Kari Tapani Arkko

Assosiaatioiden ja sekvenssien louhinta suurista datamassoista

Tietotekniikan (TLA)
pro gradu -tutkielma
20. huhtikuuta 2006

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Kari Tapani Arkko
Yhteystiedot: ktarkko@cc.jyu.fi
Työn nimi: Assosiaatioiden ja sekvenssien louhinta suurista datamassoista
Title in English: Mining associations and sequences from large data sets
Työ: Tietotekniikan (TLA) pro gradu -tutkielma
Sivumäärä: 77
Tiivistelmä: Tutkielman tavoite on antaa yleiskuva tietämyksen muodostamisesta sekä tutkia erilaisia datan analysointi- ja visualisointitekniikoita perinteisten OLAP-tekniikoiden rinnalle. Tutkimuskohteena on TietoEnator Oyj:ssä generoitu data, johon on suunniteltu haluttuja säännönmukaisuuksia, riippuvuuksia ja muita ominaisuuksia. Teoreettisessa osuudessa esitellään tietämyksen muodostusprosessi yleisellä tasolla ja käydään tarkemmin läpi tähän prosessiin liittyen klusterointia itseorganisoituvalla kartalla sekä assosiaatiosääntöjen ja sekvenssihahmojen louhintaa. Lisäksi annetaan katsaus aikariippuvaisen datan ja tutkittujen tekniikoiden tulosten visualisoinnista. Empiirisessä osassa toteutetaan tiedonlouhintatyökalun prototyyppi, jossa käytetään osana valmiita tiedonlouhinnan algoritmeja. Tiedonlouhintatyökalun tarkoituksena on löytää datan sisältämät ominaisuudet.
English abstract: The purpose of this thesis is to give an overview of knowledge discovery in databases as well as study various data analysing and visualization techniques alongside with OLAP techniques. Data used in the research is generated in TietoEnator Inc. Data includes regularities, dependencies and other features. Knowledge discovery in databases, self-organizing map, association rules and sequential patterns are reviewed in the theoretical part of the thesis. Also visualization of time series and data mining results are discussed. The data mining application is developed in the empirical part, reusing data mining algorithms. The goal of this application is to find characteristics from the data.
Avainsanat: Tietämyksen muodostaminen tietokannoista, tiedonlouhinta, visualisointi, assosiaatiosäännöt, sekvenssihahmot, SOM, Itseorganisoituva kartta, OLAP.
Keywords: Knowledge discovery in databases, assosiation rules, sequential patterns, visualization, data mining, SOM, Self-Organizing Map, OLAP.

Sisältö

1 Johdanto

Automaattiset datankeräystyökalut ovat tehneet helpoksi datan keräämisen ja samalla kasvattaneet analysoitavien tietokantojen kokoa radikaalisti viime vuosien aikana. Datatulvan seurauksena yritykset ovat pullollaan tutkimuksen ja liiketoiminnan kannalta tärkeitä tietokantoja. Näiden valtavien datamäärien analysointi perinteisin menetelmin on hidasta ja kallista tai muuten epäkäytännöllistä. Kaikkea dataa ei keritä analysoimaan riittävällä tarkkuudella.

Perinteiset tietokanta-analyysimenetelmät perustuvat siihen, että dataa haetaan yksittäisten arvojen perusteella. Hakuja tekevällä täytyy tällöin olla käsitys siitä, minkälaisia arvoja järjestelmä sisältää ja minkälaiset haut ovat tarkoituksenmukaisia. Usein törmätään kuitenkin tilanteeseen, jossa ei ole etukäteen täysin selvää, mitä laajasta datakokonaisuudesta on saatavissa selville. Ratkaisuksi on kehitetty tiedonlouhintamenetelmiä. Näillä menetelmillä saadaan suurista määristä dataa esille sen kiinnostavimmat ominaisuudet puoliautomaattisesti. Motivaationa näiden menetelmien kehitykselle on epäilys siitä, että tietokannoissa saattaa olla hyödyllistä tietoa piilossa analysoimattomien datamassojen alla.

Tutkimuksen lähtökohtana on TietoEnator Oyj:ssä esille tullut tarve uusille tiedon analysointitekniikoille tietokantojen koon kasvaessa. Tutkimuksen kohteena olevasta järjestelmästä kerätään havaintoaineistoa N:llä mittauspisteellä ja jokaisessa mittauspisteessä on $M(n)$ kappaletta sensoreita. Kaikki mittaustulokset ovat kiinnostavia, myös ne, jotka puuttuvat syystä tai toisesta. Kokonaisuudessaan mittaustuloksia yhdistää aika, joten kaikki mittaustulokset (tai paremmin yritykset) ovat aikaleimattuja. Lisäksi järjestelmästä ei saada mittaustuloksia säännöllisesti vaan vaihtelevalla jaksolla. Vaihtelevuus tässä tarkoittaa sitä, että myös eri sensoreiden mittaustulosten määrät voivat vaihdella vapaasti. Mittaustulokset tallentuvat yrityksen operatiivisiin tietokantoihin, joista havaintoaineisto siirretään ja koostetaan tietovarastoon eriasteisiksi yhteenvetotiedoiksi. Muodostetun tietovaraston tietoja on analysoitu OLAP-tekniikoilla ja edelleen visualisoitu käyttäjälle eri tavoilla.

Tutkimuksessa halutaan käyttää TietoEnator Oyj:ssä simuloitua dataa, vaikka tutkimuksen kohteena oleva data on olemassa. Tähän simuloituun dataan on suunniteltu säännönmukaisuuksia, riippuvuuksia ja muita ominaisuuksia, joita tiedetään oikean

havaintoaineiston sisältävän. Hyvänä puolena simuloidussa datassa on se, että tällöin tiedetään tarkalleen, mitä havaintoaineistosta pitäisi löytyä. Näin saadaan valittua parhaat menetelmät näiden ominaisuuksien louhintaan. Lisäksi tietämyksen muodostamisessa ei tarvita kohdealueen asiantuntijaa, koska pidättäydytään simuloidussa datassa. Datasta tunnetaan muuttujat, jotka ovat tarkoituksenmukaisia tiedonlouhinnan kannalta. Huonona puolena on se, että näillä simuloituun aineistoon soveltuvilla menetelmillä ei välttämättä löydetä oikean datan kaikkia ominaisuuksia, mutta oletettavasti niillä kuitenkin saadaan hyvä lähtökohta tiedonlouhinnalle.

Tutkimuksen tavoitteena on tutkia uusia datan analysointi- ja visualisointitekniikoita tavanomaisten OLAP-tekniikoiden rinnalle. Näillä uusilla tekniikoilla pyritään löytämään helposti muutokset ja riippuvuudet analysoitavasta havaintoaineistosta sekä visualisoimaan löydetyt tulokset selkeästi käyttäjälle. Tutkimuksen empiirisessä osuudessa toteutetaan prototyyppi tiedonlouhinnan työkalusta, joka käyttää hyväkseen valmiita tiedonlouhinnan algoritmeja ja ohjelmistoja. Prototyypillä pyritään saamaan kuva, miten valitut tiedonlouhintatekniikat parantavat olemassa olevaa analyysityökalua.

Tutkielma jakautuu seuraaviin osiin: luvussa 2 määritellään työssä käytettäviä termejä. Luvussa 3 kerrotaan mitä tiedonlouhinta tarkoittaa ja mitä tiedonlouhinnalla voidaan löytää sekä miten tiedonlouhinta ja OLAP liittyvät tietämyksen muodostamiseen. Luvussa 4 esitellään itseorganisoituva kartta. Luvussa 5 tutkitaan yleisesti käytössä olevaa tiedonlouhinnan menetelmää, assosiaatioanalyysia sekä assosiaatiosääntöjen laajennuksena sekvenssihahmoja. Luvuissa 6 ja 7 esitellään näiden menetelmien tuloksien visualisointia sekä näiden menetelmien toteutusta tiedonlouhintatyökalun prototyypissä. Luvussa 8 on yhteenveto.

2 Määritelmiä

2.1 Data, informaatio, tietämys ja viisaus

Kuronen [39, Luku 2] määrittelee ”tietämisen portaat” seuraavasti: data, informaatio, tieto, ymmärrys ja viisaus. Kaksi ensiksi mainittua ovat ihmisen tajunnan kannalta pääasiassa ulkoisia, kolme jälkimmäistä kuuluvat yksinomaan mielen ja tajunnan piiriin. Näitä tietämisen eri asteita on pohtittu tarkemmin lähteessä [44].

Koska suomen kielessä tiedon ja informaation käsitteet ovat hyvin vaikea määritellä yksikäsitteisesti, tukeudutaan ”tietämisen portaissa” informaatiotutkimuksen piirissä esiintyvään *tiedon arvoketjuun (value chain of information)*. Ideana on tiedon/informaation jalostuminen ihmisen käyttöön [30, sivu 5]. Tässä työssä tiedon arvoketju määritellään seuraavasti:

$$\text{data} \rightarrow \text{informaatio/tieto} \rightarrow \text{tietämys/ymmärrys} \rightarrow \text{viisaus/osaaminen} \qquad (2.1)$$

Ymmärrystä sen enempää kuin viisauttakaan ei pystytä määrittelemään täsmällisellä tavalla. Tavanomaisin ajattelutapa lienee sellainen, että tietojen käyttäminen ihmisen aktiivisessa toiminnassa johtaa vähitellen laajempien kokonaisuuksien hallintaan eli ymmärrykseen. Ymmärryksestä taas elämänkokemus vähitellen jalostaa ripauksen viisautta, puhutaanhan esimerkiksi iän mukanaan tuomasta viisaudesta. Tämän ajattelutavan mukaan jokainen ihminen pääsisi vähitellen nauttimaan sekä ymmärryksestä että viisaudesta [39, Luku 2].

Määritelmä 2.1.1 *Data (data)* on tässä työssä ymmärretty olevan aineistoa kohdealueesta, joka on koodattu tulkittavaan muotoon. Esimerkiksi sensoreiden keräämää havaintoaineistoa tallennettuna binaariseen muotoon tietokantoihin, toimittajan uutiset merkkijonoina sanomalehdessä tai puheen ääniaallot. Dataan ei välttämättä liity mitään merkitystä eli se ei sisällä informaatiota [39, Luku 2]. Esimerkiksi avaruuden kohina on dataa vailla merkitystä. Data on siis potentiaalista informaatiota, joka voi tulla välityksen kohteeksi [30, sivu 5].

Määritelmä 2.1.2 *Informaatio/tieto (information)* on välitettävänä olevaa dataa, johon on liitetty tai johon on liitettävissä jokin merkitys tai tulkinta [39, Luku 2]. Se on

seurausta sekä informaation tuottajan tavoitteellisesta tietämyksen muokkauksesta että jotakin, jonka havaitseminen vaikuttaa vastaanottajan tietämyksen tilaan ja muokkaa sitä [30, sivu 5].

Informaatiolla tarkoitetaan siis datan ilmaisemaa viestiä. Atk-sanakirjan mukaan informaatio on *"vastaanotetun datan ihmiselle tuottama mielle tai merkitys"* [44, sivu 16]. Data on informaation kantaja. Datan (sensorin mittausaineiston) sisältämää informaatiota voidaan ilmaista esimerkiksi (x,y)-kuvaajalla, jossa x-akselilla on aika ja y-akselilla mitattu data-arvo.

Tieto on klassisen tiedon käsityksen mukaan hyvin perusteltu tosi uskomus. Pieni osa kaikesta informaatiosta on luonteeltaan sellaista, että se on oppimisen ja omaksumisen avulla muunnettavissa tiedoksi [39, Luku 2]. Niiniluodon [44, sivu 27] kriittisen tieteen realismin käsityksen mukaan tiedoksi kutsutaan niitä väitteitä, joille löytyvät parhaat perustelut, siitäkin huolimatta, että niiden totuutta saatetaan epäillä ja olla jopa oikeassa näissä epäilyissä. Tiedon perustelemisen tulisi vedota perusteisiin, jotka ovat kenen tahansa yhteisön jäsenen hyväksyttävissä. Uuden tiedon keksiminen voi olla arvauksiin tai hyvään onneen perustuvaa, mutta niiden perustelussa täytyy pyrkiä objektiivisuuteen.

Tässä työssä tieto/informaatio on uskomus kohdealueesta, jolle pystytään määrittämään perusteluilla totuusarvo. Informaatio kuvaa siis niitä väitteitä, joille on olemassa paras perustelu, huolimatta siitä, että saatamme epäillä niiden totuutta.

Määritelmä 2.1.3 *Tietämys/ymmärrys (knowledge)* on ihmisellä itsellään oleva ymmärrys itsestään ja ympäröivästä maailmasta tietyllä hetkellä [30, sivu 5]. Tämä muodostuu käytettävissä olevien ja hyväksyttyjen tietojen kokonaisuudesta [44, sivu 27].

Tietämys ei ole siis mikään erityinen, lisäehtoja edellyttävä tiedon laji, vaan tietojen kokonaisuuden määrää ja laajuutta ilmaiseva termi. Tietoa ilmaisevilla väitteillä on kuitenkin yleensä yhteyksiä, joiden kautta tietojen kokonaisuudella on leveyttä, järjestystä ja syvyyttä. Näihin tiedon ulottuvuuksiin viitataan usein sanalla tietämys [44, sivu 27].

Määritelmä 2.1.4 *Viisaus (wisdom)* on kyky käyttää tietämystä omassa toiminnassa [30, sivu 5]. Viisaudessa on kysymys kokonaisvaltaisesta ja tasapainoisesta maailmankatsomuksen käsitteestä. Viisauteen kuuluu näkemys asioiden laajoista yhteyksistä ja merkityksistä sekä käsitys tiedon hankintatavoista ja luotettavuuden asteesta.

Niiniluoto [44, sivu 27] painottaa, että kokeneen ammattilaisen tieto on elämäntaitoa, joka on hankittu työllä ja elämällä. Viisauteen kuuluu tiedon ainesten lisäksi

erottamattomasti moraalinen ulottuvuus sekä omakohtaisesti punnittu ja ihmiskunnan kokemukseen nojautuva arvojärjestelmä hyvän elämän päämääristä.

Määritelmien perusteella voidaan tietämisen vaiheeet kuvata seuraavasti: Data sisältää informaatiota kohdealueesta. Kohdealueen asiantuntija pystyy tulkitsemaan, mitä informaatiota data sisältää käyttämällä erilaisia menetelmiä ja tekniikoita. Jos asiantuntija löytää kiinnostavaa informaatiota ja pystyy tälle informaatiolle määrittämään totuusarvon ja liittämään sen kohdealueen taustatietoon on hän saanut tietämystä. Näitä informaation jyväsistä koostuvia tietämyksiä yhdistelemällä ja soveltamalla asiantuntija kasvattaa viisauttaan kohdealueesta.

2.2 Malli ja hahmo

Tiedonlouhinnassa usein käytettyjä termejä ovat *malli* ja *hahmo*.

Määritelmä 2.2.1 *Malli (model)* on globaali tiivistelmä datajoukosta. Esimerkki yksinkertaisesta mallista on regressiomalli Y = aX + b, missä Y ja X ovat satunnaismuuttujia ja a ja b ovat mallin parametreja [45, sivu 12].

Määritelmä 2.2.2 *Hahmo (pattern)* on väite tai sääntö, joka kuvaa rajoitettua osaa datasta [45, sivu 12]. Kattava hahmo on puolestaan datassa usein toistuva hahmo, joka induktiivisesti yleistää eli tiivistää tietokannan lokaalia tai globaalia informaatiosisältöä [60, sivu 16].

3 Tiedonlouhinta

Tiedonlouhinta (data mining, DM) on yksi osa tietämyksen muodostusprosessia (knowledge discovery in databases, KDD). Tietämyksen muodostaminen tarjoaa yleisen kehyksen ja näkökulman myöhemmissä luvuissa esitettäville tiedonlouhinnan menetelmille. Luvuissa 3.1 ja 3.2 esitellään mihin kohtaan tietämyksen muodostusprosessia tiedonlouhinta kuuluu sekä miten OLAP-tekniikka liittyy tietämyksen muodostamiseen. Luvuissa 3.3 ja 3.4 esitellään yleisesti mitä tiedonlouhinta on ja millaisia malleja tiedonlouhinnalla voidaan löytää.

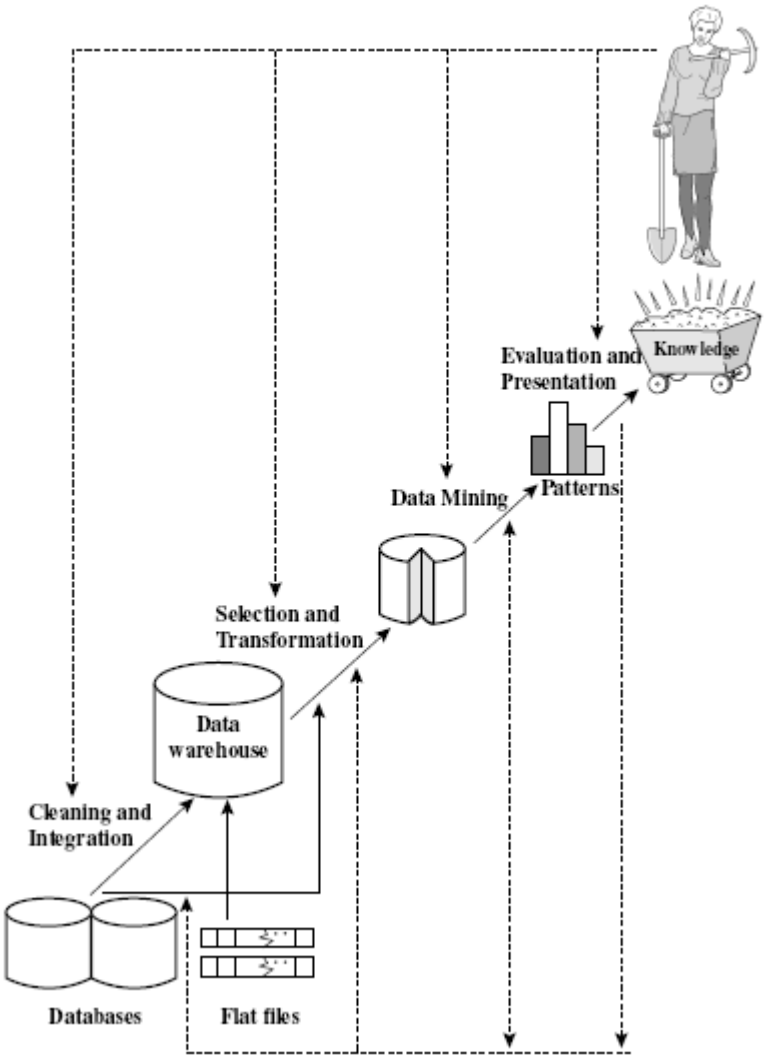
3.1 Tietämyksen muodostaminen

KDD eli tietämyksen muodostaminen tietokannoista tähtää hyödyllisen tiedon löytämiseen suurista tietokannoista. Tietämyksen muodostusprosessi vaatii käyttäjältään paljon eri alojen tietämystä, koska tietämyksen muodostaminen on synteesi koneoppimisen, tilastollisen mallintamisen ja tietokantatekniikoiden menetelmärepertuaareista [60, sivu 16]. Tietämyksen muodostamisen prosessit kuvaavat tiedon löytämisen vaiheita aina tiedon kokoamisesta päätöksentekoon (kuva 3.1).

Han ja Kamber [21, sivu 7] määrittelevät tietämyksen muodostusprosessin koostuvan seitsemästä vaiheesta:

Vaihe 1 *Datan puhdistaminen (data cleaning)*, jossa datasta poistetaan häiriöitä ja ristiriitaisuuksia, koska reaalimaailman data on yleensä puutteellista, häiriöistä ja epäkonsistenttia. Datan puhdistusrutiinit pyrkivät täyttämään tyhjiä arvoja, tasoittamaan häiriötä samalla kun tunnistetaan ulkopuolisia havaintoja ja korjataan datan ristiriitaisuuksia [21, sivut 109–112]. Esimerkiksi jos ilmiön jonkin ominaisuuden mittaaminen ei ole onnistunut niin tällöin data on puutteellista tämän ominaisuuden osalta. Edelleen mittausolosuhteet ovat voineet aiheuttaa dataan häiriöitä, jolloin saman muuttujan arvot vaihtelevat vaikka todellisuudessa ne ovat pysyneet samana. Lisäksi datassa voi esiintyä ristiriitaisuuksia, kuten muuttujan arvo, jonka esiintyminen todellisuudessa ei ole mahdollista.

Vaihe 2 *Datan integrointi (data integration)*, jossa dataa yhdistellään eri dataläh-



Kuva 3.1: Tietämyksen muodostamisen prosessi [21, s. 6]. Jiawei Han on antanut luvan kuvan käyttöön.

teistä, jolloin louhinnan kohteena olevasta datasta saadaan yhdenmukainen. Tietokannoissa voi olla esimerkiksi tarpeettomia attribuutteja, erinimiset attribuutit saattavat tarkoittaa samaa asiaa tai samaa tarkoittavat attribuutit voivat saada eri arvoja eri tietokannoissa [21, sivut 112–114].

Vaihe 3 *Datan valinta (data selection)*, jossa tietokannasta valitaan analyysia varten tarkoituksenmukainen data. Tiedonlouhintaan ei ole järkevää valita kaikkea mahdollista dataa, koska ylimääräinen kohdealueeseen kuulumaton data voi peittää alleen tärkeämpiä tuloksia tai aiheuttaa vääristymiä tuloksiin. Datan valinnassa koko havaintoaineistosta voidaan valita vain kiinnostavat muuttujat tai piirteet. Esimerkiksi klusteroinnilla saadaan esille datan yleinen rakenne. Tästä rakenteesta voidaan valita yhden tai muutaman klusterin sisältämät näytteet tarkempaa analysointia varten tai jokaisesta klusterista voidaan valita yksi näyte edustamaan kyseistä klusteria jatkoanalyysissa. Lisäksi näytteistä voidaan valita vain "täydelliset" näytteet. Täydellinen näyte on sellainen, johon ei ole tarvinnut tehdä merkittäviä datan puhdistusoperaatioita.

Vaihe 4 *Datan muuntaminen (data transformation)*, jossa data muunnetaan louhinnalle edulliseen muotoon. Esimerkiksi datasta voidaan poistaa häiriöitä tasoittamalla dataa, summaamalla ja yleistämällä dataa eri merkitystasoille, normalisoimalla dataa sekä rakentamalla uusia attribuutteja annetuista attribuuteista. Normalisointi on hyödyllinen lajittelualgoritmeissa, joissa käytetään neuroverkkoja tai etäisyysmittaan perustuvissa menetelmissä, kuten klusteroinnissa [21, sivut 114–115].

Vaihe 5 *Tiedonlouhinta (data mining)*, jossa valitusta datajoukosta etsitään hahmoja. Erilaisia louhinnan kohteena olevia malleja ja hahmoja on esitelty luvussa 3.4. Voidaan sanoa, että tiedonlouhinnalla pyritään tiivistämään datan ominaisuuksia.

Vaihe 6 *Hahmojen arviointi (pattern evaluation)*, jossa saaduista hahmoista etsitään todelliset ja käyttäjää kiinnostavat hahmot. Assosiaatiosääntöjen kiinnostavuusmetriikoita on esitelty luvussa 5.6.

Vaihe 7 *Tietämyksen esittäminen (knowledge presentation)*, jossa tiedonlouhinnasta ja arvioinnista saadut tulokset visualisoidaan selkeästi käyttäjälle. Perustuen kappaleessa 2.1 määriteltyihin termeihin tämä vaihe on *datan esittämistä (data presentation)*, koska tietämys muodostuu vasta tämän visualisoinnin seurauksena. Itseorganisoituvaan karttaan ja assosiaatiosääntöihin liittyviä sekä muita visualisointitekniikoita on esitelty luvussa 6.

