | 0.0 | 0.000000 |

Taulukko 5: Category utilityn testaus

## 5.2 Ryhmittelymenetelmät

Ryhmittelymenetelmien oikeellisuuden todistaminen on huomattavasti hankalampaa ja aivan täydellistä varmuutta on hankala saavuttaa. Kuvassa 6 on testiaineistolle ja testiaineistolle käsinlaskettu tulos. K-* algoritmien alustuksessa vektorit 1-5 on asetettu ryhmään 1 ja vektorit 6-10 on asetettu ryhmään 2. Vastaavat tulokset on saatu ohjelmien avulla. Nopeutetut versiot tuottavat samat tulokset kuin vastaavat perusversiot eikä niiden toimintaa ole todistettu erikseen.

| | Dimensio 1 | Dimensio 2 | Dimensio 3 | ACE | K-distr. | K-entropies | K-histograms | K-medoids | K-modes | ROCK $(\theta=0.15)$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Vektori 1 | 2 | 3 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |
| Vektori 2 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 |
| Vektori 3 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Vektori 4 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 2 | 1 |
| Vektori 5 | 1 | 3 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Vektori 6 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 |
| Vektori 7 | 2 | 3 | 3 | 1 | 2 | 1 | 2 | 2 | 2 | 1 |
| Vektori 8 | 2 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| Vektori 9 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Vektori 10 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 2 | 1 | 1 |

Taulukko 6: Yksinkertainen testiaineisto ja menetelmien antamat tulokset

Kuvan 6 testiaineisto sekä ryhmittelyn välivaiheet löytyvät hakemistosta *doc/proves*. Lisäksi menetelmiä on testattu todellisilla sieni- ja soijapapuaineistoille, jotka löytyvät hakemistosta *datasets*. Sieniaineisto sisältää 8124 vektoria ja 21 ulottuvuutta. Soijapapuaineisto sisältää 47 vektoria ja 35 attribuuttia.

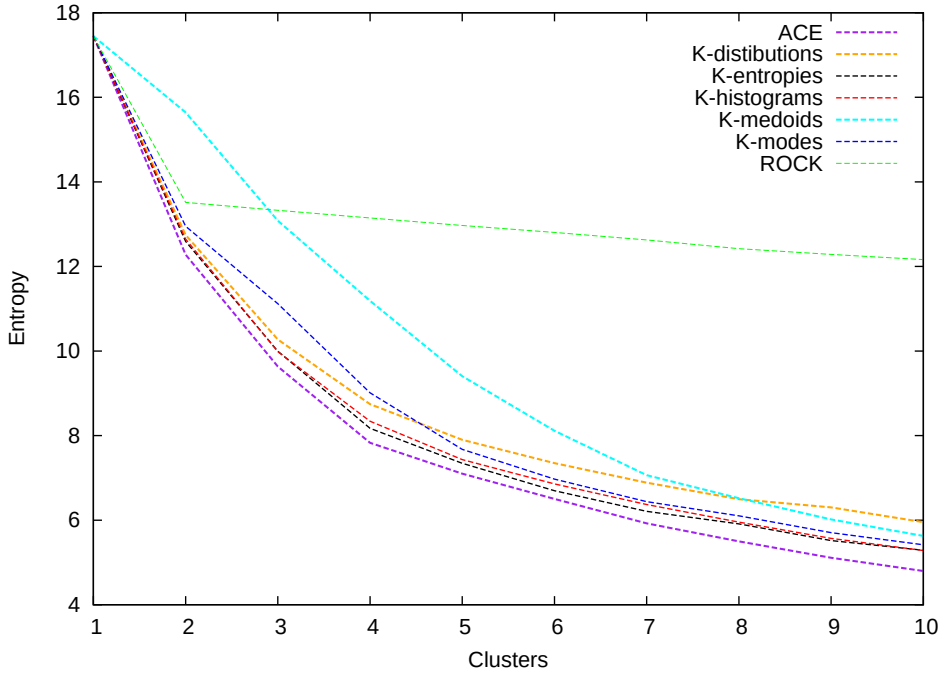
Taulukossa 7 on eri menetelmien antamien tuloksien luokitteluvirheet. Luokitteluvirhe voidaan laskea järkevästi ainoastaan oikalla ryhmien lukumäärällä. Sieniaineistossa on kaksi luokkaa ja soijapapuaineistossa on neljä luokkaa. Tulokset perustuvat sadan ajon keskiarvoon lukuunottamatta sieniaineiston ryhmittelyä ACE ja ROCK menetelmillä. [1)] ACE menetelmän tulos perustuu vain yhteen ajoon ja [2)] ROCK algoritmi ei kykene tuottamaan kahta ryhmää. ROCK

menetelmässä on käytetty parametrina $\theta = 0.8$ sieniaineiston ryhmittelyssä ja $\theta = 0.5$ soijapapuaineiston ryhmittelyssä.