Peruslähtökohdat tietämyksen muodostamiselle ovat kohdealueen ymmärtäminen ja datan olemassaolo. Tietämyksen muodostamisprosessin eri vaiheissa on hyvä olla ymmärrys kohdealueesta. Esimerkiksi vaiheessa 1 on hyvä olla ymmärrys siitä, mitkä ominaisuudet datassa ovat häiriöitä tai ristiriitaisuuksia sekä mitkä datat liittyvät toisiinsa. Lisäksi mitkä osat datasta ovat tarkoituksenmukaisia tietämyksen muodostamisprosessin vaiheissa 2 ja 3 sekä kuinka dataa saa muuntaa vaiheessa 4 ettei siitä häviäisi olennaista informaatiota. Esimerkiksi tästä syystä tietämyksen muodostamisprosessi on pakosti iteratiivinen ja interaktiivinen. Tiedonlouhinnan tulokset voivat osoittaa, että olisi syytä tehdä muutoksia tietämyksen muodostusprosessin aikaisemmissa vaiheissa [43, s. 4–5]. Esimerkiksi datan valintavaiheessa pitäisi lisätä tai poistaa analysoitavia muuttujia, jotta saataisiin parempi ymmärrys kohdealueesta tai tiedonlouhinnan menetelmiä pitäisi vaihtaa, koska kaikki menetelmät eivät sovellu kaikkien ominaisuuksien louhintaan. Iteratiivisuus johtuu myös siitä, että yleensä tiedonlouhinnan ja kohdealueen asiantuntija ovat eri henkilöitä, jolloin tietämys kohdealueesta kasvaa analyysin myötä ja uusia näkökulmia ilmaantuu. Yleensä datan määrä ja saatavuus ei ole ongelma, koska yritykset ovat tallentaneet jo pitkään tietoja, jotka ovat tärkeitä yrityksen toiminnan kannalta tai sisältävät luultavasti arvokasta tietoa yrityksen kohdealueesta.

3.2 Tietovarastointi ja OLAP

Tietokannat ovat yleinen yritysten tiedonkeräyspaikka. Tietokanta on määritelty olevan: *"yhteinen kokoelma loogisesti yhteenkuuluvaa dataa ja kuvaus tästä datasta sekä suunniteltu kattamaan organisaation informaatiotarpeet"* [17, sivu 14]. Kuitenkaan yritysten operatiiviset tietokannat eivät usein täytä käyttäjien raportointi- ja analysointitarpeita riittävän kattavasti. Operatiivisten tietokantojen päätarkoituksena on suorassa käytössä syötettyjen tapahtumien tehokas käsittely. Operatiiviset tietokannat ovat tyypillisesti suunniteltu useamman käyttäjän yhtäaikaiskäyttöön ja niissä haun kohteena on yleensä yksi tai muutama tietoalkio [28, sivu 2].

Tietovarastoinnin (data warehousing) ideana on kerätä yrityksen operatiivisissa tietokannoissa oleva data yhteen, fyysisesti erillään olevaan tietovarastoon [21, sivu 40] niin, että joustava raportointi ja analysointi on mahdollista [37, sivu 1]. *"Tietovarasto on kohdeorientoitunut, integroitu, aika-variantti ja pysyvä kokoelma dataa, joka tukee johdon päätöksentekoprosessia"* [17, sivu 1047]. Tämä tarkoittaa, että tietovarasto on organisoitu pääaiheiden ympärille, kuten asiakas, toimittaja, tuote ja myynti. Tieto-

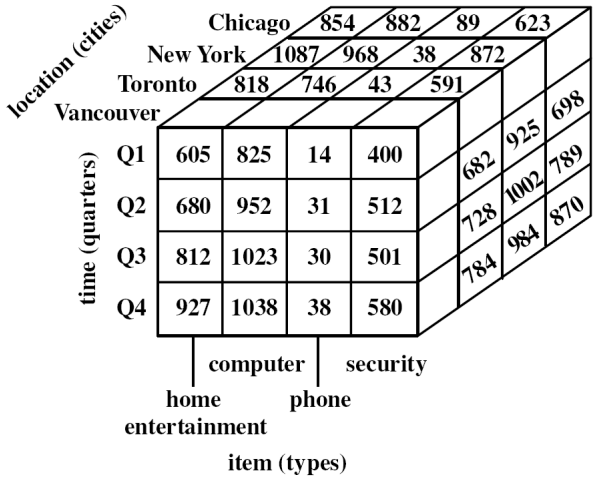
varasto ei keskity tapahtumien prosessointiin vaan datan mallinnukseen ja analyysiin päätöksentekijöiden avuksi. Näin ollen tietovarasto antaa yksinkertaisen ja tiiviin näkymän tietystä alueesta ilman kohteita, jotka eivät ole tarpeellisia päätöksenteossa [21, sivu 40]. Operatiivisista järjestelmistä data siirretään tietovarastoon, jossa ne on organisoitu rakenteeltaan moniulotteiseksi. Siirtoprosessi vastaa siitä, että siirretyt datat ovat oikeita, tarkkoja, ajankohtaisia ja luotettavia [34, sivu 4]. Yleensä tähän datojen siirtovaiheeseen kuuluvat relevantin datan *erottaminen (extraction)*, ylimääräisen ja toisteellisen datan *siivous (cleansing)*, esitysmuotojen *muunnokset (transformation)* sekä datojen *koostaminen (summarization, aggregation)* eri tasoille [37, sivu 1]. Esimerkiksi myynti voi olla valmiiksi summattuna päivän, viikon tai kuukauden jaksoissa. Siirtovaihe datalähteistä tietovarastoon sisältää siis tietämyksen muodostamisprosessin vaiheet 1 ja 2. Tietovarastoon voidaan siirtää dataa myös muista tietolähteistä, kuten säätilasta, pörssikursseista jne. Tietovarastossa oleva data eroaa operationaalisessa tietokannassa olevasta datasta pääasiassa siinä, että tietovaraston data on valmiiksi koostettua [37, sivu 2]. Tietovarastossa datat ovat myös usein pidemmältä aikajaksolta kuin operatiivisissa järjestelmissä säilytettävät datat, joten tietovarasto sisältää historiallisen informaation. Tietovarastossa data voi käsittää 10 vuoden aikajakson, kun operatiivisessa tietokannassa jakso voi olla vain murto-osan tästä. Tämä antaa mahdollisuuden tarkastella datoissa tapahtuneita muutoksia pitkällä aikajänteellä [34, sivu 5].

Tietovarastointia ja siihen liittyvää raportointia kutsutaan OLAP-tekniikaksi (online analytical processing). OLAP ei ole mikään yksittäinen ohjelmisto tai menetelmä, vaan se kattaa suuren joukon menetelmiä datan talletuksesta raportointiin ja visualisointiin [37, sivu 1]. OLAP-järjestelmässä on yleensä yhtäaikaisia käyttäjiä vain muutamia ja haun kohteena on jalostettua yhteenvetodataa [28, sivu 2]. Tämä takaa sen, että haut ovat nopeampia kuin operatiivisista tietokannoista.

Tietovaraston sisältöä mallinnetaan usein *datakuutioina (data cube)* (kuva 3.2) [17, sivut 1105–1106]. Datakuutio on moniulotteinen analyysimalli, joka koostuu dimensioista ja faktoista. Dimensiot muodostavat moniulotteisen ristiintaulukoinnin. Fakta tarkoittaa dimensioiden leikkauspisteessä olevaa dataa, joka lasketaan halutulla *koostefunktiolla (aggregate function)*. Datakuutioon voidaan kohdistaa useita erilaisia operaatioita. Kuutiota voidaan kääntää, jolloin datan tarkastelukulma vaihtuu. Lisäksi kuutiosta voidaan valita haluttuja osia *(slicing, dicing)* tarkastelun kohteeksi. Kuutio voi sisältää jokaisesta dimensiosta useita hierarkisia tasoja, joita voidaan käsitellä *porautumalla (drill down)* tai *yleistämällä (roll up)*. Porautumisessa tarkastellaan sa-

	location="Chicago"				location="New York"				location="Toronto"				location="Vancouver"			
time	home ent.	comp	phone	sec.	home ent.	comp	phone	sec.	home ent.	comp	phone	sec.	home ent.	comp	phone	sec.
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	874	927	1038	38	580

Taulukko 3.1: Kolmiulotteinen taulukko myyntidatasta. Taulukon dimensioina on aika, tuote (kodin viihde, tietokone, puhelin, turvallisuus) ja sijainti. Myyntiluvut ovat tuhansia dollareita [23, sivu 8]. Jiawei Han on antanut luvan taulukon käyttöön.



Kuva 3.2: Kolmiulotteinen datakuutio, joka esittään taulukon 3.1 dataa [23, sivu 8]. Jiawei Han on antanut luvan kuvan käyttöön.

maa dataa suuremmalla tarkkuudella. Esimerkiksi, jos tehdään analyysi tunneittain, voidaan porautumalla saada vastaava analyysi minuuteittain. Porautumisen maksimitarkkuutta rajoittaa se, millä tarkkuudella datat on talletettu tietovarastoon [37, sivu 2].

Tietovarastot ja OLAP-työkalut antavat hyvän pohjan interaktiiviseen analyysiin moniulotteisesta datasta erilaisine rakeisuuksineen. OLAP-tekniikan päälle voidaan integroida muita tiedonlouhinnan toiminnallisuuksia, kuten luokittelu, ennustaminen, assosiaatioanalyysi ja klusterointi [21, sivu 40].

3.3 Mitä tiedonlouhinta on?

Tiedonlouhinta nähdään osana tietämyksen muodostamisprosessia (kuva 3.1), jonka kohteena voivat olla mitkä tahansa datajoukot, eivät pelkästään tietokannat. Tiedonlouhinta on määritelty olevan *"suurten, tiettyä tarkoitusta varten kerättyjen tietojoukkojen analyysia, jonka tavoitteena on löytää odottamattomia suhteita ja tiivistää dataa uusilla tavoilla, jotka ovat sekä ymmärrettäviä että käyttökelpoisia"* [45, sivu 9]. Kirjoittajan mielestä määritelmään pitäisi lisätä, että suhteiden ei tarvitse välttämättä olla odottamattomia. Arvokasta on myös saada varmistus tai pohja uskomukselle. Tässä työssä tiedonlouhintaa käsitellään tietämyksen muodostamisen osana, vaikka näitä käytetään usein kirjallisuudessa synonyymeina. Tietämyksen muodostamisprosessi painottaa erityisesti tietokannoissa olevan datan analysointia, mutta tiedonlouhintaa voidaan tarkastella myös yleisemmin ottamatta kantaa syötedatan formaattiin [45, sivu 9].

Tiedonlouhinta on tieteidenvälistä toimintaa, jonka piiriin kuuluu joukko erilaisia menetelmiä ja algoritmeja. Tiedonlouhinnan lähitieteitä ovat tilastotiede, tietokannat, koneoppiminen, hahmontunnistus, tekoäly ja visualisointi [45, sivu 9]. Tiedonlouhintaa koskevan tutkimuksen tavoitteena on kehittää menetelmiä ja prosesseja, joilla datajoukon säännönmukaisuudet saadaan kuvattua laskennallisesti tehokkaasti, tiiviissä ja helposti tajuttavassa muodossa [60, sivu 15]. Kiinnostuksen kohteena voivat olla esimerkiksi globaali kuvaus kohdealueesta, kuten kokonaistodennäköisyysjakauma, havaintojen ryhmittely samankaltaisuuden perusteella omiin ryppäisiinsä, visuaaliset yhteenvedot ja tunnusluvut tai lokaalit ominaisuudet, kuten usein toistuvat hahmot tai muusta aineistosta erityisen poikkeavat hahmot, tilastolliset riippuvuudet sekä aikasarja-aineistojen hahmot (trendit, syklit, poikkeamat, usein toistuvat tapahtumasarjahahmot, jne.) [59]. Datajoukon mielenkiintoisia säännönmukaisuuksia etsitään

muodostamalla hierarkioita, soveltamalla ryvästämis- eli klusterointimenetelmiä ja luokittelumenetelmiä sekä yleisemmin etsimällä kattavia tai toistuvia hahmoja (frequent pattern) [60, sivu 15]. Tiedonlouhinnan menetelmät voidaan jakaa karkeasti klusterointiin, luokitteluun ja assosiaatiosääntöjen etsimiseen [45, sivu 9].

3.4 Millaisia malleja voidaan louhia?

Tiedonlouhintamenetelmät voidaan jakaa ryhmiin niiden ominaisuuksien ja tekniikoiden mukaan. Kuitenkin suurin osa näistä menetelmistä on sovellettavissa yhteen kohdealueeseen, kuten rikosten louhimiseen [15]. Tällöin saadaan erilaisia näkökulmia kohteena olevasta ilmiöstä. Tavallisesti tiedonlouhinta jaetaan kahteen kategoriaan: selittävä/kuvaileva ja ennustava. Selittävä louhinta pyrkii esittämään datan yleiset ominaisuudet ja ennustava louhinta antamaan pohjan ennustamiselle [21, sivu 21]. Aliluvuissa kuvataan yleisellä tasolla millaisia malleja tai hahmoja datasta voidaan löytää.

3.4.1 Erotteluanalyysi

Yksinkertaisin tapa tiedonlouhinnasta on erotteluanalyysi, jossa data jaetaan luokkiin tai käsitteisiin, joten erotteluanalyysi voidaan katsoa osaksi luokittelua. Jakaminen ja käsitteiden muodostaminen tehdään jo olemassa olevan tietämyksen mukaan [21, sivut 21–22]. Erotteluanalyysissa etsitään erottelufunktioita, joilla havainnot voidaan jakaa toisensa poissulkeviin luokkiin joidenkin muuttujien perusteella. Löydettyjä erottelufunktiota voidaan käyttää uusien havaintojen luokitteluun tai luokittelu voidaan tehdä parhaiten erottelevien muuttujien mukaan [4, sivu 14]. Kohdeluokasta voidaan tehdä yhteenveto (datan karakterisointi (characterization)) sen sisältämistä ominaisuuksista. Kohdeluokan yleisiä ominaisuuksia voidaan myös vertailla yhden tai useamman luokan yleisiin ominaisuuksiin (datan diskriminointi (discrimination)) [21, sivut 179–224]. Esimerkiksi OLAP-tekniikoilla tehty erottelu kaksiulotteisessa taulukossa 3.2, jossa havaintoaineisto on jaettu viiden vuoden väleihin ja esitetään havainnot summattuna mittauspisteittäin.

3.4.2 Assosiaatioanalyysi

Assosiaatioanalyysillä tarkoitetaan assosiaatiosääntöjen etsimistä annetusta datajoukosta [21, sivu 22]. Assosiaatiosäännöt kertovat, mitkä tapahtumat ovat todennäköisiä esiintymään yhdessä tai jossain tietyssä aikajaksossa *"Jos tapahtuma A esiintyy, niin*

Vuosi	1991-1995	1996-2000	2001-2005
mp1	1002	1244	2376
mp2	233104	246810	301728
mp3	704	806	10233
mp4	8712	9110	476
mp5	1111	516	2479

Taulukko 3.2: Esimerkki erotteluanalyysistä

tapahtuma B esiintyy todennäköisyydellä $P(A \Rightarrow B)$". Aikajaksoisessa tapauksessa tutkimuksen kohteena voisi olla ostoskuittien yhteenlasketut ostokset kolmen päivän ajalta kun taas ensimäisessä ainoastaan yhden ostoskuitin sisältämät ostokset. Ajan ollessa mukana näiden avulla voidaan ymmärtää kuinka erilliset tapahtumat käyttäytyvät toisiinsa nähden [4, sivu 35]. Esimerkiksi tutkittaessa sinapin assosiaatioita voidaan havaita, että kesällä sinappi assosioi voimakkaasti grillimakkaran kanssa kun taas talvella joulukinkun kanssa.

Yksinkertaisin esimerkki hahmojen esityskielestä löytyy ostoskorianalyysistä, johon assosiaatiosääntöjä ensimmäisen kerran sovellettiin [2]. Tällöin hahmo voisi olla esimerkiksi sääntö: *"Jos asiakas ostaa vaippoja, sinappia ja makkaraa, ostaa hän myös todennäköisesti olutta"*.

Assosiaatiosäännöt kertovat usein sen mikä jo tiedetään, mutta myös uusia yllättäviä suhteita saattaa löytyä. Assosiaatiosääntöjen vahvuutena on se, että ne antavat selvät ja ymmärrettävät tulokset. Lisäksi niiden takana oleva laskenta on ymmärrettävää. Huonona puolena on se, että muuttujia pystyy olemaan vain rajallinen määrä ja oikeaa muuttujien määrää voi olla hankalaa päättää. Muuttujien rajallinen määrä johtuu siitä, että muuttujien määrän kasvaessa assosiaatiosääntöjen määrä kasvaa eksponentiaalisesti. Analyysi saattaa antaa myös sääntöjä, joille ei löydy järkevää selitystä [21, sivu 259]. Assosiaatiosääntöjä on esitelty tarkemmin luvussa 5.

3.4.3 Luokittelu

Luokittelu on prosessi, jolla etsitään joukko malleja (tai funktioita), jotka kuvaavat ja erottelevat datan luokat tai käsitteet. Tavoitteena on käyttää näitä malleja ennustettaessa luokkia, joiden merkitystä ei tunneta. Malli on peräisin opetusdatan analyysistä, jonka merkitykset tunnetaan. Malli voidaan esittää useissa muodoissa (jos–niin)-muodossa, päätöspuulla, matemaattisilla kaavoilla tai neuroverkolla. Luokittelussa ana-

lysoidaan tunnettuja dataluokkia [21, sivu 24]. Luokittelu ja regressio ovat ennustavia menetelmiä. Luokittelussa data-alkiolle pyritään määrittämään jokin ennalta määrätyistä luokista. Luokittelumallia rakennettaessa pyritään aina minimoimaan luokitteluvirhe. Luokittelun yleistyksenä voidaan pitää regressiota, jossa data-alkioille määritellään numeerinen arvo luokan sijaan. Toisaalta myös mallipohjaisen klusteroinnin tulosta voidaan käyttää luokittelumallina [45, sivu 12].

3.4.4 Klusterointi

Klusterointi on kuvaileva menetelmä, jossa data-alkiot ryhmitellään äärelliseen määrään klustereita niiden keskinäisen samanlaisuuden perusteella [45, sivu 12]. Ryhmät muodostetaan siten, että objektit ovat hyvin samanlaisia ryhmän sisällä ja erilaisia toisiin ryhmiin verrattuna [4, sivu 30]. Samanlaisuus määritellään yleensä etäisyysfunktiolla [45, sivu 12]. Klusterointia käytetään yleensä tiedonlouhintaprosessin ensimmäisissä vaiheissa.

Klusterointialgoritmeilla (clustering algorithms) etsitään tietokannasta sopivia ryhmittelyjä. Klusterointia kutsutaan niin sanotuksi ohjaamattomaksi oppimiseksi (unsupervised learning), koska käyttäjä ei itse määrittele tiedonlouhinnassa käytettäviä luokkia vaan klusterointialgoritmi löytää ne itse [60, sivu 16]. Näin ollen klusterointia käytetään kun ei tiedetä datan jaottelua [4, sivu 30]. Yksi suurta suosiota saanut klusterointityökalu on Teuvo Kohosen kehittämään itseorganisoituvaan karttaan [38] pohjautuva SOM ToolBox [5], jossa on mukana hyvät visualisointityökalut. Tarkemmin itseorganisoituvaa karttaa on esitelty luvussa 4.

3.4.5 Poikkeavat havainnot

Tietokanta saattaa sisältää myös poikkeavia dataobjekteja (outliers), jotka eivät kuulu tavalliseen datan käyttäytymiseen tai malliin. Tiedonlouhinta-algoritmien kannalta nämä havainnot ovat häiriöitä tai poikkeuksia, joiden vaikutus pyritään minimoimaan tai ne voidaan eliminoida kokonaan. Riskeinä tässä on se, että tällöin saatetaan menettää arvokasta tietoa poikkeavista arvoista. Tätä tekniikkaa käytetään mm. vakuutuspetosten löytämiseen, asiakaskohtaiseen markkinointiin ja lääketieteellisiin analyyseihin. Tähän voidaan soveltaa esimerkiksi tilastollisia tai etäisyysmittoihin perustuvia menetelmiä [21, sivut 381–389].

3.4.6 Evoluutioanalyysi

Datan evoluutioanalyysi kuvailee ja mallintaa säännöllisyyksiä tai suuntauksia objekteista, joiden käyttäytyminen muuttuu ajan myötä. Säännöllisyyksien ja suuntauksien etsiminen saattaa sisältää edellä mainittuja tekniikoita sovellettuna aikariippuvaiseen dataan [21, sivu 26]. Esimerkiksi voidaan etsiä pitkän aikavälin liikkeitä ja suuntauksia, kuten jaksollisia liikkeitä tai muutoksia, jotka kuvaavat pitkän aikavälin heilahteluja. Heilahtelut eivät välttämättä tapahdu tarkasti samoilla aikaväleillä. Lisäksi voidaan etsiä kausittaisia liikkeitä tai muutoksia, kuten myynnin kasvu joulun aikana sekä epäsäännöllisiä tai satunnaisia liikkeitä, jotka kuvaavat yksittäisiä muutoksia aikasarjadatassa [21, sivu 419].

4 Itseorganisoituva kartta

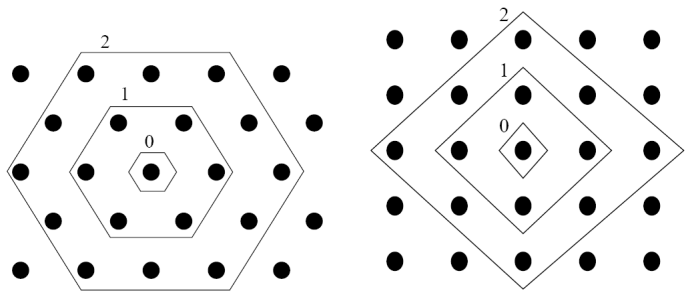
Itseorganisoituva kartta (Self-Organizing Map, SOM) on yksi suosituimmista neuroverkkomalleista. Algoritmi perustuu ohjaamattomaan oppimiseen, jossa opetus tapahtuu täysin dataohjatusti. Ohjatun oppimisen algoritmit vaativat, kuten (Multi-Layered Perceptron, MLP), että datavektoreiden kohdearvot ovat tiedossa. SOM -algoritmilla tätä rajoitusta ei ole. Itseorganisoituva kartta esiteltiin vuonna 1981 moniulotteisen datan epälineaaristen suhteiden visualisointiin, jonka jälkeen sitä on sovellettu lukuisiin eri tehtäviin [61, sivu 1] [38, sivut 191–213] [35], mutta pääasiassa sitä on käytetty datan visualisointiin ja klusterointiin [62, sivu 11].

Itseorganisoituva kartta on neuraalilaskentamenetelmä, jolla voidaan kuvata alkuperäinen syötedata-avaruus $\mathbb{R}^n$, alempiulotteiseen avaruuteen niin, että alunperin lähellä toisiaan olevat näytteet ovat lähellä toisiaan myös alempiulotteisessa kuvauksessa eli kartalla [38].

Luvussa 4.1 kuvataan itseorganisoituvan kartan rakennetta. Luvuissa 4.2 ja 4.3 esitellään kartan alustusta ja opetusta. Luvussa 4.4 tutkitaan, mistä voidaan päätellä onko tuloksena saatu kartta hyvä. Kartan visualisointitekniikoita on kuvattu luvussa 6.3.

4.1 Kartan rakenne

Itseorganisoituva kartta koostuu joukosta karttayksiköitä eli neuroneja, joita on C kappaletta. Neuronit on sijoiteltu säännöllisesti matalaulotteiseen ristikkoon eli karttaan, yleensä 1- tai 2-ulotteiseen. Useampiulotteiset kartat ovat mahdollisia, mutta niitä ei ole yleensä käytetty, koska niiden visualisointi on hankalaa. Jokainen neuroni $\boldsymbol{n}_i$ on d-ulotteinen prototyyppivektori $\boldsymbol{m}_i = [m_{i1}, \ldots, m_{id}]^T \in \mathbb{R}^d$, missä d on syötevektorien dimensio. Kartta sopeutuu dataan päivittämällä prototyyppivektoreitaan syötevektoreiden avulla [62, sivu 11]. Neuroneista $\boldsymbol{n}_i$ muodostuu $d_{\mathcal{L}}$-ulotteinen hila $\mathcal{L}$, missä jokaiseen neuroniin liittyy prototyyppivektori $\boldsymbol{m}_i$ ja neuronin hilanaapurit. Syötedatavektorin $\boldsymbol{x}$ ja prototyyppivektorien $\boldsymbol{m}_i$ välille määritellään jokin etäisyys- tai yleisemmin



(a) Hexagonal grid (b) Rectangular grid

Kuva 4.1: Erikokoisia diskreettejä naapuruuksia: (a) kuusikulmainen ristikko, (b) suorakulmainen ristikko. Sisällä olevat monikulmiot kuvaavat eri kokoisia naapuruuksia. Juha Vesanto on antanut luvan kuvan käyttöön.

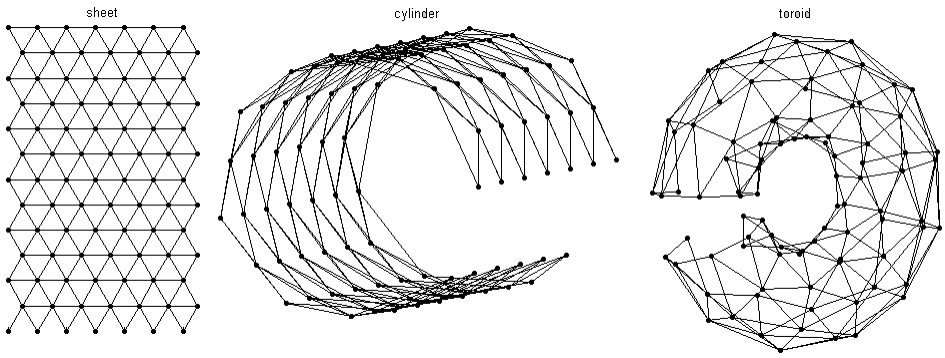
erilaisuusmitta, esimerkiksi euklidinen etäisyys [38, sivu 78]

$$\xi_{xm_i} = \|\boldsymbol{x} - \boldsymbol{m}_i\| = \sqrt{\sum_{j=1}^{n}(x_j - m_{i_j})^2}. \tag{4.1}$$

Ristikon tyyppi (kuva 4.1) (= karttayksikön hilanaapuruston tyyppi) voidaan määritellä suorakulmaiseksi, kuusikulmaiseksi tai jopa epäsäännölliseksi. Näistä kuusikulmainen on tehokas visuaalisessa esityksessä [38, sivu 78]. Yleisesti valitaan naapurustoksi kuusikulmio- tai neliönaapurusto, kartan muodoksi suorakulmio ja etäisyysmitoiksi euklidinen etäisyys. Muita käytettyjä kartan muotoja ovat esimerkiksi sylinteri ja toroidi (kuva 4.2), jolloin kuvauksen reunat saadaan poistettua. Kartan hilassa tulisi käyttää kuusikulmionaapurustoa, jolloin kartan visualisointi ei suosi pysty- ja vaakasuoria suuntia siinä määrin kuin neliönaapurusto [38, sivu 112].

4.2 Alustus

Itseorganisoituva kartta alustetaan ja opetetaan jollakin aineistolla eli opetusjoukolla $\mathcal{D}$. Opetusjoukko koostuu datavektoreista $\boldsymbol{x}_i$, missä $i = 1, \ldots, N$ ja datavektorien dimensio on $d_{\mathcal{D}}$. Ennen varsinaista opetusta karttayksiköt alustetaan eli prototyyppivektoreille annetaan alkuarvot $\boldsymbol{m}_i(0)$. Yksinkertaisia alustusmenetelmiä ovat satunnainen alustus tai alustus satunnaisesti valittuihin opetusjoukon syötevektoreihin. On todettu, että opetus näinkin karkean alustuksen jäljiltä johtaa usein järkevään lopputu-



Kuva 4.2: Kartan muoto vasemmalla on suorakulmainen, keskellä on sylinterin muotoinen oikealla toroidin muotoinen. Hilan muoto on kaikissa tapauksissa kuusikulmionaapurustohila [26].

lokseen, koska alunperin epäjärjestyksessä olevat vektorit järjestäytyvät ajan mittaan. Lineaarisessa alustuksessa vektorit sijoitetaan tasaisesti pääkomponenttitasoon, jonka viirittävät näytejoukon kovarianssimatriisin $d_{\mathcal{L}}$ dominanttia ominaisvektoria. Vektorit asetellaan siten, että prototyyppivektorijoukon painopiste ja datajoukon painopiste yhtyvät. Kartan dimensiot ("sivujen pituudet") ovat puolestaan suhteessa ominaisarvojen suuruuteen. Alustuksen laatu vaikuttaa kartan järjestäytymisen nopeuteen opetuksen aikana. Lineaarinen alustus asettaa karttayksiköt tässä mielessä hyviin lähtöasemiin ja on alustusmenetelmistä suositeltavin [38, sivu 106–107] [26].

4.3 Opetus

Opetuksen aikana itseorganisoituva kartta käyttäytyy kuin joustava verkko, joka taipuu opetusdatan mukaan. Johtuen naapurustosuhteista, naapuriprototyyppejä päivitetään samaan suuntaan ja siten naapurineuronit saavat samankaltaiset prototyyppivektorit [62, sivu 12].

Luvussa 4.3.1 käsitellään yleisiä asioita kartan opetuksessa. Luvuissa 4.3.2 ja 4.3.3 esitellään kaksi itseorganisoituvan kartan opetuksessa käytettyä algoritmia, perusalgoritmi ja eräalgoritmi.

4.3.1 Yleistä

Opetuksen tavoitteena on saada kartta järjestymään ja kvantisointivirhe mahdollisimman pieneksi. Nämä voivat olla ristiriitaisia tavoitteita, sillä useimmiten kartan dimensio on pienempi kuin datan. Asettamalla karttahilan dimensio samaksi kuin opetusdatan luonnollinen dimensio saavutettaisiin hyvä oppimistulos [26].

Kartan järjestymiseen voidaan vaikuttaa opetusparametrien oikealla valinnalla. Yleinen käytännön menetelmä on opettaa kartta ensin laajalla naapurustolla, jolloin naapurustofunktion h parametri σ on suuri suhteessa kartan tilavuuteen. Kohonen esittää tähän ensimmäiseen, karkean opetuksen vaiheeseen, naapuruston sädettä, joka voi olla aluksi yli puolet kartan halkaisijasta pienentyen yhteen yksikköön ja oppimisnopeuskertoimen alkuarvoksi $\alpha \approx 1$, esimerkiksi $\alpha(t) = 0.9(1 - t/1000)$. Tällöin kartta on jäykkä ja saadaan järjestymään globaalisti. Seuraavassa pidemmässä vaiheessa karttaa hienosäädetään ja opetus aloitetaan pienemmällä naapurustolla sekä pienemmällä opetusparametrin alkuarvolla (luokkaa 0.02). Tällöin kvantisointivirhe pienenee, mutta kartta ei toivottavasti enää pääse globaalisti epäjärjestymään [38, sivu 80] [26].

4.3.2 Perusalgoritmi

SOM -algoritmi on iteratiivinen. Jokaisella opetusaskeleella t valitaan satunnaisesti datavektori $\boldsymbol{x_i}$ opetusaineistosta. Seuraavaksi lasketaan etäisyydet kaikkien prototyyppivektoreiden ja datavektoreiden $\boldsymbol{x_i}$ välillä. Parasta, datavektoria $\boldsymbol{x_i}$, vastaavaa yksikköä (best-matching unit, BMU) merkitään

$$c = \|\boldsymbol{x}_i - \boldsymbol{m}_c\| = \text{argmin}_j\{\|\boldsymbol{x}_i - \boldsymbol{m}_j(t)\|\}, \tag{4.2}$$

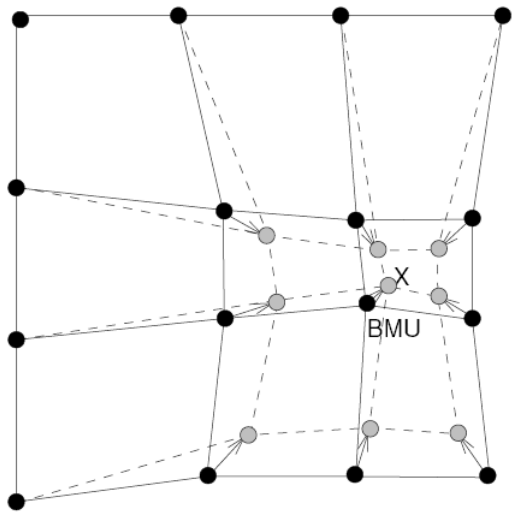
missä c on neuroni, jonka prototyyppivektori on lähimpänä datavektoria $\boldsymbol{x}_i$ ja termillä *arg* tarkoitetaan minimilausekkeen muuttujaa j. Jos vektorista $\boldsymbol{x}_i$ puuttuu arvoja, niin nämä arvot jätetään yleensä huomiotta etäisyyslaskennassa [62, sivu 12].

Seuraavaksi prototyyppivektoreita päivitetään kaavalla

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + \alpha(t)h_{cj}(t)[\boldsymbol{x}_i - \boldsymbol{m}_j(t)], \tag{4.3}$$

joka siirtää niitä kohti datavektoria $\boldsymbol{x}_i$, kuten kuvassa 4.3 on esitetty. Kaavassa $t = 0, 1, 2, \ldots$ on opetusaskeleen indeksi, $\alpha(t)$ on oppimisnopeuskerroin ja $h_{cj}(t)$ on naapurustofunktio [62, sivu 12]. Oppimisnopeuskerroin α on yleensä joko lineaarisesti tai funktion

$$\alpha(t) = \frac{A}{t + B} \tag{4.4}$$



Kuva 4.3: Päivitetään parasta vastaavaa yksikköä ja sen naapureita kohti syötevektoria $\boldsymbol{x}_i$. Mustat ja harmaat ympyrät kuvaavat tilannetta ennen ja jälkeen päivityksen. Kiinteä viiva ja katkoviiva kuvaavat naapuruston suhteita [62, sivu 12]. Juha Vesanto on antanut luvan kuvan käyttöön.

mukaisesti ajan suhteen vähenevä ja $0 < \alpha(t) < 1$. Oppimisnopeuskertoimen arvon tulisi vähetä kaavan 4.4 tyyppisesti, jotta kaikki opetusjoukon näytteet vaikuttaisivat muodostuvaan karttaan yhtä paljon [26].

Naapurustofunktiolla $h_{cj}(t)$ on keskeinen rooli algoritmissa. Konvergenssin kannalta on oleellista, että $h_{cj}(t) \to 0$, kun $t \to \infty$. Yleensä $h_{cj}(t) = h(\|\boldsymbol{r}_c - \boldsymbol{r}_j\|, t)$, missä $\boldsymbol{r}_c \in \mathbb{R}^n$ ja $\boldsymbol{r}_j \in \mathbb{R}^n$ ovat yksiköiden c ja j sijainteja SOM-ristikossa. Kun naapuriyksikköjen etäisyys $\xi_r = \|\boldsymbol{r}_c - \boldsymbol{r}_j\|$ kasvaa, niin naapurostofunktio $h_{cj} \to 0$. Voidaan sanoa, että naapurustofunktio määrää "elastisen pinnan" jäykkyyden, joka asetetaan vastaamaan datapisteitä [38, sivu 79].

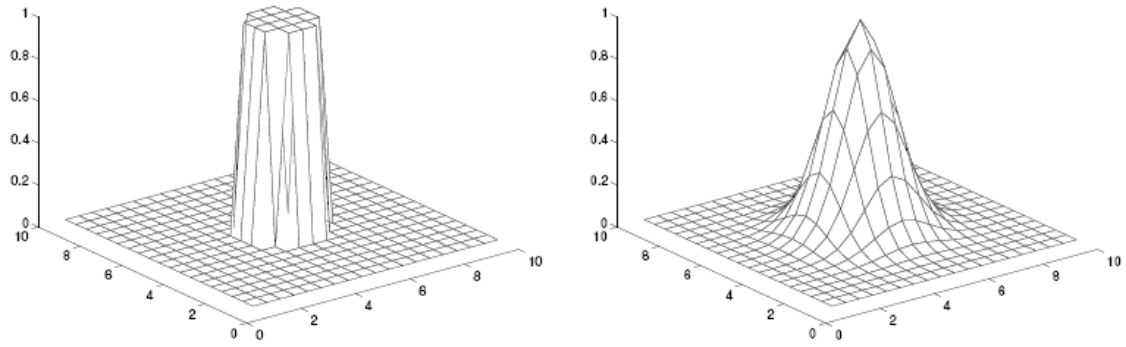
Naapurustofunktio voidaan esimerkiksi laskea Gaussian muodossa

$$h_{cj}(t) = e^{-\frac{\|\boldsymbol{r}_c - \boldsymbol{r}_j\|}{2\sigma^2(t)}}. \tag{4.5}$$

tai kuplanaapurusto muodossa

$$h_{cj}(t) = \begin{cases} 1, & \text{kun } \|\boldsymbol{r}_c - \boldsymbol{r}_j\| \leq \sigma(t) \\ 0, & \text{muulloin.} \end{cases} \tag{4.6}$$

Kaavoissa 4.5 ja 4.6 $\sigma(t)$ on naapuruussäde. Molemmat opetusvauhti $\alpha(t)$ ja naapuruussäde $\sigma(t)$ vähenevät monotonisesti opetuksen aikana, opetusvauhti kohti nollaa ja



Kuva 4.4: Vasemmalla kuvassa kupla naapurustofunktio ja oikealla gaussinen naapurustofunktio.

naapuruussäde kohti jotain sopivaa arvoa $y \neq 0$, yleensä kohti ykköstä [62, sivut 12–13].

On esitetty, että naapurustofunktion konveksisuudesta on etua. Gaussisesta naapurustofunktiosta voidaan leikata konveksit hännät pois, jolloin saadaan katkaistu gaussinen naapurustofunktio

$$h_{cj}(t) = \begin{cases} e^{-\frac{\|\boldsymbol{r}_c - \boldsymbol{r}_j\|}{2\sigma^2(t)}}, & \text{kun } \|\boldsymbol{r}_c - \boldsymbol{r}_j\| \leq \sigma(t) \\ 0, & \text{muulloin.} \end{cases} \tag{4.7}$$

Oppimisnopeuskerroin- ja naapurustofunktion muodot eivät ilmeisesti ole erityisen kriittisiä parametrivalintoja. On kuitenkin olemassa tuloksia, joiden perusteella konkaavi naapurustofunktio on suotuisampi kartan järjestymiselle kuin konveksi. Esimerkiksi Epanechnikovin naapurustofunktio

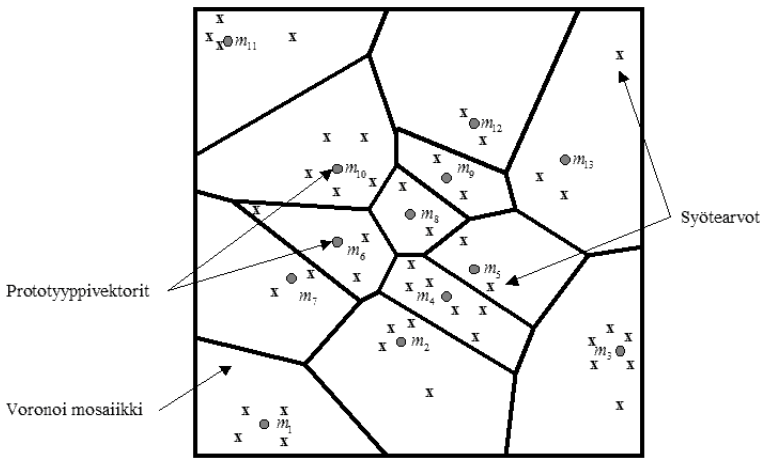
$$h_{cj}(t) = \begin{cases} 1 - \frac{\|\boldsymbol{r}_c - \boldsymbol{r}_j\|}{\sigma^2}, & \text{kun } \|\boldsymbol{r}_c - \boldsymbol{r}_j\| \leq \sigma(t) \\ 0, & \text{muulloin} \end{cases} \tag{4.8}$$

on konkaavi ja täyttää naapurustofunktion ehdot [26].

4.3.3 Eräalgoritmi

Eräalgoritmi (Batch Map) on SOM-versio, jossa yhdellä opetusaskeleella käydään läpi koko datajoukko [26]. Jokaisella opetusiteraatiolla etsitään parhaimmat vastaavat yksiköt (BMU) c kaikille datavektoreille kaavalla 4.2. Tämän jälkeen uudet prototyypit lasketaan kaavalla

$$\boldsymbol{m}_j = \frac{\sum_{i=1}^{N} h_{cj} \boldsymbol{x}_i}{\sum_{i=1}^{N} h_{cj}}. \tag{4.9}$$



Kuva 4.5: Voronoi-mosaiikki, jossa syötevektorit on merkitty symbolilla $\boldsymbol{x}$, mallivektorit $\boldsymbol{m}_i$ harmailla palloilla ja viivat on piirretty erottamaan mallivektorit ja niitä lähimpänä olevat syötevektorit.

Uudet prototyypit ovat datanäytteiden painotettuja keskiarvoja siten, että jokaisen datanäytteen paino on keskusnaapurustoarvo h_{cj} sen parhaimman vastaavan yksikön (BMU) c luona.

Vaihtoehtoisesti päivitys voidaan laskea painotettuna keskiarvona Voronoi-joukkojen keskiöistä $n_j = \frac{1}{N_j} \sum_{\boldsymbol{x}_i \in V_j} \boldsymbol{x}_i$ kaavalla

$$\boldsymbol{m}_j = \frac{\sum_{i=1}^{M} N_i h_{ij} n_i}{\sum_{i=1}^{M} N_i h_{ij}}. \tag{4.10}$$

Kaavassa N_j on näytteiden määrä Voronoi-joukossa V_j. Tämä mahdollistaa tehokkaamman matriisipohjaisen toteutuksen kuin kaavan 4.9 käyttö [62, sivu 13].

Määritelmä 4.3.1 *Voronoi-joukko* [38, sivut 37–38]

$$V_i = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{m}_i\| < \|\boldsymbol{x} - \boldsymbol{m}_j\|, \textit{kaikille } j \neq i\}, \textit{kun } i = 1, 2, \ldots, n. \tag{4.11}$$

Käytännössä mallivektorin $\boldsymbol{m}_i$ Voronoi-joukko V_i muodostuu mallivektoria $\boldsymbol{m}_i$ lähimpinä olevista syötevektoreista. Kun mallivektorit ja syötevektorit jaetaan omiin alueisiinsa saadaan kuvan 4.5 mukainen *Voronoi-mosaiikki* [33, sivu 38].

4.4 Karttojen arviointia

Algoritmin ja hyvien karttojen ymmärtäminen on tärkeää, koska eri parametreilla muodostetut kartat kuvaavat havaintoaineistoa eri tarkkuudella. Monissa algoritmeissa kustannus- tai voimafunktiot määrittelevät selkeästi optimaalisen tilanteen [62, sivu 13–14].

Yksinkertaisin mittari kartan erottelukyvyn mittaamiseksi on keskimääräinen kvantisointivirhe

$$\epsilon_q = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{m}_{ci}\|. \tag{4.12}$$

Kvantisointivirheessä lasketaan yhteen jokaisen datavektorin etäisyys lähinpään prototyyppivektoriin (BMU). Kahden kartan vertaaminen toisiinsa kvantisointivirheen avulla ei ole aina järkevää, sillä suuremmalla kartalla voidaan tietysti kvantisoida opetusjoukko tarkemmin. Samoin on ilmeistä, että pienempi naapuruston säde opetuksessa tuottaa usein tarkemman kvantisoinnin, sillä prototyyppivektorit voivat liikkua toisistaan suuremmin välittämättä. Vasta samankokoisten ja samalla naapurustofunktiolla opetettujen karttojen vertailu on mielekästä [26] [61, sivut 16–17].

Kuvauksen jatkuvuutta kuvataan topografisella virheellä ϵ_t. Yksinkertainen topografinen virhemitta on Kiviluodon esittämä virhemitta. Tässä kullekin datavektorille $\boldsymbol{x}_i$ lasketaan kaksi lähintä yksikköä $\boldsymbol{n}^i_{c1}$ ja $\boldsymbol{n}^i_{c2}$. Seuraavaksi lasketaan, kuinka monen datavektorin voittajayksikköparit $\boldsymbol{n}^i_{c1}$ ja $\boldsymbol{n}^i_{c2}$ eivät ole toistensa välittömiä naapureita hilassa. Virhemitta on tämän määrän suhde datavektorien määrään.

$$\begin{aligned} \epsilon_t &= \frac{1}{N} \sum_{i=0}^{N} u(\boldsymbol{n}^i_{c1}, \boldsymbol{n}^i_{c2}), \text{missä} \\ u(\boldsymbol{n}^i_{c1}, \boldsymbol{n}^i_{c2}) &= \begin{cases} 1, & \text{kun } \boldsymbol{n}^i_{c1} \text{ ja } \boldsymbol{n}^i_{c2} \text{ ovat välittömiä naapureita} \\ 0, & \text{muulloin.} \end{cases} \end{aligned} \tag{4.13}$$

Yksinkertaisuuden lisäksi tämän mitan etu on erilaisten karttojen ja datajoukkojen virheitten vertailukelpoisuus, koska karttojen koko, datan määrä ja muut parametrit eivät vaikuta tämän virhemitan skaalaan [26] [61, sivu 17].

Kaski ja Lagus ovat ehdottaneet yhdistettyä mittaa, topografista kvantisointivirhettä ϵ_{tq}, jossa erottelukyvyn ja jatkuvuuden mitat yhdistetään. Mitta on summa kahdesta etäisyydestä. Lasketaan syötevektorin etäisyys voittajayksikköön $\boldsymbol{n}_{c1}$, siis kvantisointivirhe. Tähän lisätään lyhin mahdollinen etäisyys $d = \|\boldsymbol{n}_{c1} - \boldsymbol{n}_{c2}\|$ niistä etäisyyksistä,

jotka saadaan $\boldsymbol{n}_{c1}$ ja $\boldsymbol{n}_{c2}$ välille aina vierekkäisestä yksiköstä toiseen kulkien. Kartan yhdistetty virhe datajoukon suhteen on datajoukon vektoreiden virheiden keskiarvo.

$$\epsilon_{tq} = \frac{1}{N}\sum_{i}^{N}(\|\boldsymbol{x} - \boldsymbol{m}_{c1}^{i}\| + \min \sum_{k=\boldsymbol{c1}(i)}^{\boldsymbol{c2}(i)} \|\boldsymbol{m}_{k' \in N_{k,1}}^{i} - \boldsymbol{m}_{k}^{i}\|), \tag{4.14}$$

jossa k korvataan k':lla jokaisen summausaskeleen jälkeen. $N_{k,1}$ tarkoittaa yksikön $\boldsymbol{n}_k$ lähimpiä naapureita hilassa [26] [61, sivu 18].

Kuitenkin, perus SOM-algoritmille on osoitettu, että ei ole olemassa hyvää kustannusfunktiota yleisessä tapauksessa [62, sivu 14]. Erilaisia virhemittoja on esitelty ja listattu lähteissä [62, sivut 13–16] ja [36].

5 Assosiaatiosäännöt

Assosiaatiosäännöt ovat yksi suosituimmista tiedonlouhinnan menetelmistä. Assosiaatiosääntöjen louhimisen tavoitteena on löytää kiinnostavia korrelaatioita ja säännönmukaisuuksia suuresta datajoukosta. Data koostuu alkiojoukoista ja assosiaatiosäännöt kuvaavat, kuinka todennäköisesti alkioiden eri kombinaatiot esiintyvät samassa alkiojoukossa [43, sivu 11]. Lisäksi assosiaatiosääntöongelmaa on sovellettu moniin erikoistarkoituksiin, kuten sekvenssisääntöihin/hahmoihin. Sekvenssihahmoissa etsitään alkioita, jotka edeltävät toisia alkioita. Sekvenssihahmo on tutkimus alkioiden järjestyksistä kun taas assosiaatiosääntö on tutkimus alkioiden yhteenkuulumisesta [65, sivu 1].

Sääntöjen louhinta koostuu toistuvien alkiojoukkojen etsimisestä, sääntöjen generoimisesta ja kiinnostavien sääntöjen etsimisestä sekä tulosten esittämisestä. Kaksi ensimmäistä vaihetta kuuluvat tietämyksen muodostamisprosessin tiedonlouhinnan osaan ja kaksi viimeistä hahmojen arviointiin ja tulosten esittämiseen.

Luvussa 5.1 esitellään assosiaatiosääntöjen ja historiaa ja lähtökohtia. Luvussa 5.2 määritellään assosiaatio- ja sekvenssianalyysin perusteita. Luvussa 5.3 esitellään toistuvien alkiojoukkojen etsimistä ja ongelmia. Luvuissa 5.4 ja 5.5 esitellään Apriori-algoritmi assosiaatiosääntöjen louhintaan ja PrefixSpan-algoritmi sekvenssihahmojen louhintaan. Luvussa 5.6 kerrotaan, kuinka laajasta sääntöjoukosta voidaan etsiä kiinnostavimmat säännöt. Tulosten visualisointitekniikoita on esitelty luvussa 6.4.