| Ryhmittelymenetelmä | Sieniaineisto | Soijapapuaineisto |
|---|---|---|
| ACE | 0.109798 [1)] | 0.000000 |
| K-distributions | 0.299581 | 0.156170 |
| K-entropies | 0.310426 | 0.076596 |
| K-histograms | 0.266475 | 0.099149 |
| K-medoids | 0.471672 | 0.367234 |
| K-modes | 0.495579 | 0.156809 |
| ROCK | - [2)] | 0.489362 |

Taulukko 7: Ryhmittelymenetelmien tuottamat luokitteluvirheet

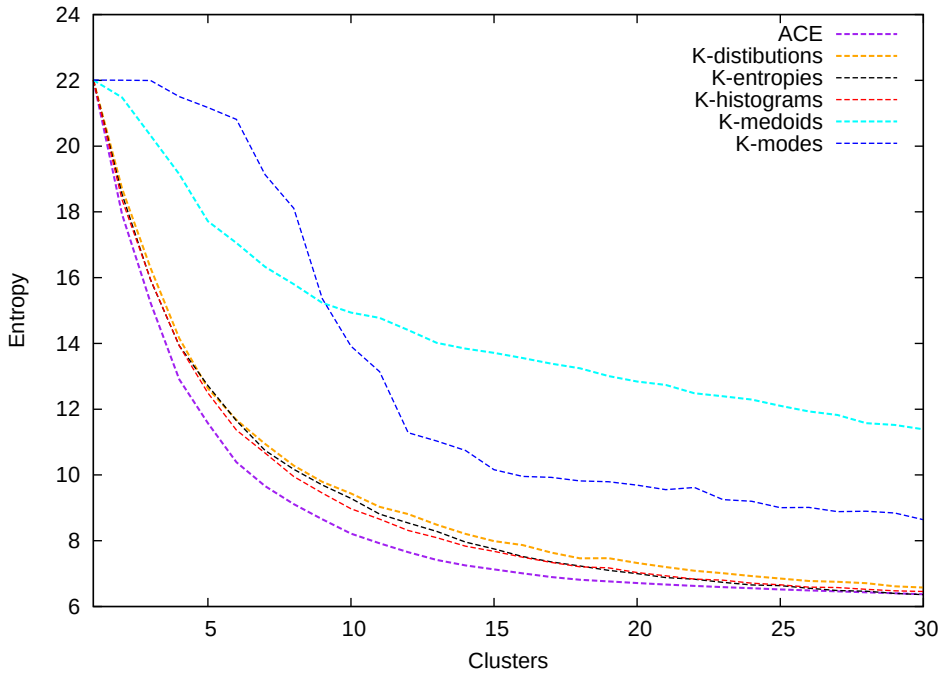
Kuvassa 2 on eri ryhmittelymenetelmien tuottamat entropiat soijapapuaineistolle. Kaikissa menetelmissä on otettu sadan ajon keskiarvo jokaiselle ryhmien lukumäärälle (1-30) eli jokaista menetelmää kohti on suoritettu 3000 ajoa.




Kuva 2: Entropia soijapapuaineistolla

Kuvassa 2 ROCK menetelmän tulos on huomiota herättävä. Kyseessä ei kuitenkaan ole virhe. Usein ROCK menetelmässä ryhmä, joka muodostuu edellisellä kierroksella, on myös mukana seuraavalla kierroksella. Kuvassa kymmenestä kahteen ryhmään, ROCK on todennäköisesti yhdistellyt yksittäisiä vektoreita toiseen kahdesta suuresta ryhmästä. Yhtenä osana ROCK menetelmään kuuluu ryhmien lukumäärän päättäminen ja menetelmä ei toimi hyvin kiinnitetyllä ryhmien lukumäärällä.

Kuvassa 3 on eri ryhmittelymenetelmien tuottamat entropiat sieniaineistolle. Koejärjestely on vastaava kuin soijapapuaineiston tapauksessa lukuunottamatta ACE, jonka tulokset perustuvat yhteen ajoon. ROCK aineistoa on jätetty pois, koska menetelmä tuottaa vähintään 21 ryhmää kuten alkuperäinen ROCK menetelmä [5].




Kuva 3: Entropia sieniaineistolla

Kokonaisuudessaan tulokset ovat melko järkeviä. K-modes ja k-medoids suoriutuvat heikommin, koska niiden ryhmän edustajana on yksittäinen vektori. K-distributions, k-entropies, k-histograms tuottivat jakaumaa ryhmän edustaja käyttäen parempia tuloksia. ROCK tuottaa kohtalaisia tuloksia niiltä osin kuin vertailu on mahdollista. Suurimpia yllätyksiä oli agglomeratiivisen ACE menetelmän tuottamien ryhmittelyjen korkea laatu.

## Viitteet

[1] Cai, Z., Wang, D., and Jiang, L. K-distributions: A new algorithm for clustering categorical data. In *Lecture Notes in Computer Science* (8 2007), vol. 4682, Springer Berlin / Heidelberg, pp. 436–443.