5.1 Historiaa ja lähtökohtia

Agrawal, Imielinski ja Swami [2] esittelivät assosiaatiosäännöt vuonna 1993, jonka jälkeen ne ovat olleet keskeinen tiedonlouhintamenetelmä ja tutkimuksen kohde. Ensimmäisenä sovelluskohteena oli kaupan asiakkaiden ostostapahtumien analysointi, jossa asiakkaiden ostostapoja tutkittiin etsimällä assosiaatioita eri tuotteiden välillä. Esimerkkinä assosiaatiosäännöstä voidaan ajatella seuraavaa: *"30% olutta sisältävistä ostoksista sisältää myös vaippoja, 2% kaikista ostoksista sisältää molempia artikkeleita"*. Tässä tapauksessa sääntö voi kannustaa seuraavaan päätöksentekoon: *"sijoitetaan oluet ja vaipat lähekkäin, jolloin asiakakkaan on helpompi poimia molemmat tuotteet"*. Vuon-

na 1994 Agrawal ja Srikant [1] esittelivät uraauurtavan Apriori-algoritmin toistuvien alkiojoukkojen totuusarvoisten assosiaatiosääntöjen louhintaan.

Alkuperäistä ongelmaa on myöhemmin kehitetty erikoistarkoituksiin, kuten määrälliset (quantitative) assosiaatiosäännöt [52], yleistetyt (generalized) assosiaatiosäännöt [51] [27] ja sekvenssihahmot (sequential patterns) [3] [42]. Lisäksi on olemassa useita yleistyksiä assosiaatiosääntöongelmasta [16] [58]. Edelleen on kehitetty algoritmeja, jotka louhivat hyvin sääntöjoukkojen osajoukkoja, esimerkiksi ennalta annettujen alkioiden tai kiinnostavuusmittojen mukaan, kuten yleiset rajoitteet [40] [54], optimoidut säännöt [19] [49], toistuvat maksimaaliset alkiojoukot [67] ja toistuvat suljetut alkiojoukot [46] [47]. Lisäksi on olemassa algoritmeja tiheiden tietokantojen louhintaan [8]. Näihin lähestymistapoja on sovellettu on-line-louhinta [25] ja inkrementaalisiin algoritmeihin [57] [7].

Sekvenssihahmot ovat laajennus assosiaatiosääntöihin, missä mukana on tapahtumien järjestys. Esimerkkinä sekvenssihahmosta voisi olla seuraavaa: *"asiakkaat vuokraavat "Star Wars" -elokuvan ja sitten "Empire Strikes Back" -elokuvan ja edelleen "Return of the Jedi" -elokuvan"*. Näiden tapahtumien ei välttämättä tarvitse olla peräkkäisiä. Asiakkaat, jotka vuokraavat muita elokuvia näiden elokuvien välissä tukevat myös tätä sekvenssihahmoa. Tapahtumat voivat olla myös alkiojoukkoja eivätkä pelkästään yksittäisiä alkioita.

Sekvenssihahmojen louhinta on monimutkaisempaa kuin assosiaatiosääntöjen louhinta, koska hahmoja ei muodosteta pelkästään vertailemalla alkioita vaan myös alkiojoukkojen järjestyksistä. Ensimmäiset algoritmit tämän ongelman ratkaisemiseksi olivat AprioriAll [3] ja GSP [53] (Agrawal ja Srikant). Näistä molemmat perustuvat Apriori-heuristiikkaan ja GSP suoriutui louhinnasta 20 kertaa nopeammin kuin AprioriAll. Sääntöjen louhiminen pohjautuu toistuvien alkiojoukkojen etsimiseen. Ensimmäisten Apriori-heuristiikkaan perustuvien algoritmien jälkeen on esitetty muita lähestymistapoja omaavia algoritmeja. Esimerkiksi FreeSpan [24] ja PrefixSpan [48], jotka perustuvat "pattern-growth"-menetelmään. Uusissa algoritmeissa toistuvien sekvenssihahmojen louhinnan suorituskyky on huomattavasti parempi kuin ensimmäisissä algoritmeissa.

5.2 Perusteet

Assosiaatiosäännöt ja sekvenssihahmot koostuvat samoista alkioista. Ainoana poikkeuksena on se, että sekvenssihahmojen louhinnassa on mukana aika. Myös näiden

tid	Tapahtuma
1	a, b, c, d, g, h
2	a, b, e, f
3	b, i, k
4	a, b, h, i
5	e, g, j

Taulukko 5.1: Esimerkki tapahtumatietokannasta.

louhimisessa on samoja ongelmia. Luvuissa 5.2.1 ja 5.2.2 määritellään assosiaatiosäännöt ja sekvenssihahmot. Luvussa 5.2.3 esitellään Apriori-ominaisuus ja yleisiä ongelmia sääntöjen louhinnassa.

5.2.1 Assosiaatiosääntö

Assosiaatiosääntöongelma määriteltiin alunperin seuraavasti [2]: *"Joukko tapahtumia (elementtejä), missä jokainen elementti koostuu joukosta alkioita sekä olkoon rajaarvo minimituki annettu. Assosiaatiosääntöjen louhinnan tavoitteena on löytää kaikki toistuvat alitapahtumat"*. Assosiaatiosääntö koostuu pohjimmiltaan alkioista. Olkoon $I = \{i_1, i_2, ..., i_n\}$ joukko erillisiä alkioita. Merkitään tapahtumaa $\alpha_j = (i_1 i_2 \dots i_k)$, missä alkio i_l kuuluu alkiojoukkoon I ($i_l \in I$) ja $l = 1, \dots, k$. Olkoon D (taulukko 5.1) louhinnan kannalta merkityksellinen joukko tietokannan tapahtumia, missä jokainen tapahtuma α_j on alkiojoukko siten, että α_j on alkiojoukon I epätyhjä osajoukko ($\alpha_j \subseteq I$ ja $\alpha_j \neq \emptyset$) ja α_j kuuluu tapahtumatietokantaan D ($\alpha_j \in D$). Jokainen tapahtuma on yksilöity omalla tunnuksella *tid*. Olkoon A epätyhjä alkiojoukko. Joukkoa $A \subseteq I$, missä $k = |A|$, kutsutaan k-alkiojoukoksi tai pelkästään alkiojoukoksi ja merkintä $|A|$ tarkoittaa alkiojoukon A alkioiden lukumäärää. Tapahtuman α_j sanotaan sisältävän alkiojoukon A, jos ja vain jos $A \subseteq \alpha_j$ [21, sivut 227–228]. Alkiojoukon A *tuki (support)* joukossa D on

$$support_D(A) = \frac{|\{\alpha_j \in D | A \subseteq \alpha_j\}|}{|D|}. \tag{5.1}$$

Assosiaatiosääntö on implikaatio $A \Rightarrow B$, missä $A \subset \alpha_j$, $B \subset \alpha_j$ ja $A \cap B = \emptyset$. Assosiaatiosääntö $A \Rightarrow B$ kiinnittää tapahtumajoukkoon D *tuen (support)*, joka on osuus tapahtumista tapahtumajoukossa D, jotka sisältävät $A \cup B$. Tämä voidaan esittää

todennäköisyytenä $P(A \cup B)$ tai kaavana

$$support_D(A \Rightarrow B) = \frac{|\{\alpha_j \in D | A \subset \alpha_j, B \subset \alpha_j \text{ ja } A \cap B = \emptyset\}|}{|D|}. \tag{5.2}$$

Assosiaatiosäännöllä $A \Rightarrow B$ on myös *luottamus (confidence)* joukossa D. Luottamus määritellään olevan osuus tapahtumista α_j, jotka sisältävät alkiojoukon A lisäksi alkiojoukon B, verrattuna kaikkiin tapahtumiin α_j, jotka sisältävät alkiojoukon A [21, sivut 227–228]. Tämä voidaan esittää ehdollisena todennäköisyytenä $P(B|A)$ tai kaavana

$$confidence_D(A \Rightarrow B) = \frac{support_D(A \cup B)}{support_D(A)}. \tag{5.3}$$

Kaava 5.3 osoittaa, että luottamus säännölle $A \Rightarrow B$ voidaan johtaa helposti alkiojoukkojen A ja $A \cup B$ tukien lukumääristä.

Tutkitaan tapahtumatietokantaa D (taulukko 5.1) minimituella 2. Tietokanta koostuu alkioista $\{a, b, c, d, e, f, g, h\}$. Tietokannasta saadaan muodostettua esimerkiksi seuraavat assosiaatiosäännöt: $a \Rightarrow b$: $tuki = \frac{3}{5} = 60\%$ ja $luottamus = \frac{3}{3} = 100\%$ ja $(a, b) \Rightarrow h$: $tuki = \frac{2}{5} = 40\%$ ja $luottamus = \frac{2}{3} = 66\%$. Laskettaessa säännön $(a, b) \Rightarrow h$ luottamusta kolme tapahtumaa sisältävät tapahtuman (a,b) (1, 2 ja 4), mutta vain kaksi näistä sisältävät lisäksi tapahtuman h (1 ja 4).

5.2.2 Sekvenssisääntö

Agrawal ja Srikant esittelivät vuonna 1995 sekvenssihahmojen louhintaongelman seuraavasti [3]: *"Joukko sekvenssejä, missä jokainen sekvenssi koostuu listasta elementtejä ja jokainen elementti koostuu joukosta alkioita sekä olkoon raja-arvo minimituki annettu. Sekvenssihahmojen louhinnan tavoitteena on löytää kaikki toistuvat alisekvenssit".*

Sekvenssi (sequence) on järjestetty lista tapahtumia tapahtumahetkien mukaan. Merkitään sekvenssiä $\alpha = \langle \alpha_1 \alpha_2 \cdots \alpha_q \rangle$, missä $\alpha_i \subseteq I$ on tapahtuma (tai elementti) ja $i = 1, \ldots, q$. Alkio voi esiintyä monta kertaa sekvenssissä eri tapahtumissa, mutta vain kerran yhdessä tapahtumassa. Alkioiden esiintymien lukumäärä sekvenssissä on sekvenssin pituus [48, sivu 3]. Esimerkiksi sekvenssiä, jossa on k alkiota ($k = \sum_j |\alpha_j|$) kutsutaan *k-sekvenssiksi (k-sequence)*. Sekvenssi $\langle b(ac) \rangle$ on tällöin 3-sekvenssi, jossa b merkitsee yhden alkion sisältämää tapahtumaa ja (ac) tapahtumaa joka sisältää kaksi alkiota.

Sanotaan, että sekvenssi $\alpha = \langle \alpha_1 \alpha_2 \cdots \alpha_n \rangle$ on sekvenssin $\beta = \langle \beta_1 \beta_2 \cdots \beta_n \rangle$ *alisekvenssi* ($\alpha \sqsubseteq \beta$) ja sekvenssi β sekvenssin α *ylisekvenssi*, jos on olemassa yksi-yhteen

sid	Sekvenssi
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle (ad)c(bc)(ae)\rangle$
30	$\langle (ef)(ab)(df)cb\rangle$
40	$\langle eg(af)cbc\rangle$

Taulukko 5.2: Esimerkki sekvenssitietokannasta S.

järjestyksen säilyttävä funktio f, joka kuvaa sekvenssin α tapahtumat sekvenssin β tapahtumiksi [68]. Toisin sanoen on olemassa kokonaisluvut $1 \leq j_1 < j_2 < \cdots < j_1 \leq m$ siten, että $\alpha_1 \subseteq \beta_{j_1}, \alpha_2 \subseteq \beta_{j_2}, \cdots, \alpha_n \subseteq \beta_{j_n}$ [48, sivu 3]. Eli $\alpha_i \subseteq f(\alpha_i) = \beta_{j_i}$ ja jos $\alpha_i < \alpha_j$ niin $f(\alpha_i) < f(\alpha_j)$.

Esimerkiksi sekvenssi $\langle b(ac)\rangle$ on sekvenssin $\langle (ab)e(acd)\rangle$ alisekvenssi, koska $b \subseteq (ab)$ ja $(ac) \subseteq (acd)$ ja tapahtumien järjestys säilyy. Toisaalta sekvenssi $\langle (ab)e\rangle$ ei ole sekvenssin $\langle (abe)\rangle$ alisekvenssi eikä päinvastoin.

Sekvenssien louhinnan tietokanta S (taulukko 5.2) koostuu kokoelmasta syötesekvenssejä (*sid,s*), missä *sid* on sekvenssin yksilöllinen tunnus ja s on sekvenssi. Lisäksi jokaisella tapahtumalla annetussa sekvenssissä on myös yksilöllinen tunnus *tid*. Sekvenssin *tuki (support)* tietokannassa S, on lukumäärä syötesekvensseistä tietokannassa S, jotka sisältävät sekvenssin α

$$support_S(\alpha) = |\{(sid, s)|(sid, s) \in S \text{ ja } \alpha \sqsubseteq s\}|. \tag{5.4}$$

Sanotaan, että sekvenssi α on toistuva, jos se esiintyy useammin kuin käyttäjän määrittelemä *minimituki (min_ support)* raja-arvo. Sekvenssiä α kutsutaan sekvenssihahmoksi sekvenssitietokannassa S, kun $support_S(\alpha) \geq min\_support$. Joukkoa toistuvista k-sekvensseistä merkitään F_k:lla. Toistuva sekvenssi on maksimaalinen, jos se ei ole minkään toisen toistuvan sekvenssin toistuva alisekvenssi [68].

Tutkitaan sekvenssitietokantaa S (taulukko 5.2) minimituella 2. Tietokanta S koostuu alkioista $\{a, b, c, d, e, f, g\}$. Sekvenssissä $\langle a(abc)(ac)d(cf)\rangle$) on viisi tapahtumaa (a), (abc), (ac), (d) ja (cf), joissa alkiot a ja c esiintyvät useammin kuin kerran eri tapahtumissa. Sekvenssi on 9-sekvenssi, koska siinä esiintyy yhdeksän alkiota. Taulukosta löydetään esimerkiksi kolmen mittainen sekvenssihahmo $s = \langle (ab)c\rangle$, koska minimituki on kaksi ja s on sekvenssin 10 sekä sekvenssin 30 alisekvenssi.

5.2.3 Apriori-ominaisuus ja louhinnan ongelmia

Useat assosiaatiosääntöjen ja niiden variaatioiden louhintaan kehitetyistä algoritmeista käyttävät Apriori-ominaisuutta, kuten Apriori [1], AprioriAll [3] ja GSP [53].

Määritelmä 5.2.1 (Apriori-ominaisuus) *Kaikki toistuvan alkiojoukon epätyhjät osajoukot ovat myös toistuvia.* Ominaisuus perustuu seuraavaan havaintoon: Jos alkiojoukko I ei täytä minimitukea, niin I ei ole toistuva eli $P(I) < min\_support$. Jos alkio A lisätään alkiojoukkoon I, niin tuloksena oleva alkiojoukko $I \cup A$ ei voi esiintyä useammin kuin I. Siksi $I \cup A$ ei ole toistuva eli $P(I \cup A) < min\_support$.

Sääntöjen louhinnassa on pääasiassa kaksi ongelmaa ratkaistavana. Ensimmäiseksi algoritmin kompleksisuus, koska alkiojoukkojen ja sääntöjen määrä kasvaa eksponentiaalisesti suhteessa alkioiden määrään [60, sivu 16]. Esimerkiksi tutkittaessa toistuvaa 100-alkiojoukkoa $\{i_1, i_2, \ldots, i_{100}\}$, alkiojoukko sisältää $\binom{100}{1}$ =100 toistuvaa 1-alkiojoukkoa: $\{i_1, i_2, \ldots, i_{100}\}$, $\binom{100}{2}$ toistuvaa 2-alkiojoukkoa: $\{(i_1, i_2), (i_1, i_3), \ldots, (i_{99}, i_{100})\}$ ja niin edelleen. Kokonaisuudessaan toistuvia alkiojoukkoja, joita edellä mainittu joukko sisältää on

$$\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}. \tag{5.5}$$

Alkiojoukkojen valtava määrä johtuu Apriori-ominaisuudesta. Pitkä alkiojoukko sisältää kombinatorisen luvun lyhempiä, toistuvia alialkiojoukkoja [22, sivu 6].

Toiseksi kiinnostavat säännöt pitää poimia generoitujen sääntöjen joukosta. Tämä saattaa olla vaivalloista, koska generoituja sääntöjä on normaalisti paljon verrattuna käyttökelpoisiin sääntöihin, joita on tyypillisesti vain pieni murto-osa. Louhinnasta saatuja joukkoja on tyypillisesti rajattu minimirajoitteilla, (laatumitoilla) *minimituki (min_support)* ja/tai *minimiluottamus (min_confidence)* sekä mahdollisesti muilla kiinnostavuusmetriikoilla (luku 5.6). Sääntöjä, jotka noudattavat määriteltyjä rajaarvoja kutsutaan vahvoiksi säännöiksi [21, sivut 227–228].

5.3 Toistuvien hahmojen louhinta

Hahmojen louhinta generoi usein hyvin suuren määrän toistuvia alkiojoukkoja ja sääntöjä (taulukko 5.3). Tällöin käyttäjä joutuu seulomaan läpi suuren määrän louhittuja sääntöjä löytääkseen näistä hyödylliset säännöt, mikä puolestaan taas vähentää louhinnan suorituskykyä ja tehokkuutta.

Data	Alkioita	Tuki [%]	Alkiojoukkoja	Luottamus [%]	Sääntöjä [kpl]
data2	32	10	188 642	80	1 241 871
data3	18	10	1 679	80	4 680
data2	32	50	198	80	781
data3	18	50	31	80	67

Taulukko 5.3: Alkiojoukkojen määrien vaikutus assosiaatiosääntöjen määriin.

Pasquier, Bastide, Taouil ja Lakhal [46] esittivät vuonna 1999 vaihtoehtoisen tavan toistuvien alkiojoukkojen louhintaan. Tässä tavassa ei louhita kaikkia toistuvia alkiojoukkoja vaan ainoastaan *toistuvat suljetut alkiojoukot (frequent closed itemset)* ja näitä vastaavat assosiaatiosäännöt.

Määritelmä 5.3.1 (Aito ylialkiojoukko) Alkiojoukko Y on alkiojoukon X *aito ylialkiojoukko,* jos X on alkiojoukon Y *aito osajoukko* eli $X \subset Y$.

Määritelmä 5.3.2 (Suljettu alkiojoukko) Alkiojoukko X on *suljettu* datajoukossa S, jos siellä ei esiinny aitoa ylialkiojoukko Y siten, että alkiojoukolla Y on sama tukilukumäärä joukossa S kuin alkiojoukolla X joukossa S.

Määritelmä 5.3.3 (Toistuva suljettu alkiojoukko) Alkiojoukko X on *toistuva ja suljettu alkiojoukko* joukossa S, jos X on sekä suljettu että toistuva joukossa S.

Toistuvien suljettujen alkiojoukkojen louhinnalla saadaan pienennettyä louhinnassa löytyvien alkiojoukkojen määrää hävittämättä kuitenkaan informaatiota. Tämä puolestaan parantaa suorituskykyä ja louhinnan tehokkuutta, koska assosiaatiosääntöjen muodostamiseen tarvittavien toistuvien alkiojoukkojen määrä vähenee.

Olkoon tutkittavana tapahtumatietokanta, joka sisältää ainoastaan kaksi tapahtumaa, $\{\langle i_1, i_2, \ldots, i_{100}\rangle; \langle i_1, i_2, \ldots, i_{50}\rangle\}$. Olkoon minimitukikynnys $min\_support = 1$. Tällöin löydetään kaksi toistuvaa suljettua alkiojoukkoa ja niiden tukiluvut, jotka ovat $C = \{(\langle i_1, i_2, \ldots, i_{100}\rangle : 1); (\langle i_1, i_2, \ldots, i_{50}\rangle : 2)\}$. Tämä on huomattava parannus verrattuna kaikkien toistuvien alkiojoukkojen määrään kaavassa 5.5. Joukko toistuvia suljettuja alkiojoukkoja sisältää kokonaisuudessaan tiedon toistuvista alkiojoukoista. Esimerkikisi joukosta C voidaan johtaa $\{\langle i_2, i_{45}\rangle : 2\}$, koska $\{\langle i_2, i_{45}\rangle : 2\}$ on osajoukko alkiojoukosta $\{\langle i_1, i_2, \ldots, i_{50}\rangle : 2\}$ sekä $\{\langle i_8, i_{55}\rangle : 1\}$, koska $\{\langle i_8, i_{55}\rangle : 1\}$ ei ole osajoukko edellisestä, mutta on osajoukko joukosta $\{\langle i_1, i_2, \ldots, i_{100}\rangle : 1\}$. Suljettuja toistuvia alkiojoukkoja on sovellettu mm. assosiaatiosääntöjen (CLOSET [47]) ja sekvenssihahmojen louhintaan (CloSpan [66]).

Scan i	C_i	F_i
1	$\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$, $\{f\}$, $\{g\}$, $\{h\}$, $\{i\}$, $\{j\}$, $\{k\}$	$\{a\}$, $\{b\}$, $\{e\}$, $\{g\}$, $\{h\}$, $\{i\}$
2	$\{a,b\}$, $\{a,e\}$, $\{a,g\}$, $\{a,h\}$, $\{a,i\}$, $\{b,e\}$, $\{b,g\}$, $\{b,h\}$, $\{b,i\}$, $\{e,g\}$, $\{e,h\}$, $\{e,i\}$, $\{g,h\}$, $\{g,i\}$, $\{h,i\}$	$\{a,b\}$, $\{a,h\}$, $\{b,h\}$, $\{b,i\}$
3	$\{a,b,h\}$	$\{a,b,h\}$
Generoidut assosiaatiosäännöt $(a \Rightarrow b : 60\%, 100\%)$, $(a \Rightarrow h : 60\%, 67\%)$, $(b \Rightarrow a : 80\%, 75\%)$, $(h \Rightarrow a : 40\%, 100\%)$, $(h \Rightarrow b : 40\%, 100\%)$, $(i \Rightarrow b : 40\%, 100\%)$, $(a, b \Rightarrow h : 60\%, 67\%)$, $(a, h \Rightarrow b : 40\%, 100\%)$, $(b, h \Rightarrow a : 40\%, 100\%)$		

Taulukko 5.4: Assosiaatiosääntöjen louhiminen Apriori-algoritmilla tietokannasta D minimituella 40% ja minimiluottamuksella 60%.

Edelleen voidaan vähentää louhinnan kohteena olevia alkiojoukkoja louhimalla ainoastaan *toistuvat maksimaaliset alkiojoukot (frequent maximal itemset)* [9].

Määritelmä 5.3.4 (Maksimaalinen toistuva alkiojoukko) Alkiojoukon X sanotaan olevan *maksimaalinen toistuva alkiojoukko* (tai max-alkiojoukko) joukossa S, jos X on toistuva ja ei löydy ylialkiojoukkoa Y siten, että $X \subset Y$ ja Y on toistuva joukossa S.

Maksimaalisesta alkiojoukosta voidaan johtaa molemmat toistuvat alkiojoukot $\{i_2, i_{45}\}$ ja $\{i_8, i_{55}\}$, mutta ei alkiojoukkojen tarkkoja tukilukuja. Tämä haitta on ainoastaan maksimaalisten alkiojoukkojen louhinnalla. Maksimaalisista toistuvista alkiojoukoista ja niiden tukiluvuista tiedetään, että kaikki osajoukot ovat toistuvia ja näiden osajoukkojen tukiluvut ovat vähintään sama kuin maksimaalisella toistuvalla alkiojoukolla [22, sivu 7]. Maksimaalisten alkiojoukkojen louhintaan on kehitetty algoritmeja, kuten MAFIA [12] ja GenMax [20].

5.4 Apriori-algoritmi

Agrawal and Srikant [1] esittelivät vuonna 1994 uraauurtavan Apriori-algoritmin toistuvien alkiojoukkojen totuusarvoisten assosiaatiosääntöjen louhintaan. Algoritmin nimi perustuu siihen, että algoritmi käyttää aikaisempaa tietämystä toistuvista alkiojoukoista (Apriori-ominaisuus määritelmä 5.2.1). Algoritmi käyttää iteratiivista lähestymista-

paa tunnetulle tasottaiselle haulle (level-wise search), jossa k-alkiojoukkoja käytetään etsittäessä $(k+1)$-alkiojoukkoja.