[2] Chen, K., and Liu, L. The "best k" for entropy-based categorical data clustering. In *SSDBM'2005: Proceedings of the 17th international conference on Scientific and statistical database management* (Berkeley, CA, US, 2005), Lawrence Berkeley Laboratory, pp. 253–262.

[3] He, Z., Xu, X., Deng, S., and Dong, B. K-histograms: An efficient clustering algorithm for categorical dataset. *CoRR abs/cs/0509033* (2005).

[4] Huang, Z. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min. Knowl. Discov. 2*, 3 (1998), 283–304.

[5] Rajeev, S. G., Rastogi, R., and Shim, K. Rock: A robust clustering algorithm for categorical attributes. In *Information Systems* (1999), pp. 512–521.

[6] San, O. M., nam Huynh, V., and Nakamori, Y. An alternative extension of the k-means algorithm for clustering categorical data.

[7] Theodoridis, S., and Koutroumbas, K. *Pattern Recognition, Third Edition.* Academic Press, February 2006.

# 6 LIITE 1: Puuttuvien arvojen käsittely

Usein aineistoissa esiintyy puuttuvia attribuuttien arvoja. Nykyinen toteutus ei huomioi puuttuvien arvojen olemassa oloa. Aineistojen puuttuvat arvot on huomioitava esikäsittelyssä. Työtä tehdessä puuttuvien arvojen käsittelyyn tuli joitain ideoita, jotka on esitelty tässä liitteessä. Puuttuvien arvojen käsittelyä ei ole kuitenkaan toteutettu, koska muutokset muuttaisivat olemassa olevia menetelmiä olennaisesti ja tulokset eivät olisi enää vertailukelpoisia. Seuraavaksi esiteltyjä menetelmistä ei ole testattu kattavasti. Liitteessä viitataan puuttuvaan arvoon epsilonilla ($\epsilon$) ja koodissa vakiolla $EPSILON$.

Pääajatuksena on poistaa puuttuvien arvojen vaikutus. Käytännössä ryhmittelymenetelmät voivat olettaa puuttuvan arvon etäisyyden olevan muihin arvoihin nollaksi. Arviointimenetelmissä puolestaan puuttuvia arvoja ei ajatella olevan ollenkaan. Sen lisäksi ettei puuttuvien arvojen aiheuttamaa virhettä ei lasketa kokonaisvirheeseen, on prosenttiosuuksien oikeellisuuden vuoksi myös vähennettävä puuttuvien arvojen osuus kokonaislukumäärästä. Tärkeintä on se, että ryhmittelymenetelmät ja arviointimenetelmät käsittelevät tilanteet yhtenäisellä tavalla.

Kuvassa 4 on ryhmän entropian laskeminen puuttuvat arvot huomioiden.

Lisäksi esimerkiksi K-modes menetelmässä voidaan olettaa, ettei moodiarvo saa olla tuntematon. Kuvassa 5 on esimerkki K-modes moodiarvon valinnasta puuttuvat arvot huomioiden:

Tästä voi aiheutua myös tilanne, jossa kaikki arvot ovat tuntemattomia eli toisin sanoen moodin arvolle ei ole yhtään ehdokasta. Tilanne voidaan kuitenkin käsitellä kuten mikä tahansa muukin tasapeli.

```
double CATClusterEntropy(CATCODEBOOK* pCCB, int i)
  {
  CODEBOOK* pCB = pCCB->pCodebook;
  double error = 0.0, probability;
  int j, k, count;

  for (j = 0; j < VectorSize(pCB); j++)
    {
    for (k = 0; k < Categories(pCCB, j); k++)
      {
      if (CategoryInstances(pCCB, i, j, k) != 0 && k != EPSILON)
        {
        count = Count(pCCB, i) - CategoryInstances(pCCB, i, j, EPSILON);
        probability = (double)CategoryInstances(pCCB, i, j, k) /
                      (double)total;
        error -= probability * log(probability);
        }
       }
     }
  return error * Count(pCCB, i);
  } /* CATClusterEntropy */
```

Kuva 4: Ryhmän entropian laskenta puuttuvat arvot huomioiden

```
void FindMode(CATCODEBOOK *pMCB, int prototype)
  {
  CODEBOOK *pCB = pMCB->pCodebook;
  int i, j, ins;
  int MaxInstances = -1, MaxCategory = -1;

  for (i = 0; i < VectorSize(pCB); i++)
    {
    MaxInstances = -1;

    for (j = 0; j < Categories(pMCB, i); j++)
      {
      ins = CategoryInstances(pMCB, prototype, i, j);

      if (j != EPSILON && ins > MaxInstances)
        {
        MaxCategory = j;
        MaxInstances = ins;
        }
      }

    VectorScalar(pCB, prototype, i) = MaxCategory;
    }
  } /* FindMode */
```

Kuva 5: Moodin valinta puuttuvat arvot huomioiden