Liitteessä C on esitetty assosiaatiosääntöjen louhinta-algoritmi pseudo-koodina. Funktiossa *apriori* I = kaikkien alkioiden alkiojoukko, T = on tapahtumajoukko ja s_{min} (minimituki), c_{min} (minimiluottamus) sekä k_{max} (assosiaatiosäännön pituus) ovat assosiaatiosääntöjen rajoitteita. Lisäksi F_k:lla merkitään toistuvien k-alkiojoukkojen joukkoa ja C_k:lla merkitään mahdollisien toistuvien k-alkiojoukkojen ehdokasjoukkoa. Kierroksella $(k+1)$ edellisen kierroksen toistuvia k-alkiojoukkoja käytetään generoitaessa toistuvien $(k+1)$-alkiojoukkojen ehdokasjoukkoa C_{k+1}. Joukosta C_{k+1} valitaan ne $(k+1)$-alkiojoukot, jotka ovat toistuvia. Taulukossa 5.4 on kuvattu tapahtumatietokannan D (taulukko 5.1) toistuvien alkiojoukkojen louhiminen minimituella 40%.

Lopuksi vahvojen assosiaatiosääntöjen muodostaminen on suoraviivaista, kun toistuvat alkiojoukot tapahtumatietokannasta D on löydetty ja niiden tukiluvut laskettu. Kaavaan 5.3 pohjautuen voidaan assosiaatiosäännöt muodostaa seuraavasti. Jokaiselle toistuvalle alkiojoukolle l muodostetaan kaikki l:n epätyhjät osajoukot. Jokaiselle alkiojoukon l osajoukolle s muodostetaan sääntö "$s \Rightarrow (l-s)$", jos $\frac{support_D(l)}{support_D(s)} \geq min\_confidence$, missä $min\_confidence$ on minimiluottamus. Koska säännöt muodostetaan toistuvista alkiojoukoista, jokainen automaattisesti täyttää minimituen [21, sivu 236]. Taulukossa 5.4 on listattu tapahtumatietokannan D toistuvista alkiojoukoista muodostuvat assosiaatiosäännöt minimiluottamuksella 60%.

5.5 PrefixSpan-algoritmi

Pei et al. [48] esittelivät vuonna 2004 PrefixSpan-menetelmän sekvenssihahmojen louhintaan, joka useimmissa tapauksissa päihittää apriori-ominaisuuteen perustuvat algoritmit. Algoritmi on kuvattu pseudokoodina kuvassa 5.1.

Havainnollistetaan algoritmin toimintaa tutkimalla sekvenssitietokantaa 5.2 minimituella 2. Sekvenssihahmot voidaan louhia "prefix-projection"-menetelmällä seuraavilla askeleilla:

1. *Etsitään 1-sekvenssihahmot.* Käydään tietokanta läpi ja etsitään kaikki toistuvat alkiot sekvensseistä. Jokainen näistä on 1-sekvenssihahmo $\langle a \rangle : 4$, $\langle b \rangle : 4$, $\langle c \rangle : 4$, $\langle d \rangle : 3$, $\langle e \rangle : 3$ ja $\langle f \rangle : 3$, missä $\langle$hahmo$\rangle$: tukilukumäärä.

2. *Jaetaan etsintäavaruus.* Joukko sekvenssihahmoja voidaan jakaa kuuteen osajoukkoon etuliitännäisten (prefix) mukaan: (1) etuliite $\langle a \rangle$, (2) etuliite $\langle b \rangle$, ..., (6)

PrefixSpan($\langle\rangle$, 0,S)

1. Käydään läpi $S|_\alpha$ ja etsitään joukko toistuvia alkioita b siten, että
   (a) b voidaan koota sekvenssin α viimeisestä elementistä sekvenssihahmon muotoon tai
   (b) $\langle b\rangle$ voidaan liittää sekvenssihahmon muotoon.
2. Jokaiselle toistuvalle alkiolle b, liitä se sekvenssiin α
3. Muodostetaan jokaiselle α' α'-projisoitu tietokanta $S|_{\alpha'}$ ja kutsutaan funktiota *PrefixSpan*(α', $l+1$, $S|_{\alpha'}$).

Kuva 5.1: PrefixSpan-algoritmi.

etuliite $\langle f\rangle$. Katso taulukko 5.5.

3. *Etsitään sekvenssihahmojen osajoukot.* Osajoukot voidaan louhia muodostamalla vastaavat projisoidut tietokannat ja louhimalla jokainen rekursiivisesti. Projisoidut tietokannat ja sekvenssihahmot on listattu taulukossa 5.5. Louhintaprosessi etenee seuraavasti:

   a. *Etsitään sekvenssihahmot etuliitteellä* $\langle a\rangle$. Kerätään ainoastaan sekvenssit, jotka sisältävät etuliitteen $\langle a\rangle$. Lisäksi näistä sekvensseistä huomioidaan ne alisekvenssit, jotka alkavat etuliitteellä $\langle a\rangle$. Tällöin esimerkiksi sekvenssissä $\langle(ef)(ab)(df)cb\rangle$ huomioidaan ainoastaan alisekvenssi $\langle(\_b)(df)cb\rangle$ louhittaessa sekvenssihahmoja etuliitteellä $\langle a\rangle$. Merkintä $\langle(\_b)\rangle$ tarkoittaa, että viimeinen tapahtuma etuliitteessä (tässä tapauksessa etuliite on a) alkion b kanssa muodostavat yhden tapahtuman. Sekvenssit, jotka sisältävät etuliitteen $\langle a\rangle$ on projisoitu $\langle a\rangle$-projisoituun tietokantaan, joka sisältää neljä suffix-sekvenssiä: $\langle(abc)(ac)d(cf)\rangle$, $\langle(\_d)c(bc)(ae)\rangle$, $\langle(\_b)(df)cb\rangle$ ja $\langle(\_f)cbc\rangle$. Käymällä läpi $\langle a\rangle$-projisoitu tietokanta löydetään kaikki etuliitteellä $\langle a\rangle$ olevat 2-sekvenssihahmot, jotka ovat: $\langle aa\rangle : 2$, $\langle ab\rangle : 4$, $\langle(ab)\rangle : 2$, $\langle ac\rangle : 4$, $\langle ad\rangle : 2$ ja $\langle af\rangle : 2$. Edelleen kaikki $\langle a\rangle$ etuliitteiset sekvenssihah-

prefix	Projected (suffix) database	sekvenssihahmot
$\langle a \rangle$	$\langle (abc)(ac)d(cf) \rangle$, $\langle (\_d)c(bc)(ae) \rangle$, $\langle (\_b)(df)cb \rangle$, $\langle (\_f)cbc \rangle$	$\langle a \rangle$, $\langle aa \rangle$, $\langle ab \rangle$, $\langle a(bc) \rangle$, $\langle a(bc)a \rangle$, $\langle aba \rangle$, $\langle abc \rangle$, $\langle (ab) \rangle$, $\langle (ab)c \rangle$, $\langle (ab)d \rangle$, $\langle (ab)f \rangle$, $\langle (ab)dc \rangle$, $\langle ac \rangle$, $\langle aca \rangle$, $\langle acb \rangle$, $\langle acc \rangle$, $\langle ad \rangle$, $\langle adc \rangle$, $\langle af \rangle$
$\langle b \rangle$	$\langle (\_c)(ac)d(cf) \rangle$, $\langle (\_c)(ae) \rangle$, $\langle (df)cb \rangle$, $\langle c \rangle$	$\langle b \rangle$, $\langle ba \rangle$, $\langle bc \rangle$, $\langle (bc) \rangle$, $\langle (bc)a \rangle$, $\langle bd \rangle$, $\langle bdc \rangle$, $\langle bf \rangle$
$\langle c \rangle$	$\langle (ac)d(cf) \rangle$, $\langle (\_c)(ae) \rangle$, $\langle b \rangle$, $\langle bc \rangle$	$\langle c \rangle$, $\langle ca \rangle$, $\langle cb \rangle$, $\langle cc \rangle$
$\langle d \rangle$	$\langle (cf) \rangle$, $\langle c(bc)(ae) \rangle$, $\langle (\_f)cb \rangle$	$\langle d \rangle$, $\langle db \rangle$, $\langle dc \rangle$, $\langle dcb \rangle$
$\langle e \rangle$	$\langle (\_f)(ab)(df)cb \rangle$, $\langle (af)cbc \rangle$	$\langle e \rangle$, $\langle ea \rangle$, $\langle eab \rangle$, $\langle eac \rangle$, $\langle eacb \rangle$, $\langle eb \rangle$, $\langle ebc \rangle$, $\langle ec \rangle$, $\langle ecb \rangle$, $\langle ef \rangle$, $\langle efb \rangle$, $\langle efc \rangle$, $\langle efcb \rangle$
$\langle f \rangle$	$\langle (ab)(df)cb \rangle, \langle cbc \rangle$	$\langle f \rangle$, $\langle fb \rangle$, $\langle fbc \rangle$, $\langle fc \rangle$, $\langle fcb \rangle$

Taulukko 5.5: Projisoidut tietokannat ja sekvenssihahmot.

mot voidaan jakaa kuuteen osajoukkoon: (1) etuliitteellä $\langle aa \rangle$, (1) etuliitteellä $\langle ab \rangle, \ldots$ (6) etuliitteellä $\langle af \rangle$. Nämä osajoukot voidaan louhia muodostamalla näille vastaavat projisoidut tietokannat.

i. $\langle aa \rangle$-projisoitu tietokanta koostuu vain yhdestä epätyhjästä alisekvenssistä etuliitteellä $\langle aa \rangle$: $\langle (\_bc)(ac)d(cf) \rangle$. $\langle aa \rangle$-projisoitu tietokannan tutkiminen lopetetaan, koska yhdestä sekvenssistä ei voida muodostaan toistuvaa alisekvenssiä.

ii. $\langle ab \rangle$-projisoitu tietokanta koostuu kolmesta suffix-sekvenssistä: $\langle (\_c)a \rangle$, $\langle c \rangle$ ja $\langle (\_c)(ac)d(cf) \rangle$. Louhittaessa $\langle ab \rangle$-projisoitua tietokantaa löydetään neljä sekvenssihahmoa: $\langle (\_c) \rangle$, $\langle (\_c)a \rangle$, $\langle a \rangle$ ja $\langle c \rangle$. Kokonaisuudessaan nämä sekvenssihahmot ovat: $\langle a(bc) \rangle$, $\langle a(bc)a \rangle$, $\langle aba \rangle$ ja $\langle abc \rangle$.

iii. $\langle (ab) \rangle$-projisoitu tietokanta koostuu kahdesta sekvenssistä: $\langle (df)cb \rangle$ ja $\langle (\_c)(ac)d(cf) \rangle$. Tästä löydetään seuraavat sekvenssihahmot: $\langle c \rangle$, $\langle d \rangle$, $\langle f \rangle$ ja $\langle dc \rangle$.

iv. Loput $\langle ac \rangle$-, $\langle ad \rangle$- ja $\langle af \rangle$-projisoidut tietokannat muodostetaan ja louhitaan samalla tavalla.

b. *Etsitään sekvenssihahmot etuliitteellä $\langle b \rangle, \langle c \rangle, \langle d \rangle, \langle e \rangle$ ja $\langle f \rangle$ vastaavasti.* Tämä tehdään muodostamalla $\langle b \rangle$-, $\langle c \rangle$-, $\langle d \rangle$-, $\langle e \rangle$- ja $\langle f \rangle$-projisoidut tietokan-

nat ja louhimalla niitä samalla tavalla kuten edellä. Projisoidut tietokannat ja niistä löydetyt sekvenssihahmot on listattu taulukossa 5.5.

4. *Sekvenssihahmojen joukko on kokoelma hahmoja, jotka on löydetty yllä olevalla louhintaprosessilla.*

5.6 Kiinnostavat säännöt

Kun louhitaan sääntöjä, pitäisi pystyä kertomaan, mitkä säännöt ovat todennäköisimmin kiinnostavia käyttäjän kannalta. Useimmat algoritmit käyttävät tuki-luottamus -kehystä. Siitä huolimatta, vaikka minimituki ja minimiluottamus auttavat hylkäämään merkitsemättömät säännöt, useita ei kiinnostavia sääntöjä saattaa silti löytyä. Näin ollen kaikki vahvat säännöt eivät välttämättä ole kiinnostavia. Säännön kiinnostavuus voidaan päättää joko subjektiivisesti tai objektiivisesti. Lopullisesti ainoastaan käyttäjä voi päättää onko annettu sääntö kiinnostava vai ei. Siten subjektiivisesti päätellyn säännön kiinnostavuus voi vaihdella eri käyttäjillä. Ei-kiinnostavien sääntöjen poistamiseen voidaan käyttää objektiivisia kiinnostavuusmittareita, jotka perustuvat tilastollisiin tunnuslukuihin. Lisäksi oikean ja parhaan metriikan valitseminen on usein hankalaa, koska metriikoiden toimivuus riippuu kohdealueesta [55, sivu 5], jossain tapauksissa metriikka saattaa antaa virheellistä informaatiota säännön kiinnostavuudesta. Luvuissa 5.6.1 ja 5.6.2 käsitellään tapoja ja metriikoita, joilla sääntöjen kiinnostavuuksia voidaan tutkia.

5.6.1 Kiinnostavuus assosiaatioissa

Esimerkkinä voidaan analysoida ostotapahtumia pelien ja videoiden suhteen. Olkoon X tapahtumatietokanta asiakkaiden ostotapahtumista siten, että 10 000 tapahtumasta, 6000 sisältää pelejä ja 7500 videoita sekä 4000 tapahtumaa sisältävät näitä molempia. Etsitään sääntöjä siten, että minimituki on 30% ja minimiluottamus 60%. Tällöin löydetään sääntö $''ostaa(X, pelin) \Rightarrow ostaa(X, videon)''$, missä tuki on 40% ja luottamus 66%. Kuitenkin todennäköisyys videon ostamiselle on 75%, joka on enemmän kuin 66%. Tosiasiassa pelin ja videon osto on negatiivisesti assosioitu, koska toisen ostaminen vähentää toisen ostamisen todennäköisyyttä.

Esimerkki osoittaa, että säännön $A \Rightarrow B$ luottamus voi olla harhaanjohtava, koska se on vain estimaatti ehdollisesta todennäköisyydestä. Se ei mittaa todellista vahvuutta (tai vahvuuden puutetta) alkiojoukkojen A ja B implikaatiosta [21, sivut 259–260].

Monet tekniikat assosiaatiosääntöjen louhinnassa ja ominaisuuksien valinnassa tarvitsevat sopivan metriikan. Esimerkiksi metriikat tuki, luottamus, noste (lift) ja kokonaisvahvuus ovat usein käytettyjä määräämään assosiaatiohahmojen kiinnostavuutta [55, sivu 1]. Kuitenkin monet mittareista antavat ristiriitaista informaatiota kiinnostavuudesta, kuten edellä osoitettiin. Tan, Kumar ja Srivastava [55] ovat listanneet erilaisia metriikoita (liite A) sekä hyvien metriikoiden toivottavia ominaisuuksia.

Esimerkiksi kun tutkitaan metriikkaa *Interest (I)* alkiojoukon A esiintyminen on riippumaton alkiojoukon B esiintymisestä, jos $P(A,B) = P(A)P(B)$, muutoin alkiojoukot A ja B ovat toisistaan riippuvia Riippuvuussuhde alkiojoukkojen A ja B esiintymien välillä voidaan mitata kaavalla

$$I_{A,B} = \frac{P(A,B)}{P(A)P(B)}. \tag{5.6}$$

Jos tulos kaavasta 5.6 on vähemmän kuin 1, niin silloin alkiojoukon A esiintyminen on negatiivisesti korrelaatiossa alkiojoukon B esiintymisen kanssa. Jos kaavan tulos on isompi kuin yksi, niin A ja B ovat positiivisesti korrelaatiossa, mikä tarkoittaa, että toisen esiintyminen merkitsee toisenkin esiintymistä. Jos tulos on tasan yksi, niin silloin A ja B ovat riippumattomia ja niiden välillä ei ole korrelaatiota.

Kaava 5.6 on ekvivalentti kaavan $P(B|A)/P(B)$, jota kutsutaan myös assosiaatiosäännön $A \Rightarrow B$ nosteeksi (lift). Esimerkiksi, jos A merkitsee pelien ja B videoiden myyntiä, pelien myynti lisää (tai nostaa) videoiden myynnin todennäköisyyttä kaavan antamalla arvolla [21, sivut 260–261].

5.6.2 Kiinnostavuus sekvensseissä

Sekvenssien kiinnostavuuden määrittely on hankalaa, koska sekvenssissä esiintyviä alkiojoukkoja on yleensä useita. Yleisesti käytössä oleva kiinnostavuusmetriikka myös sekvenssihahmoissa on minimituki. Hyvä ominaisuus sekvenssin tuessa on se, että sekvenssin α alisekvensseillä on vähintään yhtä suuri tukiluku kuin sekvenssillä α, $support_S(\alpha) \leq support_S(\beta)$, missä $\beta \sqsubset \alpha$. Tämä pätee myös assosiaatiosäännön alkiojoukoille $support_D(A) \leq support_D(B)$, missä $B \subset A$. Riippuen kohdealueesta tämä tuki voidaan laskea eri tavoin ja määritellä joko absoluuttisena lukumääränä tai prosentteina johonkin lähtökohtaan pohjautuen.

Joshi et al. [32, sivut 8–12] ovat määritelleet erilaisia tapoja sekvenssihahmojen tuen laskemiseen. Tavat on jaoteltu kolmeen eri ryhmään. Ensimmäisessä lasketaan pelkästään sekvenssihahmon esiintymiset sekvensseissä. Tämä voidaan tehdä niin, että

lasketaan kaikki esiintymät yhdessä sekvenssissä tai huomioidaan vain yksi esiintymä jokaisesta sekvenssistä. Seuraavassa ryhmässä sekvenssihahmojen esiintymät lasketaan annetussa aikaikkunassa. Kolmannessa ryhmässä laskeminen perustuu sekvenssien erillisiin esiintymisiin.

Kun sekvenssihahmoja käytetään ennustamisessa, voidaan niille määrittää luottamus samalla tavoin kuin assosiaatiosäännössä (5.3)

$$confidence_S((ab)c \rightarrow d) = \frac{support_S(\langle(ab)cd\rangle)}{support_S(\langle(ab)c\rangle)}. \tag{5.7}$$

Esimerkiksi sekvenssihahmo $\langle a(bc)\rangle$ kertoo, että kun a esiintyy niin (bc) esiintyy annetussa aikarajassa. Tämä hahmo voidaan muuttaan muotoon $a \rightarrow (bc)$. Samoin hahmo $\langle(ab)cd\rangle$ ennustaa alkion d esiintymistä hahmon $\langle(ab)c\rangle$ esiintymisen jälkeen $(ab)c \rightarrow d$ todennäköisyydellä $confidence_S((ab)c \rightarrow d)$.

Sekvenssihahmojen ja assosiaatiosääntöjen määrää voidaan vähentää myös klusteroimalla datajoukko ja valitsemalla jokaisesta muodostuneesta klusterista yksi prototyyppi edustamaan kyseistä joukkoa. Tällöin sääntöjä muodostuu vähemmän ja kiinnostavien sääntöjen löytäminen on helpompaa. Toisaalta sääntöjä voidaan tutkia myös siten, että käyttäjä määrittelee kiinnostavien sääntöjen ominaisuuksia. Tämän jälkeen säännöt, jotka vastaavat määrittelyä etsitään sääntöjoukosta. Lisäksi sääntöjoukot voidaan klusteroida samankaltaisiin joukkoihin [41].

6 Visualisointi

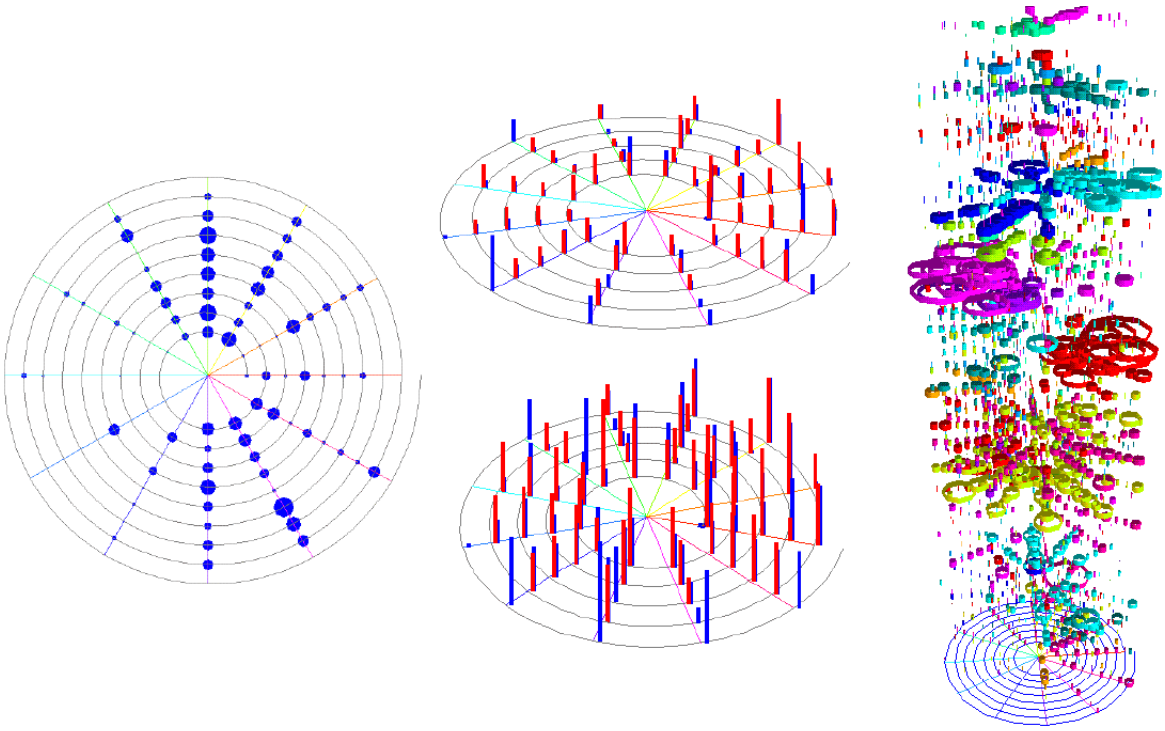
Tietämyksen muodostamisprosessi ei pääty siihen, kun kiinnostavat säännöt tai hahmot on löydetty. Data tutkittavasta ilmiöstä täytyy visualisoida käyttäjälle ymmärrettävässä ja tiiviissä muodossa, jotta datassa olevan informaation muodostaminen olisi helppoa.

Luvussa 6.1 kerrotaan visualisoinnin lähtökohtia. Luvussa 6.2 esitellään tekniikoita aikariippuvan datan visualisointiin. Luvuissa 6.3 ja 6.4 käydään läpi itseorganisoituvan kartan sekä sääntöjen visualisointeja.

6.1 Visualisoinnin lähtökohtia

Yleisesti ottaen tiedonlouhinnassa on olemassa kaksi lähestymistapaa visualisointiin ja visuaaliseen tiedonlouhintaan suhteessa visualisoitavaan sisältöön [69, sivu 1]. Ensimmäinen lähestymistapa on havaintoaineiston visualisointi. Visualisoitaessa käsittelemätöntä havaintoaineistoa säilytetään alkuperäisen aineiston ominaisuudet. Tässä jokainen havaintoaineiston alkio visualisoidaan erikseen. Tällöin käyttäjä voi saada paremman ymmärryksen käsiteltävästä datasta ja sen käyttäytymisestä. Toinen lähestymistapa on käyttää visualisointitekniikoita esittämään tiedonlouhinnan avulla saatua jalostettua dataa, kuten klustereita ja assosiaatiosääntöjä. Lisäksi visualisoinnissa voidaan yhdistää edellä mainitut tekniikat. Louhija ymmärtää paremmin sen mitä on löydetty, kun data ja hahmot visualisoidaan samanaikaisesti, suhteessa taustatietämykseen, aikaan jne. [43, sivut 4–5].

Ihminen on erittäin hyvä käsittelemään visuaalista dataa, koska aisteistamme, näköaisti on tärkein ja informatiivisin. Ihmisen on helpompi huomata kiinnostavat ilmiöt visuaalisesti kuvasta kuin tekstimuotoisista louhintatuloksista [69, sivu 1]. Ilmiöt, joista data on kerätty, kannattaa visualisoida mahdollisimman luonnollisessa muodossa. Jos visualisoitavan ilmiön yhtenä ulottuvuutena on aika, on luonnollista esittää ilmiö animaationa tai aikajanalla. Pelkkä staattinen kuva ei välttämättä riitä. Usein ilmiölle on löydettävissä luonnollinen esitystapa (esimerkiksi uutislähetysten sääennustekuvat), mutta toisinaan joudutaan kokeilemaan ja etsimään oikeaa näkökulmaa ja visualisointitapaa. Voi myös olla, että tarvitaan useita vaihtoehtoisia menetelmiä, koska



Kuva 6.1: Spiraalipohjaisia visualisointeja. John Carlis on antanut luvan kuvan käyttöön.

yksi näkökulma ei välttämättä riitä. Tällöin visualisointityökalujen tulisikin mahdollistaa tehokas ja joustava esitystavan valinta. Parhaassa tapauksessa visualisoinnista tehdään interaktiivinen, jolloin käyttäjä voi välittömästi vaikuttaa visualisoinnin toteutukseen. Kokoelmia visualisointitekniikoista löytyy mm. lähteistä [69, sivu 1], [14, sivut 2–3], [31] ja [50]

6.2 Aikariippuvan havaintoaineiston visualisointeja

Aikariippuvan havaintoaineiston visualisoinnissa on perinteisesti käytetty (x, y)- tai (x, y, z)-kuvaajia, joissa yhtenä akselina on aika. Visualisoitaessa dataa perinteisillä kuvaajilla on järkevää visualisoida vain pieni määrä kerrallaan, koska visualisoitaessa suuria määriä yhdellä kertaa saatetaan hukkua informaation tulvaan tai datassa olevaa informaatiota on hankala havaita.

Perinteisiä kuvaajia hyödyntäviä datan esittämistyökaluja on kehitetty useita. Hyvänä esimerkkinä on Human-Computer Interaction Lab:n kehittämä TimeSearcher [6],

joka on interaktiivinen työkalu datan hahmojen ja suuntauksien etsimiseen aikariippuvaisesta datasta.

Nämä perinteiset visualisointitekniikat ovat hyödyllisiä, mutta ne ovat rajoittuneet pieniulotteiseen ja pienikokoiseen datamäärään. Jos ilmiöstä kerätty data on hyvin suuriulotteista, niin on vaarana hukkua eri muuttujien antamien näkökulmien joukkoon. Lisäämällä animaatio visualisointiin, saadaan tällöin yksi ulottuvuus lisää staattiseen visualisointiin. Tämä on hyvä tekniikka joissain tapauksissa, mutta esimerkiksi etsittäessä säännönmukaisuuksia säännöllisestä datasta on vaikea muistaa mitä animaatiossa aikaisemmin tapahtui. Lisäksi visualisoinnissa ulottuvuuksia voidaan lisätä esimerkiksi antamalla kuvakkeille erilaisia ominaisuuksia. Jos dataa on visualisoitu aikajanalla, niin aika voidaan siirtää animaatioksi. Tällöin yksi akseli aikajanalta vapautuu uudelle muuttujalle. Lisäksi värit, liike, alueiden reunat ja koot sekä pintojen muoto välittävät tehokkaasti informaatiota. Näitä kuvakkeita, jotka ilmaisevat datan ominaisuuksia, kutsutaan myös glyyfeiksi (glyphs). Glyyfit ovat visuaalisia objekteja, joiden ominaisuudet kytketään dataan joko sijainnilla, värillä muodolla tai koolla [13, sivu 7]. Toisena esimerkkinä glyyfeistä on sääkartta, jossa sade, tuuli ja pilvisyys esitetään omilla glyyfeillä eli tunnuksilla. Näiden käyttö visualisoinnissa on niin yleistä ettei niitä aina mielletä glyyfeiksi.

Carlis ja Konstan [14] esittelivät vuonna 1998 spiraalitekniikan visualisoimaan jatkuvaa (serial) ja jaksollista (periodic) dataa (kuva 6.1). Visualisointitekniikassa data esitetään pitkin spiraalia, jossa jatkuvat attribuutit visualisoidaan spiraalin akselilla ja jaksolliset attribuutit säteittäisesti. Kuvassa 6.1 vasemmalla on visualisoitu simpanssien syömiskulutuksia erään kasvin osalta kuukausittain kahdeksan vuoden ajalta. Kuvasta nähdään esimerkiksi, että simpanssit syövät kasvin uusia lehtiä sadekauden alussa ja lopussa. Kuvassa oikealla on eriteltynä kaikkien tutkittujen kasvien kulutus samalta aikajaksolta. Useita eri datajoukkoja voidaan myös visualisoida samanaikaisesti erilaisilla glyyfeillä, kuten kuvassa 6.1 keskellä ja oikealla.

6.3 Itseorganisoituvan kartan visualisointeja

Itseorganisoituvan kartan pääkäyttötarkoitus on yleensä suuriulotteisen datan visualisointi, johon se on alunperin kehitetty [38, sivu 109]. Moniulotteinen data on ihmiselle hankalasti käsitettävää, joten topologian säilyttävä kuvaus alempaan dimensioon mahdollistaa datan kuvaamisen inhimillisesti käsitettävässä muodossa.

Luvussa 6.3.1 esitellään itseorganisoituvan kartan komponettitasojen visualisointia.

Luvuissa 6.3.2 esitellään miten saadaan selville, mikä osa kartasta vastaa annettua dataa ja kuinka peräkkäiset datavektorit käyttäytyvät ajan kuluessa. Luvussa 6.3.3 esitellään klusterointia ja klustereiden nimentää.

6.3.1 Komponettitasot

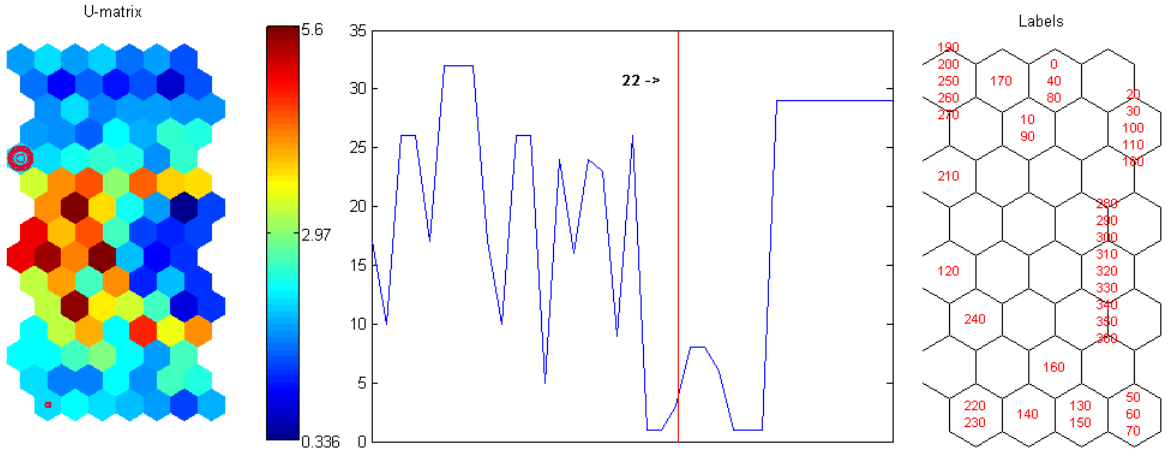
Kun itseorganisoituva kartta on muodostettu se voidaan visualisoida viipaloimalla kartta komponettitasoihin (component planes) (liiteet D ja E), jolloin nähdään kuinka tietyn komponentin (muuttujan) arvot vaihtelevat kartan eri osissa. Jokainen taso (viipale) j esittää yhden muuttujan arvoa neuroneiden prototyyppivektoreissa m_i siten, että muuttujien järjestys kartalla säilyy. Komponenttitasot ovat tärkeä tekniikka muuttujen korrelaatioden havaitsemiseen ja arvojen vertailemiseen.

6.3.2 Osumat ja trajektorit

Datavektorille x voidaan hakea kartalta voittajayksikkö (BMU), jota kutsutaan osumaksi. Osumilla voidaan tutkia mikä osa kartasta vastaa parhaiten annettua dataa. Osumien määrää klustereihin kuvassa 6.2 oikealla voidaan visualisoida laskemalla määrät ja kuvaamalla osumien suhteellista määrää visualisoitavan symbolin koolla. Jos data on kerätty prosessista, voi olla mielekästä yhdistää ajassa perättäisiä osumia toisiinsa trajektoriksi, kuten kuvassa 6.2. Visulisoinnissa trajektoria siirtämällä näytetään sen ajanhetken datavektorin osuma U-matriisissa. U-matriisilla kuvataan mallivektorien välisiä etäisyyksiä. Tässä yhteydessä karttakuvauksen topografinen virhe voi aiheuttaa ongelmia, sillä trajektori voi hyppelehtiä kartalla toisistaan kaukana olevien alueiden välillä, vaikka siirtymä syöteavaruudessa onkin todellisuudessa pieni. Tämä voi olla omiaan antamaan väärän kuvan dataa tuottaneen systeemin toiminnasta.

6.3.3 Klusterointi ja nimentä

Kartan yksiköt voidaan nimetä, jos tunnetaan joidenkin datavektorien luokka (kuvassa 6.2 oikealla). Kartalta haetaan datavektoreille voittajayksiköt ja nimetään kukin yksikkö siihen osuneen datan luokan mukaan. Jos osumia on useampia, voidaan käyttää erilaisia keinoja nimen määräämiseksi. Esimerkiksi valitaan kaikkien osuneiden vektorien nimet tai valitaan sen luokan nimi, jolla on eniten osumia (äänestys). Kartan nimentää kutsutaan myös kartan kalibroinniksi. Nimettyä karttaa voidaan vastaavati käyttää datan luokittelussa, jolloin kartalta haetaan voittajayksikkö ja annetaan datavektorille



Kuva 6.2: Itseorganisoituvan kartan visualisointeja. Vasemmalla U-matriisi. Keskellä trajektori. Oikealla nimetyt klusterit.

voittajayksikön luokkanimi tai -nimet.

6.4 Sääntöjen visualisointi

Assosiaatiosääntöjen visualisointi on yksi yhteen (one-to-one) tai monta yhteen (many-to-one) kuvaus alkioista. Visualisoitavana on vähintään viisi parametria. Joukot edeltävistä alkioista (antecedent), seurauksena olevat alkiot (consequent), assosiaatiot näiden välillä, sääntöjen tuet ja sääntöjen luottamukset. Lisäksi visualisoinnin kohteena voisivat olla liitteessä A olevat kiinnostavuusmittarit. Vallitsevat tekniikat assosiaatiosääntöjen visualisointiin ovat kaksi- ja kolmeulotteiset matriisit sekä suunnatut graafit [64]. Luvussa 6.4.1 esitellään matriisipohjaisia assosiaatioiden visualisointitekniikoita. Luvussa 6.4.2 esitellään visuaalinen tekniikka assosiaatiosääntöjen analysoimiseksi. Luvussa 6.4.3 esitellään sekvenssihahmojen visualisointia.

6.4.1 Matriisit

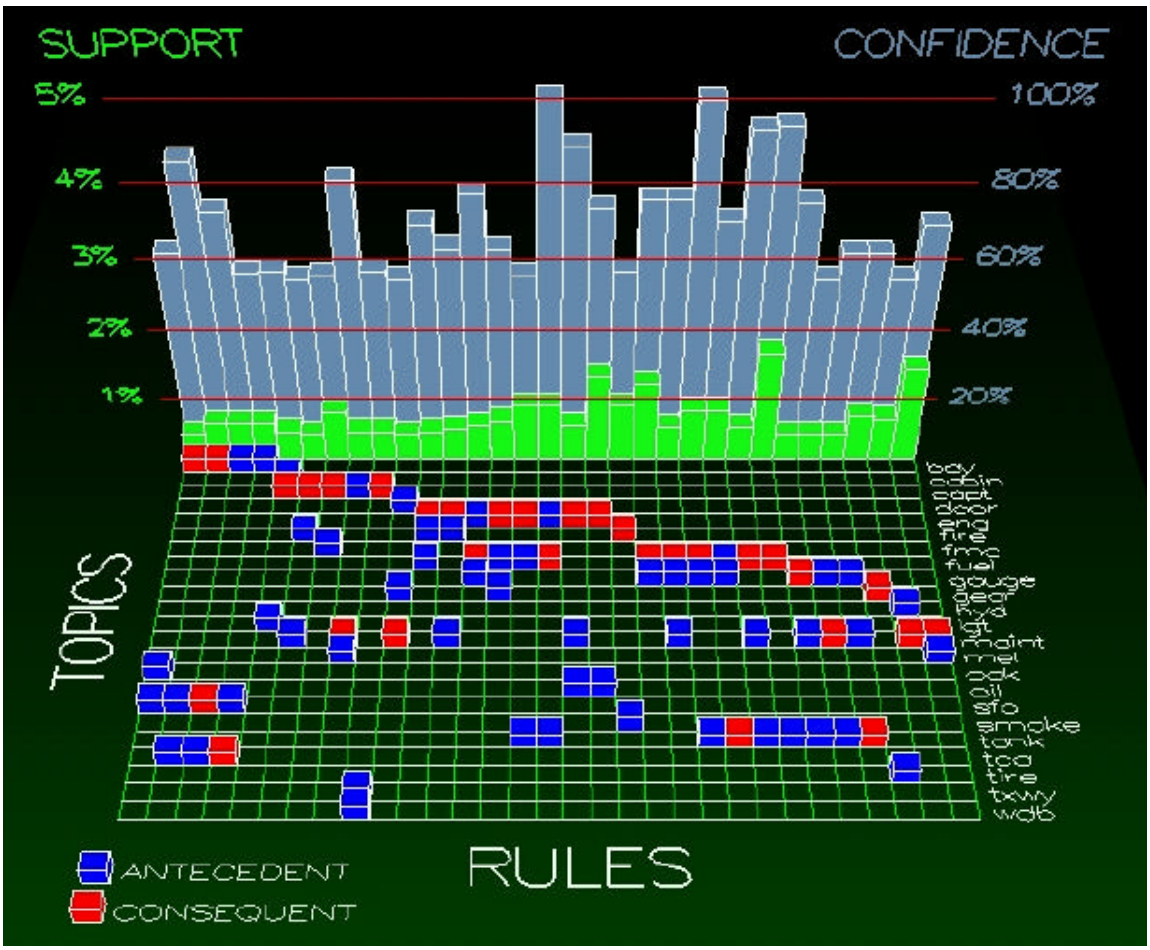
Perusajatus matriisipohjaisissa visualisointitekniikoissa on asettaa edeltävät ja seuraksena olevat alkiot eri akseleille neliömatriisissa. Piirtämällä kuvakkeita tietylle matriisin laatalle kuvataan edeltävien ja seurauksena olevan alkion välistä assosiaatiosääntöä. Kuvakkeiden muodolla, kuten pylvään korkeudella ja värillä, voidaan esittää assosiaatiosäännön ominaisuuksia. Kuvassa 6.3 vasemmanpuoleinen visualisointi esittää sään-

Kuva 6.3: Vasemmalla: assosiaatiosääntö $B \Rightarrow C$. Esittääkö keskimmäinen kuva assosiaatiosääntöä $A + B \Rightarrow C$ vai sääntöjä $A \Rightarrow C$ ja $B \Rightarrow C$. Oikealla: taustalla olevat assosiaatiosäännöt ovat näkymättömissä [64, sivut 2–3]. Pak Chung Wong on antanut luvan kuvan käyttöön.

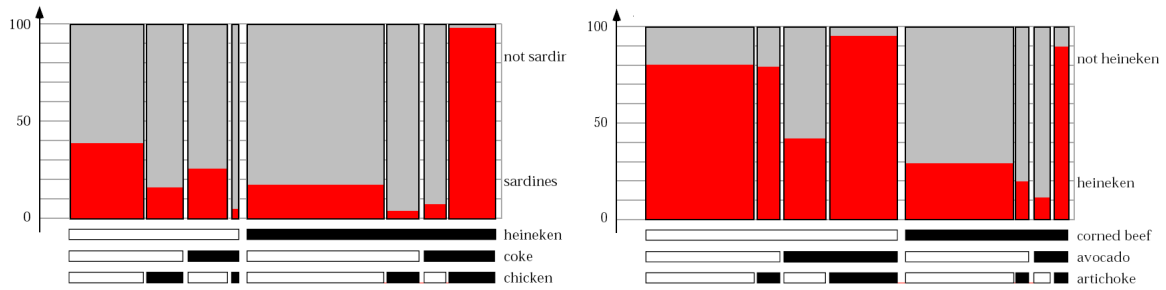
töä $B \Rightarrow C$. 2-ulotteinen matriisi on todennäköisesti tehokkain tekniikka visualisoida one-to-one suhteita.

Kun visualisoidaan monesta yhteen suhteita 2-ulotteisessa matriisissa, tulee sääntöjen visuaalinen tarkastelu vaikeaksi. Kuvan 6.3 keskimmäinen visualisointi osoittaa, että on lähes mahdotonta sanoa onko kyseessä yksi $A + B \Rightarrow C$ vai kaksi $A \Rightarrow C$ ja $B \Rightarrow C$ assosiaatiosääntöä. Tämä ongelma voidaan ratkaista muodostamalla matriisin akseleille alkiojoukkoja. Tällöin säännöt tai sääntö voidaan visualisoida yksikäsitteisesti. Ratkaisu toimii hyvin pienillä edeltäjä joukoilla. Joukkojen kasvaessa alkaa ilmentyä toisintoja, jossa alkiojoukoissa esiintyy samoja alkioita. Tämä osoittaa, että 2-ulotteinen matriisi on huono valinta useita edeltäjiä sisältävien assosiaatiosääntöjen visualisointiin. Toinen ongelma visualisoitaessa assosiaatiosääntöjä edellä esitetyllä tekniikalla ilmenee oikealla olevassa visualisoinnissa kuvassa 6.3. 2-ulotteisessa matriisissa edessä olevat säännöt tukkivat näkyvyyden takana oleville säännöille [64].

Wong, Whitney ja Thomas [64] esittelivät vuonna 1999 uuden tavan visualisoida assosiaatiosääntöjä (kuva 6.4). Tässä tavassa visualisoidaan (rule-to-item)-suhteita (item-to-item)-suhteiden sijasta. Matriisin pohjan rivit esittävät alkioita ja sarakkeet assosiaatioita alkiohin. Jokainen sarake esittää siis yhtä sääntöä ja säännön edeltävät alkiot on kuvattu eri värillä kuin seurauksena oleva. Alkioiden kuvaukset näkyvät oikealla ja säännön luottamus- ja tukiluvut on visualisoitu matriisin taustalla pylväinä. Lisäksi nämä luottamus- ja tukiluvut on skaalattu eri väleille, jotta edessä oleva tuki ei estäisi luottamuksen näkymistä. Tällä tekniikalla on monia etuja muihin aikaisem-



Kuva 6.4: Assosiaatiosäännöt kuvattuna 3d-matriisissa [64, sivu 4]. Pak Chung Wong on antanut luvan kuvan käyttöön.



Kuva 6.5: Mosaiikkipalstat assosiaatiosääntöjen visualisoinnissa [29]. Heike Hofmann on antanut luvan kuvan käyttöön.

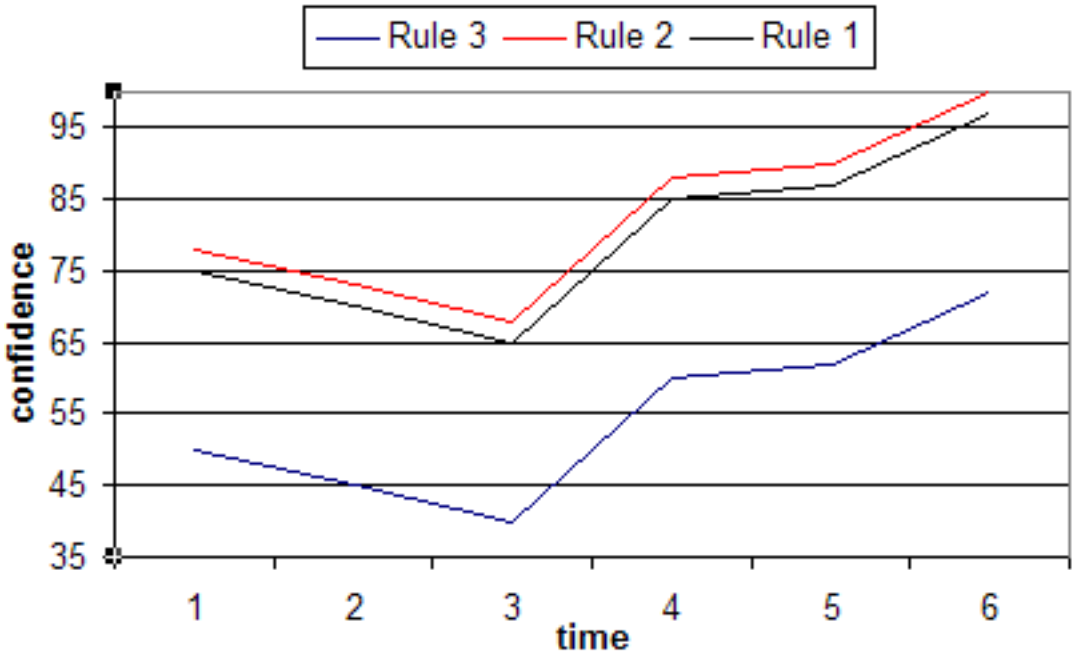
piin matriisipohjaisiin visualisointitekniikoihin verrattuna. Käytännöllisesti katsottuna edeltävien alkioiden määrälle ei ole ylärajaa. Lisäksi voidaan analysoida assosiaatiosääntöjen jakaumaa (vaakasuoralla akselilla) samanaikaisesti alkioiden kanssa (pystyakselilla). Sääntöjen edeltävät alkiot nähdään selvästi ja uusia edeltäjä joukkoja ei tarvitse lisätä matriisiin. Visualisoinnissa esiintyy hyvin vähän päällekkäisyyttä, koska sääntöjen metatieto on visualisointu matriisin taustalle sekä skaalattu. Ei tarvita ruudun vaihtoja, animaatioita tai interaktiivisuutta analysoitaessa (muuta kuin perus zoomauksia) dataa.

Assosiaatioiden tarkempaan analysointiin ja alkiojoukkojen välisten suhteiden ja riippuvuuksien ymmärtämiseen voidaan käyttää mosaiikkipalstoja (kuva 6.5). Tässä assosiaatioita esitetään palstoina ja sääntöjen kiinnostavuuksia palstan koolla ja täyttöasteella [29].

6.4.2 Sääntöjen visuaalinen analyysi

Ongelmana assosiaatiosäännöissä tosielämän ympäristöissä on se, että data muuttuu ajan kuluessa. Aikaisemmin louhitut säännöt eivät välttämättä ole päteviä tulevaisuudessa. Zhao ja Liu [69] esittelivät vuonna 2001 tekniikan assosiaatiosääntöjen käyttäytymisen tutkimiseen visuaalisesti. Ennen kuin löydettyyn sääntöön voidaan luottaa ja sääntöä voidaan käyttää, halutaan tietää kuinka sääntö käyttäytyy ajan kuluessa. Tekniikalla saadaan helposti esille säännön käyttäytyminen ja samalla parempi ymmärrys kohdealueesta.

Tekniikan ideana on jakaa data aikajaksoihin ja louhia näistä aikajaksoista assosiaatiosäännöt. Tällöin jokaiselle säännölle saadaan tuki- ja luottamusluvut jokaisesta aikajaksosta. Kun nämä arvot visualisoidaan saadaan esille sääntöjen käyttäytymishis-

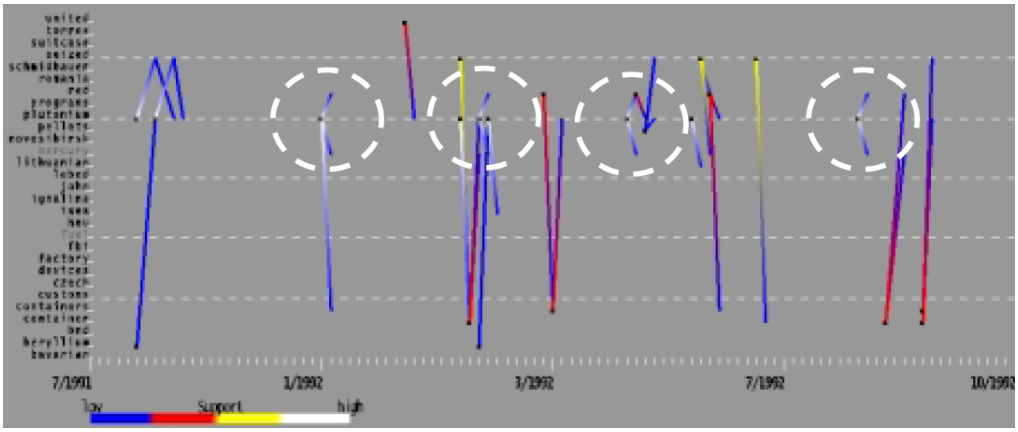


Kuva 6.6: Sääntöjen 1, 2 ja 3 luottamuksien muutokset ajan funktiona [69, sivu 2].

toria. Visualisoinnissa aika asetetaan x-akselille ja sääntöjen tuki- tai luottamusluvut y-akselille. Tällöin kuvaajasta nähdään sääntöjen suuntauksia ajan kuluessa. Edelleen visualisointia voidaan tehostaa tekemällä kyselyjä säännöistä, valitsemalla sääntöjen alijoukkoja tai klusteroimalla samoin käyttäytyviä sääntöjä.

6.4.3 Sekvenssihahmojen visualisointi

Yksinkertaisin tapa visualisoida sekvenssihahmoja on suunnattuihin graafeihin perustuva visualisointi. Tässä jokainen tapahtuma on yksi solmu graafissa ja sekvenssit kuvataan nuolilla näiden solmujen välillä. Erilaisille korostuksilla voidaan visualisoida esimerkiksi sekvenssihahmon tukilukuja. Huonona puolena tässä on se, että hahmojen valisiä suhteita ei nähdä selvästi. Lisäksi graafista ei voida sanoa milloin sekvenssit esiintyvät sekvenssitietokannassa. Parempi sekvenssihahmojen visualisointi on Wong et al. [65] esittämä visualisointi, kuva 6.7, jossa vasemmalla on listattu tapahtumat sekvenssitietokannassa. Värilliset viivat kuvaavat tapahtumien välisiä sekvenssejä. Hahmon alkupiste on merkitty mustalla pisteellä. Neljällä perusvärillä merkitään sekvenssihahmon ja sen alisekvenssien tukilukuja. Neljä valkoisella katkoviivalla olevaa ympyrää esittävät milloin sama sekvenssihahmo on esiintynyt kyseisellä aikavälillä. Saman



Kuva 6.7: Sekvenssihahmojen visualisointi [65, sivu 6]. Pak Chung Wong on antanut luvan kuvan käyttöön.

kaltaista tekniikkaa käyttämällä voidaan tutkia myös assosiaatiosääntöjä. Esimerkiksi voidaan visualisoida alkiojoukot aikajanalla ja korostamalla ne alkiojoukot, jotka ovat assosiaatiosääntöjen taustalla. Tällöin saadaan kuva siitä milloin kunkin assosiaatiosääntöön liittymät tapahtumat ovat esiintyneet.

7 Toteutus

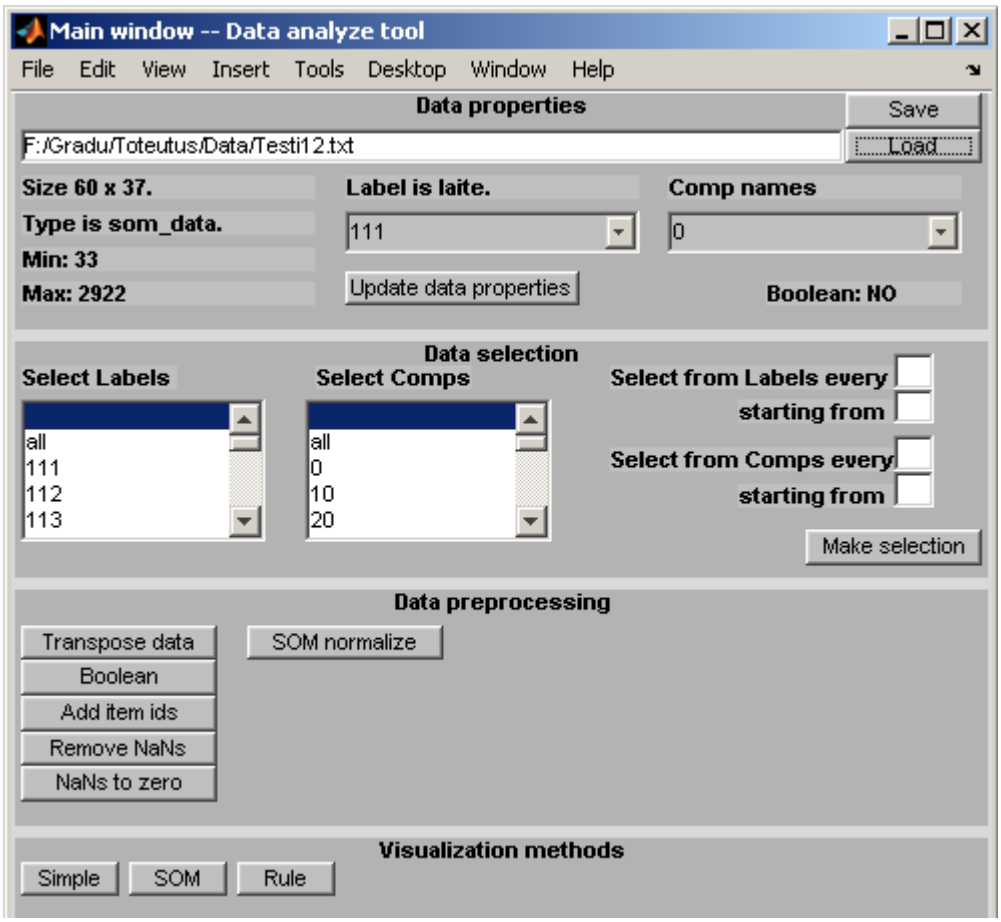
Tutkimuksen ohella toteutettiin prototyyppi tiedonlouhintatyökalusta. Luvussa 7.1 esitellään yleisesti käytettyjä algoritmeja ja ohjelmistoja sekä yleisesti louhintaprosessi. Luvuissa 7.2 ja 7.3 esitellään toteutetun prototyypin käyttöliittymään ja Matlab-ohjelmiston perusvisualisointeja. Luvussa 7.4 esitellään kuinka klusterointia käytettiin osana tietämyksen muodostamista. Luvuissa 7.5 ja 7.6 kerrotaan assosiaatiosääntöjen ja sekvenssihahmojen louhinnan totetuksesta.

7.1 Teknologia ja louhinnan kulku

Toteutus tehtiin pääosin Matlab-ohjelmistolla. Matlab [56] on The MathWorks Inc -yhtiön tuottama kaupallinen, matemaattiseen laskentaan ja visualisointiin tarkoitettu laskenta- ja ohjelmointiympäristö. Matlab on saavuttanut suosiota erityisesti tekniikan alalla niin tutkimuksessa kuin opetuksessakin.

Matlabia voidaan käyttää komento-ohjattuna, mutta komentoja voidaan kirjoittaa myös komentotiedostoiksi ja funktioiksi. Lisäksi Matlabissa on ohjelmoinnin kannalta keskeisimmät rakenteet myös oliosuuntautuneen ohjelmoinnin tarpeisiin. Tietorakenteet eivät vaadi määrittelyjä tai muistivarauksia vaan niiden rakennetta ja kokoa voidaan muutella vapaasti. Erityispiirteenä kielessä on matriisi- ja vektorikeskeisyys ja runsas valikoima matemaattisia funktioita. Usein silmukoiden käytön voidaan laskennassa välttää muuttamalla operaatiot matriisilaskuiksi.

Perusvisualisointiominaisuudet ovat Matlabissa hyvät ja näitä käytettiin analyysin alkuvaiheissa dataan tutustuttaessa. Klusteroinnissa ja datan visualisoinnissa käytettiin SOM Toolbox -työkalua [5], jossa on valmiina erilaisia datan visualisointifunktioita. Louhinnassa seuraavaksi käytettiin itseorganisoituvaa karttaa ja sen visualisointeja paremman kokonaiskuvan saamiseksi moniulotteisesta datasta. Tämän jälkeen kartan klustereista valittiin muuttujat assosiaatioanalyysiin. Aineistot piti muuttaa aina kyseisten ohjelmien välillä ohjelman ymmärtämään muotoon. Havaintoaineiston muunteluun muodosta toiseen, jälkikäsittelyyn sekä asoosiaatiosääntöjen visualisointiin ohjelmoitiin ohjelmia C/C++ ja Perl -kielillä. Assosiaatiooiden louhinnassa käytettiin Apriori-algoritmia, jonka on ohjelmoinut Christian Borgelt [11]. Sekvenssihahmojen



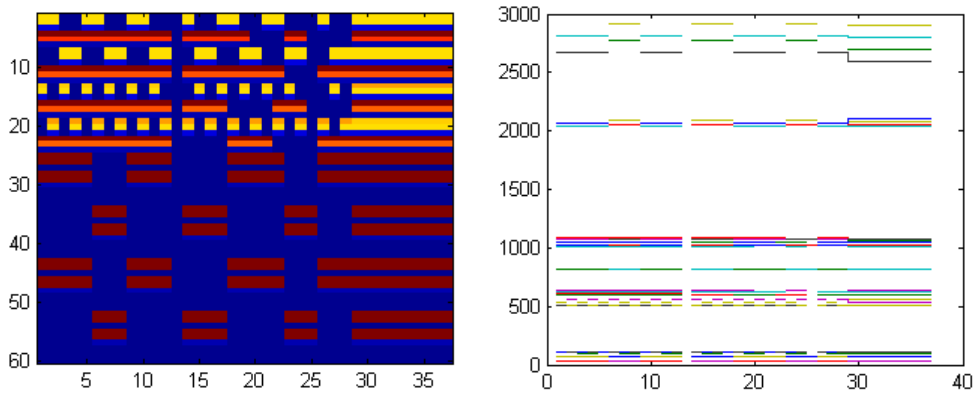
Kuva 7.1: Käyttöliittymä.

louhinnassa käytettiin PrefixSpan-algoritmia, joka on osa Illimine-ohjelmistoa [18].

7.2 Käyttöliittymä

Työkalun käyttöliittymä (kuva 7.1) ohjelmoitiin Matlab-ohjelmointiympäristössä. Ohjelmistossa on myös työkaluja graafisten käyttöliittymien toteuttamiseksi, joten niiden ohjelmointi oli kohtuullisen helppoa.

Käyttöliittymän perustoimintoja ovat havaintoaineiston lataaminen ja tallennus. Ladattaessa havaintoaineiston täytyy olla SOM Toolbox:n hyväksymässä muodossa (tarkemmin luvussa 7.4.1). Tallennusta tehdään eri muotoihin, koska työkalussa käytetyt algoritmit vaativat aineiston olevan omassa formaatissaan. Käsiteltävänä olevan datan ominaisuudet näkyvät käyttöliittymän yläosassa. Havaintoaineiston ominaisuuk-



Kuva 7.2: Yksinkertaisia visualisointeja: oikealla matriisikuva ja vasemmalla porrasfunktio.

sista on visualisoitu mm. havaintoaineiston koko, minimi- ja maksimiarvot sekä muuttujan (label) ja mittauksen (comp) tunnisteet. Havaintoaineiston valinta voidaan tehdä erikseen sarakkeiden ja rivien kesken, joko valitsemalla yksitellen halutut muuttujat tai alkaen määritellystä muuttujasta tietyin välein.

Esikäsittelyfunktioita on tehty useampia. Datan esikäsittelyllä pyritään saamaan data mahdollisimman edulliseen muotoon tiedonlouhintatyökaluja varten. Tällöin informaatio datasta on helpommin luettavissa. Havaintoaineisto voidaan mm. kääntää, muuttaa totuusarvoiseen muotoon ja normalisoida.

Käyttöliittymän visualisointiosaan on koottu Matlab ohjelmiston valmiita visualisointifunktioita sekä itseorganisoituvan kartan ja assosiaatiosääntöjen visualisointeja.

7.3 Yksinkertaiset visualisoinnit

Yksinkertaisilla visualisointitekniikoilla saadaan nopeasti kuva havaintoaineistosta. Esikäsittelemällä dataa saadaan lisäksi erilaisia näkymiä ja tuotua esille datasta eri asioita. Parhaimmiksi Matlab ohjelmiston perusvisualisoinneista todettiin matriisikuva ja porrasfunktio.

Matriisikuvassa 7.2 vaaka-akselilla on aika ja pystyakselilla on muuttujat. Muuttujien arvot on visualisoitu värein. Tumman sinisestä – tumman punaiseen (0 – maksimi). Porrasfunktiossa 7.2 vaaka-akselilla on aika ja pystyakselilla on muuttujan arvo. Eri mittauspisteet on visualisoitu eri väreillä.

Näistä yksinkertaisista kuvista saadaan jo tietoa havaintoaineiston taustalla olevas-

ta toiminnasta. Esimerkiksi nähdään mitkä muuttujat ovat saaneet saman kokoluokan arvoja, millaisella jaksolla muuttujista on saatu tuloksia ja milloin kaikkien muuttujien arvot ovat muuttuneet ja miten. Lisäksi porrasfunktiosta nähdään selvästi mistä muuttujista on saatu saman kokoluokan tuloksia. Hyviä puolia näissä on, että jokainen havainto visualisoidaan. Tällöin saadaan tarkka kuva havaintoaineistosta. Huonona puolena on se, että yhdellä kertaa ei voida visualisoida suuria datamääriä. Jos visualisoitavat datamäärät kasvavat suuriksi niin silloin yhden muuttujan havaintoja on hankala erottaa toisten muuttujien visualisoinneista.

7.4 Klusterointi

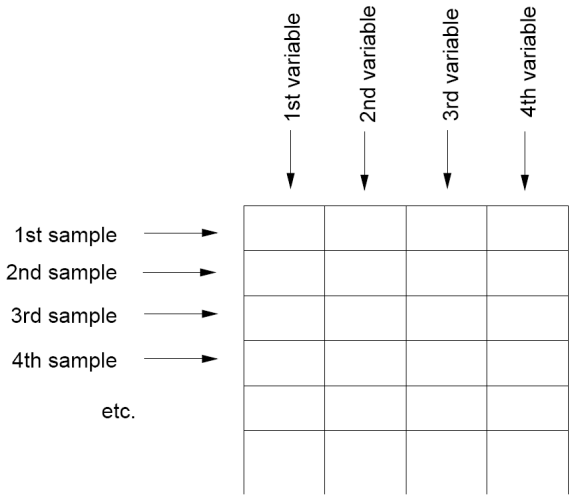
Klusterointiin valittiin Teuvo Kohosen itseorganisoituvaan karttaan [38] pohjautuva SOM Toolbox [5]. Luvussa 7.4.1 esitellään datan syöteformaatti, jota tiedonlouhintatyökalun prototyyppi ymmärtää. Luvussa 7.4.2 esitellään itseorganisoituvan kartan tuloksia.

7.4.1 Datan muoto

Data, jota käsitellään SOM ToolBox-ohjelmistossa on ns. taulukkomuotoista dataa (kuva 7.3). Datassa jokainen rivi taulukossa on yksi datanäyte ja rivillä olevat alkiot ovat datajoukon muuttujia tai komponentteja. Muuttujat voivat olla esimerkiksi objektin ominaisuuksia tai joukko mittauksia mitattuna tiettynä ajanhetkenä, joten jokainen sarake taulukossa sisältää kaikki yhden muuttujat arvot. Osa arvoista saa puuttua, mutta suurin osa arvoista pitää olla olemassa. Puuttuvat arvot korvattiin muuttujien arvojen keskiarvoilla. Dataan voidaan listätä myös metadataa. Esimerkiksi jokaiseen näytteeseen voidaan liittää tekstimuotoinen tunniste [63, sivu 12].

7.4.2 Louhinta itseorganisoituvalla kartalla

SOM Toolbox on Matlabissa käytettävä työkalupakki, joka sisältää karttoihin liittyvät perustietorakenteet, datan tyypillisimmät esikäsittelyrutiinit, kartan alustus- ja opetusrutiinit, visualisointimenetelmiä, tärkeimmät virhemitat ja graafisen käyttöliittymän opetuksen ja visualisoinnin helpottamiseksi. SOM Toolbox on toteutettu Matlabin komentokielellä, ja se on pyritty rakentamaan modulaarisesti, jotta käyttäjä voisi helposti rakentaa omia kokonaisuuksiaan. Itseorganisoituvan kartan avulla pyrittiin saamaan kokonaiskuva muuttujien toiminnasta sekä klusteroimaan samalla tavoin toi-



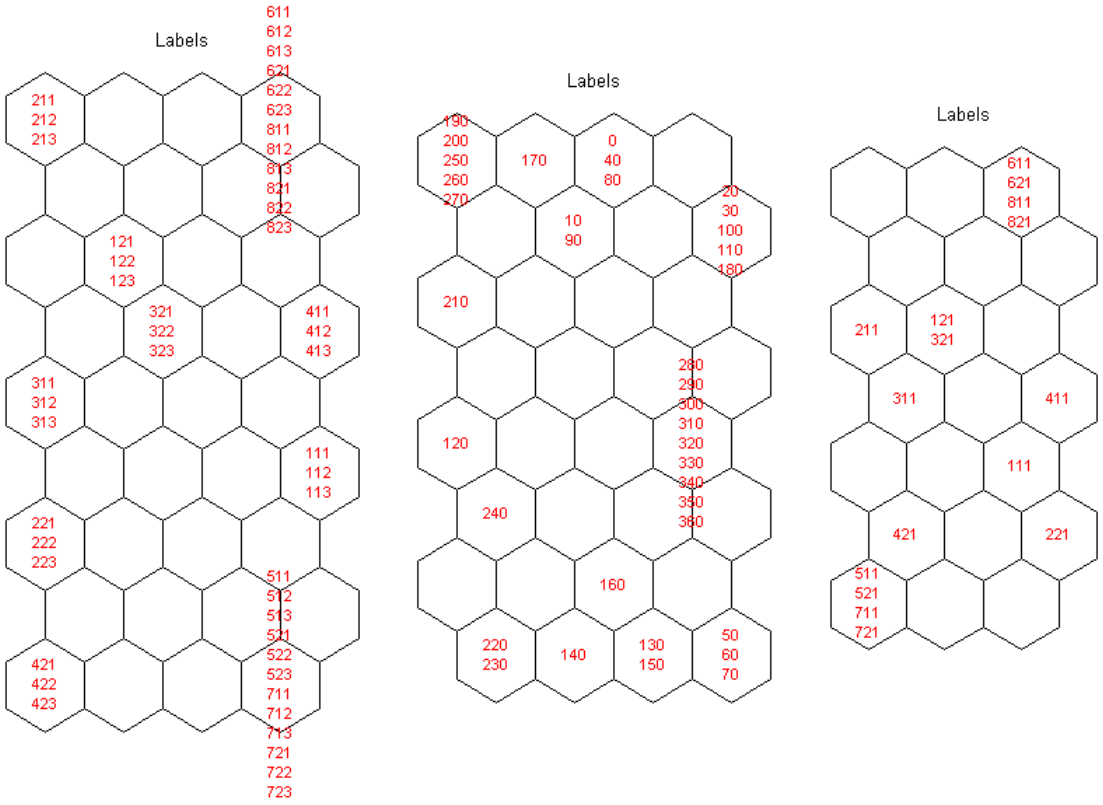
Kuva 7.3: Talukkomuotoinen data. Taulukossa voi olla miten paljon tahansa näytteitä, mutta kaikilla näytteillä on määrätty pituus ja näytteet sisältävät samat muuttujat. Juha Vesanto on antanut luvan kuvan käyttöön.

mivat muuttujat.

Kuvassa 7.4 vasemmalla ja oikealla on klusteroitu muutamia muuttujia. Tästä nähdään mitkä muuttujat ovat saaneet samanlaisia arvoja valitussa aikajaksossa. Kuvassa 7.4 keskellä datan tarkastelukulma on muutettu ja klusteroitu mittaustulokset päivittäin. Kuvasta voidaan tutkia milloin muuttujien yhteistoiminta on ollut samanlaista.

Hyväksi visualisoinniksi todettiin myös komponenttitasot (liitteet D ja E). Vertailemalla tasoja voidaan havaita visuaalisesti myös osittain korreloivat muuttujat. Esimerkiksi liitteen E kuvasta nähdään, että päivien 280–360 välisen ajan muuttujat ovat saaneet samanlaisia arvoja. Lisäksi nähdään, että päivien 0–30 ja 80–110 välisinä aikoina muuttujien arvot ovat muuttuneet samalla tavoin. Päivän 120 kohdalta ei ole saatu mittaustuloksia.

Itseorganisoituvalla kartalla saadaan hyvä yleiskuva datan taustalla olevasta ilmiöstä visualisoimalla data komponettitasoina. Tällaisen yleiskuvan muodostaminen OLAP yhteenvetotiedoista on hankalaa perinteisillä visualisointitekniikoilla. Lisäksi saadaan klusteroitua samankaltaiset muuttujat ja visualisoitua ne selkeästi. Verrattuna yksinkertaisiin visualisointitekniikoihin itseorganisoituvalla kartalla saadaan visualisoitua suuria määriä moniulotteista dataa, toisin kuin yksinkertaisilla visualisoinneilla.



Kuva 7.4: Muuttujien klusteroinnin visualisointeja.

Data	Alkioita	Alkioita säännössä	Tuki	Luottamus	Sääntöjä
1	96 kpl	1/5 kpl (min/max)	10%	80%	27 475 092 kpl
2	32 kpl	1/5 kpl (min/max)	10%	80%	84064 kpl
3	18 kpl	1/5 kpl (min/max)	10%	80%	2902 kpl
2	32 kpl	1/32 kpl (min/max)	10%	80%	1241871 kpl
3	18 kpl	1/18 kpl (min/max)	10%	80%	4680 kpl
2	32 kpl	1/5 kpl (min/max)	50%	80%	781 kpl
3	18 kpl	1/5 kpl (min/max)	50%	80%	67 kpl

Taulukko 7.1: Assosiaatiosääntöjen määriä erilaisilla alkiojoukoilla.

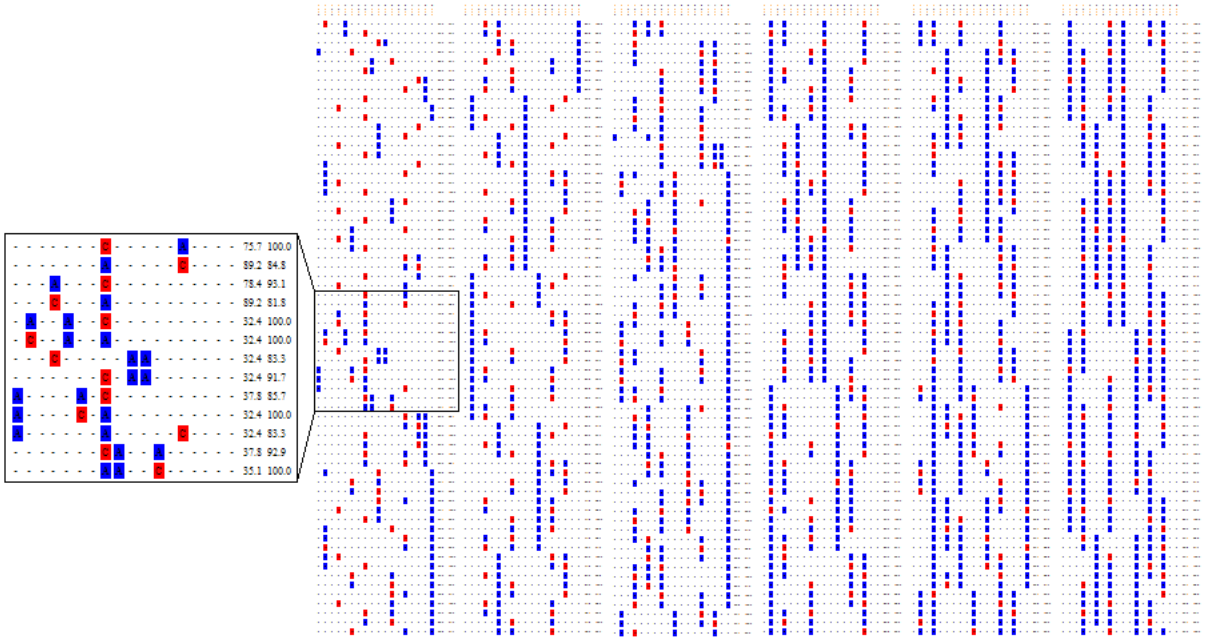
7.5 Assosiaatioiden louhinta

Assosiaatioiden louhinnassa käytettiin apriori-algoritmia. Algoritmi osaa muodostaa toistuvat alkiojoukot, suljetut ja maksimaaliset alkiojoukot sekä assosiaatiosäännöt. Lisäksi algoritmin toimintaa voidaan muuttaa monilla eri parametreilla. Ohjelman parametrit listattu liitteessä B.

Assosiaatioiden louhintaa varten havaintoaineisto muutettiin binaariseen (1 = mittaustulos saatu, 0 = ei mittaustulosta) muotoon. Muuttujat yksilöitiin tunnuksilla ja tunnusten perään merkittiin 1/0 riippuen siitä oliko kyseisellä ajanhetkellä muuttuja saanut arvoa vai ei. Tarkasteluun ei voitu ottaa ainoastaan niitä mittaustuloksia, joista arvo oli saatu vaan myös me mittaustulokset, joista arvoa ei saatu otettiin mukaan. Tämä siksi, että muuttujilla on suhteita toisiinsa siten, että kun toinen saa arvon niin toinen ei voi saada arvoa. Tästä johtuen jokaiseen tapahtumaan tuli sama määrä alkioita. Ongelmaksi assosiaatioiden louhinnassa muodostui assosiaatiosääntöjen valtava määrä. Algoritmin oletusarvoilla esikäsittelemättömästä havaintoaineistosta muodostui yli 80 miljoonaa (80 615 025) assosiaatiosääntöä. Taulukossa 7.1 on kuvattu assosiaatioiden määriä eri parametreilla.

Assosiaatiosääntöjen vähentämiseksi havaintoaineistosta poistettiin muuttujat, joista ei ollut kertaakaan saatu mittaustulosta (taulukko 7.1 data 1) ja jokaisesta mittauspisteestä valittiin vain yksi muuttuja (taulukko 7.1 data 2). Lisäksi nämä muuttujat klusteroitiin itseorganisoituvalla kartalla (kuva 7.4) ja jokaisesta klusterista valittiin ainoastaan yksi muuttuja assosiaatioanalyysiin (taulukko 7.1 data 3). Tällöin oletusarvoilla havaintoaineistosta muodostui vain 2902 assosiaatiosääntöä. Tämä määrä sääntöjä voidaan jo visualisoida järkevästi.

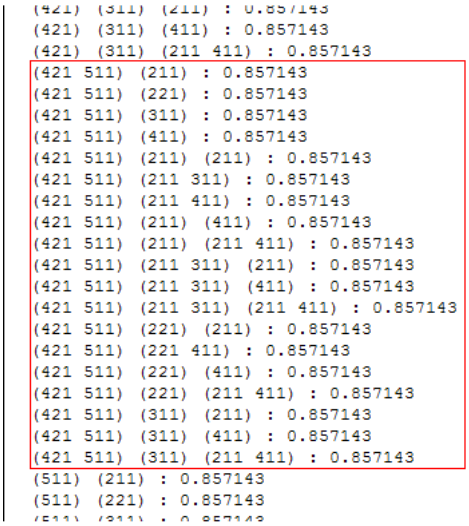
Assosiaatiosääntöjen visualisointiin toteutettiin kuvan 6.4 kaltainen visualisointi



Kuva 7.5: Assosiaatiosääntöjen visualisointi. Sinisellä edeltävänä olevat muuttujat ja punaisella seurauksena oleva muuttuja. Ylhäällä on muuttujien tunnukset.

(kuva 7.5). Poikkeuksena tässä on se, että matriisi on kaksiulotteinen. Kuvassa sinisellä merkityt muuttujat ovat edeltäjiä ja punaisella merkitty muuttuja seurauksena oleva. Muuttujien tunnukset on visualisoitu taulukon yläreunassa ja tuki sekä luottamus assosiaatiosäännön oikealla puolella. Hyvänä puolena tässä visualisoinnissa on se, että visualisoitavaksi voidaan lisätä haluttu määrä kiinnostavuuden mittareita. Lisäksi kiinnostavuuslukuja voidaan korostaa visualisoimalla ne eri värisinä ja kokoisina kuvakkeina. Kolmiulotteisessa matriisissa 6.4 ei näin monen metriikan visualisoiminen onnistuisi, koska tällöin takana olevat metriikat peittyisivät. Huono puoli kaksiulotteisessa visualisoinnissa on se, että kolmiulotteisessa matriisissa on helpompi nähdä metriikoiden arvot, koska ne on kuvattu eri pituisilla pylväillä.

Assosiaatiosäännöillä löydettiin kaikki muuttujien väliset riippuvuudet. Olkoon a, b ja c muuttujia. Assosiaatiosäännöillä löydettiin näiden muuttujien välillä seuraavan kaltaisia riippuvuuksia. Muuttujat a ja b esiintyvät 70.3 prosentin tuella ja $96,2$ prosentin luottamuksella samassa tapahtumassa. Toisaalta muuttujat a ja c eivät esiinny 56.8 prosentin tuella ja 100 prosentin luottamuksella samassa tapahtumassa. Tämän kaltaisten riippuvuuksien löytäminen OLAP-tekniikoilla on hyvin haasteellista.



Kuva 7.6: Sekvenssihahmojen louhinnan tuloksia.

7.6 Sekvenssien louhinta

Sekvenssihahmoja louhittiin PrefixSpan-algoritmilla, joka on osa Illimine-ohjelmistoa. Data muutettiin toteutuksen vaatimaan muotoon. Datassa tapahtumat erotellaan luvulla -1 ja sekvenssit luvulla -2. Lisäksi data pitää olla binaarisessa muodossa.

Sekvenssihahmoille ei toteutettu mitään erillistä visualisointia vaan louhinnan tuloksia tutkittiin ainoastaan tekstimuotoisena. Kuvassa 7.6 on louhituista sekvenssihahmoista valittu ne hahmot, jotka alkavat tapahtumalla, jossa alkiot 421 ja 511 esiintyvät yhdessä. Sekvenssistä $\langle(421\ 511)(311)(211\ 411)\rangle$ nähdään, että tapahtumaa (421 511) seuraa tapahtuma (311) ja edelleen tapahtuma (211 411). Sekvenssihahmon tukiluku on yli 85%.

Sekvenssihahmojen louhinnan hyötynä oli se, että näin monimutkaisten riippuvuussuhteiden etsiminen on käytännössä mahdotonta. Esimerkiksi edellä esitetyn sekvenssihahmon olemassa oloa ei löydetty perinteisin menetelmin eikä sitä osattu ennustaa. Huonona puolena sekvenssihahmoissa on hahmojen suuri määrä. Tämän takia sekvenssihahmojen analysointityökalun pitää sisältää hahmojen hakumahdollisuus.

8 Yhteenveto

Tiedonlouhintamenetelmät valitaan yleensä sen perusteella mitä datasta halutaan löytää. Tiedonlouhinta on *"suurten, tiettyä tarkoitusta varten kerättyjen tietojoukkojen analyysia, jonka tavoitteena on löytää suhteita ja tiivistää dataa uusilla tavoilla, jotka ovat sekä ymmärrettäviä että käyttökelpoisia"* [45, sivu 9].

Tutkimuksen lähtökohtana oli TietoEnator Oyj:ssä esille tullut tarve uusille tiedon analysointitekniikoille tietokantojen koon kasvaessa. Työssä tutkittiin tiedonlouhinnan menetelmiä perinteisten OLAP-tekniikoiden rinnalle. OLAP tarjoaa käyttäjälle eritasoihin koostettua yhteenvetotietoa. OLAP ei ole mikään yksittäinen ohjelmisto tai menetelmä, vaan se kattaa suuren joukon menetelmiä datan talletuksesta raportointiin ja visualisointiin [37, sivu 1]. Tutkimuksessa käytettiin TietoEnator Oyj:ssä simuloitua dataa, joka pohjautuu todelliseen dataan. Datasta pyrittiin löytämään helposti muutokset ja riippuvuudet sekä visualisoimaan löydetyt tulokset selkeästi käyttäjälle.

Näiden ominaisuuksien löytämiseksi tutkimuksen tiedonlouhintatekniikoiksi valittiin assosiaatioanalyysi ja sen laajennuksena sekvenssihahmot (luku 5) sekä klusterointi itseorganisoituvalla kartalla (luku 4). Itseorganisoituva kartta on neuraalilaskentamenetelmä, jolla voidaan kuvata alkuperäinen syötedata-avaruus $\mathbb{R}^n$ alempiulotteiseen avaruuteen niin, että alunperin lähellä toisiaan olevat näytteet ovat lähellä toisiaan myös alempiulotteisessa kuvauksessa eli kartalla [38]. Assosiaatiosääntöjen louhimisen tavoitteena on löytää kiinnostavia korrelaatioita ja säännönmukaisuuksia suuresta datajoukosta. Sekvenssihahmoissa etsitään alkioita, jotka edeltävät toisia alkioita. Sekvenssihahmo on tutkimus alkioiden järjestyksistä kun taas assosiaatiosääntö on tutkimus alkioiden yhteenkuulumisesta [65, sivu 1].

Tutkimuksen perusteella voidaan sanoa, että assosiaatiosäännöt ja sekvenssihahmot soveltuvat hyvin riippuvuuksien ja riippuvuusketjujen louhintaan. Nämä tekniikat antavat lisäksi ymmärrettävät tulokset. Ongelmana kuitenkin on generoituvien sääntöjen valtava määrä. Valtavasta sääntöjen joukosta voidaan valita todennäköisimmin käyttäjää kiinnostavat säännöt erilaisilla tekniikoilla. Itseorganisoituvalla kartalla ja sen visualisoinneilla saadaan selville muutakin kuin datan sisältämät klusterit. Visualisoitaessa data komponettitasoihin voidaan nähdään kuinka data on muuttunut ajan kuluessa sekä huomata osittaisia assosiaatioita. Tutkimuksen aikana huomattiin myös,

että hyvä tiedonlouhintatyökalu koostuu monista erilaisista datan analysointi- ja visualisointitekniikoista. Yhdestä visualisointitavasta voidaan päätellä joitakin asioita, mutta useiden visualisointitapojen yhtäaikainen esittäminen antaa helpommin ja nopeammin ymmärryksen tutkittavasta ilmiöstä. Lisäksi tietämyksen muodostamisprosessi on lähes aina iteratiivinen, joten tiedonlouhintatyökalun tulee tukea hyvin kaikkia tiedonlouhinnan vaiheita tekien louhinnan mahdollisimman helpoksi käyttäjälle.

Jatkokehityksen kohteena on interaktiivinen tiedonlouhintatyökalu OLAP-tekniikoiden tueksi. Työkalussa data visualisoidaan eri tasoisina näkyminä ja eri tekniikoilla, jotta sopivan visualisointitekniikan valitseminen olisi mahdollista. Interaktiivisuus tulee siitä, että visualisointitekniikat muuttuvat suhteessa toisiinsa ja yhdessä visualisointi-ikkunassa tehdyt muutokset vaikuttavat kaikkiin visualisointi-ikkunoihin. Lisäksi tietämyksen muodostamista helpotetaan louhimalla assosiaatiosääntöjä ja sekvenssihahmoja sekä visualisoimalla ne saman aikaisesti havaintoaineiston kanssa.

9 Viitteet

[1] Agrawal R. ja Srikant R., *Fast Algorithms for Mining Association Rules*. In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994. <URL: `http://citeseer.ist.psu.edu/agrawal94fast.html`>

[2] Agrawal R., Imielinski T. ja Swami A., *Mining association rules between sets of items in large databases*. In Proc. of the ACM SIGMOD Conference on Management of Data, pages 207–216, Washington, D.C., May 1993. <URL: `http://citeseer.ist.psu.edu/agrawal93mining.html`>

[3] Agrawal R. ja Srikant R., *Mining sequential patterns*, Eleventh International Conference on Data Engineering, IEEE Computer Society Press, Taipei, Taiwan, 1995, sivut 3–14, <URL: `http://citeseer.ist.psu.edu/agrawal95mining.html`>.

[4] Ahlgren M., *Tiedon louhinta myynnin, markkinoinnin ja asiakkuuksien hallinnassa*, Tilastotieteen Pro gradu -tutkielma, Jyväskylän yliopisto, 2001.

[5] Alhoniemi E., Himberg J., Parhankangas J., ja Vesanto J., *SOM ToolBox*, Laboratory of Information and Computer Science in the Helsinki University of Technology, <URL: `http://www.cis.hut.fi/projects/somtoolbox/`>.

[6] Aris A., Shneiderman B., Plaisant C., Shmueli G., Jank W., Buono P. ja Khella A., *TimeSearcher - Visual Exploration of Time-Series Data*, Human-Computer Interaction Lab, University of Maryland, <URL: `http://www.cs.umd.edu/hcil/timesearcher/`>.

[7] Ayan N. F., Tansel A. U. ja Arkun E., *An efficient algorithm to update large itemsets with early pruning*, KDD-99, San Diego, California, United States, 1999, <URL: `http://www.cs.umd.edu/users/nfa/Publications/ayan-kdd99-uwep.pdf`>.

[8] Bayardo R., Agrawal R. ja Gunopulos D., *Constraint-based rule mining in large, dense databases*, To appear in ICDE-99, 1999, <URL: `http://citeseer.ist.psu.edu/bayardo99constraintbased.html`>.

[9] Bayardo R., *Efficiently mining long patterns from databases* International Conference on Management of Data, Proceedings of the 1998 ACM SIGMOD international conference on Management of data, Seattle, Washington, United States sivut 85 – 93, 1998, <URL: http://www.almaden.ibm.com/software/projects/hdb/Publications/papers/sigmod98_max.pdf>.

[10] Borgelt C., *Apriori-algorithm PseudoCode*, <URL: http://fuzzy.cs.uni-magdeburg.de/studium/ida/txt/apriori.pdf>

[11] Borgelt C., *Apriori - Association Rule Induction / Frequent Item Set Mining - version 4.27*, 2005, <URL: http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html>.

[12] Burdick D., Calimlim M., Flannick J., Gehrke J. ja Yiu T., *MAFIA: A Performance Study of Mining Maximal Frequent Itemsets*, Workshop on Frequent Itemset Mining Implementations (FIMI'03). Melbourne, Florida, 2003, <URL: http://citeseer.ist.psu.edu/645413.html>.

[13] Bustos B., Keim D. A. ja Panse C., *Pattern Visualization*, Computer Science Institute, Konstanz, 2003, <URL: http://dke.cti.gr/panda/tasks/deliverables/DLV-2-3.pdf>.

[14] Carlis J. ja Konstan J., *Interactive visualization of serial periodic data*, Proceedings of the 11th annual ACM symposium on User interface software and technology, San Francisco, California, United States, 1998, <URL: http://citeseer.ist.psu.edu/carlis98interactive.html>.

[15] Chen H., Chung W., Xu J. J., Wang G., Qin Y. ja Chau M., *Crime data mining: A general framework and some examples*, Computer, 37(4), sivut 50–56, 2004, <URL: http://citeseer.ist.psu.edu/chen04crime.html>.

[16] Cohen E., Datar M., Fujiwara S., Gionis A., Indyk P., Motwani R., Ullman J. D. ja Yang C., *Finding Interesting Associations without Support Pruning*, Technical Report, Computer Science Department, Stanford University, 1999, <URL: http://citeseer.ist.psu.edu/article/cohen99finding.html>.

[17] Connolly T. ja Begg C., *Database Systems – A Practical Approach to Design, Implementation, and Management. Third Edition*, Addison-Wesley, 2002, ISBN: 0201708574.

[18] Data Mining Research Group, *IlliMine system package Release 1.0.0*, DAIS (Data And Information Systems) Research Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005, <URL: http://illimine.cs.uiuc.edu/>.

[19] Fukuda T., Morimoto Y., Morishita S. ja Tokuyama T., *Mining optimized association rules for numeric attributes*, Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, Montreal, Quebec, Canada, 1996, <URL: http://www.gi.k.u-tokyo.ac.jp/ moris/paper/pods96.pdf>.

[20] Gouda K. ja Zaki M. J., *Efficiently Mining Maximal Frequent Itemsets*, Proc. of the IEEE Int. Conference on Data Mining, San Jose, 2001. <URL: http://citeseer.ist.psu.edu/gouda01efficiently.html>.

[21] Han J. ja Kamber M., *Data Mining: Concepts and Techniques 2001*, Morgan Kaufmann, 2001, ISBN: 1-55860-489-8.

[22] Han J. ja Kamber M., *Data Mining: Concepts and Techniques Second edition, Draft, Chapter 5 Mining Frequent Patterns, Associations, and Correlations*, 2005. <URL: http://www.cs.uiuc.edu/class/fa05/cs412/chaps/5.pdf>

[23] Han J. ja Kamber M., *Data Mining: Concepts and Techniques Second edition, Draft, Chapter 3 Data Warehouse and OLAP Technology: An Overview*, 2005. <URL: http://www.cs.uiuc.edu/class/fa05/cs412/chaps/3.pdf>

[24] Han J., Pei J., Mortazavi-Asl B., Chen Q., Dayal U. ja Hsu M.-C., *FreeSpan: frequent pattern-projected sequential pattern mining*, Conference on Knowledge Discovery in Data, Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, Massachusetts, United States, Sivut 355 – 359, 2000 <URL: http://portal.acm.org/citation.cfm?doid=347090.347167>.

[25] Hidber C., *Online association rule mining*, Proceedings of the 1999 ACM SIGMOD international conference on Management of data, Philadelphia, Pennsylvania, United States, 1999, <URL: http://portal.acm.org/citation.cfm?id=304195>.

[26] Himberg J., *Itseorganisoituvaan karttaan perustuva työkalu ja sen soveltaminen puheludatan analyysiin.*, Diplomityö, Teknillinen korkeakoulu, 1997. <URL: http://www.cis.hut.fi/jhimberg/dippa/dippa.html.

[27] Hipp J., Myka A., Wirth R. ja Güntzer U., *A new algorithm for faster mining of generalized association rules*, In Proc. 2nd PKKD, 1998, <URL: http://citeseer.ist.psu.edu/hipp98new.html>.

[28] Hirvonen L., *Helppokäyttöisen OLAP-kyselykielen suunnittelu ja toteutus*, Pro gradu -tutkielma, Tampereen yliopisto, 2001. <URL: http://www.cs.uta.fi/research/theses/masters/Hirvonen_Lasse.pdf>

[29] Hofmann H., Siebes A. ja Wilhelm A., *Visualizing Association Rules with Interactive Mosaic Plots*, Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, Massachusetts, United States, 2000, <URL: http://portal.acm.org/citation.cfm?doid=347090.347133>.

[30] Huotari M-L., *Informaatiotutkimuksen perusteet - oppimateriaali*, Oulun yliopisto, Suomen kielen, informaatiotutkimuksen ja logopedian laitos, 2005, <URL: http://www.oulu.fi/informaatiotutkimus/perusteet/materiaali/perusteet-kalvot.pdf>.

[31] Jirka T., *Multidimensional Data Visualization*, State of the Art and Concept of Doctoral Thesis, University of West Bohemia, Department of Computer Science and Engineering, Pilsen, Czech Republic, 2003 <URL: http://www.kiv.zcu.cz/publications/2003/tr-2003-03.pdf>.

[32] Joshi M., Karypis G. ja Kumar V., *Universal formulation of sequential patterns*, Technical Report Under Preparation, Department of Computer Science, University of Minnesota, Minneapolis, 1999. <URL: http://citeseer.ist.psu.edu/article/joshi99universal.html>.

[33] Junno H., *Itseorganisoituvista kartoista ja niiden käytöstä pistehitsauksen laaduntarkkailuun*, Pro Gradu -tutkielma, Matematiikan koulutusohjelma, Oulu, 2003, <URL: http://www.ee.oulu.fi/research/isg/publications/pdf/masters_theses/Junno-2003.pdf>.

[34] Kanerva K., *Lähestymistapoja OLAP-kieliin*, Pro gradu -tutkielma, Tampereen yliopisto, 2003. <URL: http://www.cs.uta.fi/research/theses/masters/Kanerva_Kaarlo.pdf>

[35] Kaski S., Kangas J. ja Kohonen T., *Bibliography of Self-Organizing Map (SOM) Papers: 1981–1997*, Helsinki University of Technology, Neural Networks Research Centre, 1997, <URL: http://www.cse.ucsc.edu/NCS/VOL1/vol1_4.pdf>.

[36] Kaski S. ja Lagus K., *Comparing Self-Organizing Maps*, In: Artificial Neural Networks – ICANN'96, Eds. C. v. d. Malsburg, W. v. Seelen, J. C. Vorbruggen, B. Sendhoff, Springer, Berlin, Heidelberg, New York, 809-814, 1996, <URL: http://citeseer.csail.mit.edu/kaski96comparing.html>.

[37] Kiviniemi J., Wolski A., *OLAP-tekniikoiden käyttömahdollisuudet teollisuusprosessien analysoinnissa*, VTT Tietotekniikka, 1999, <URL: http://virtual.vtt.fi/inf/julkaisut/muut/1990s/ap99-olap.pdf>

[38] Kohonen T., *Self-Organizing Maps*, Springer Series in Information Sciences, 1995, ISBN 3-540-58600-8.

[39] Kuronen T., *Hajautettu dokumenttien hallinta: Johdatus tekstin ja dokumenttien käsittelyyn tietoverkoissa*, ISBN 951-42-4759-0, Oulun yliopiston kirjasto, Oulu, 1997, <URL: http://herkules.oulu.fi/isbn9514248635/html/k.html>.

[40] Lakshmanan R., Han J. ja Pang A, *Exploratory mining and pruning optimizations of constrained associations rules*, In Proc. 1998 ACMSIGMOD Int. Conf. Management of Data, pages 13–24, Seattle, Washington, June 1998, <URL: http://citeseer.ist.psu.edu/ng98exploratory.html>.

[41] Lent B., Swami A. ja Widom J., *Clustering association rules*, In Proc. 1997 Int. Conf. Data Engineering (ICDE'97), pages 220–231, Birmingham, England, April 1997, <URL: http://citeseer.ist.psu.edu/lent97clustering.html>.

[42] Mannila H., Toivonen H. ja Verkamo A. I., *Discovery of frequent episodes in event sequences*, Data Mining and Knowledge Discovery, 1997, <URL: http://citeseer.ist.psu.edu/mannila97discovery.html>.

[43] Mannila H. ja Toivonen H., *Knowledge Discovery in Databases: The Search for Frequent Patterns*, 2002. <URL: http://www.cis.hut.fi/Opinnot/T-61.5060/t122103-2003.pdf>

[44] Niiniluoto I., *Informaatio, tieto ja yhteiskunta. Filosofinen käsiteanalyysi*, Helsinki, 5. täydennetty painos, 1996. <URL: http://www.uta.fi/laitokset/tiedotus/opiskelu/P1_lukemisto/Niiniluoto2.pdf>.

[45] Nurminen M., *Tiedonlouhinta rakenteisista dokumenteista*. Pro gradu -tutkielma, Jyväskylän yliopisto, 2005, <URL: http://thesis.jyu.fi/05/URN_NBN_fi_jyu-200594.pdf>

[46] Pasquier N., Bastide Y., Taouil R. ja Lakhal L., *Discovering Frequent Closed Itemsets for Association Rules*, In 7th Intl. Conf. on Database Theory, January 1999, <URL: http://citeseer.ist.psu.edu/pasquier99discovering.html>.

[47] Pei J., Han J., ja Mao R., *CLOSET: An efficient algorithm for mining frequent closed itemsets*, ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000, sivut 21–30, <URL: http://citeseer.ist.psu.edu/pei00closet.html>.

[48] Pei J., Han J., Mortazavi-Asl B., Wang J., Pinto H., Chen Q., Dayal U., Hsu M., *Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach.*, IEEE Transactions on Knowledge and Data Engineering, 2004. <URL: http://citeseer.ist.psu.edu/655475.html.

[49] Rastogi R. ja Shim K., *Mining optimized support rules for numeric attributes*, In Proc. of the 15th Int'l Conf. on Data Engineering, Sydney, Australia, March 1999, IEEE Computer Society Press, <URL: http://citeseer.ist.psu.edu/67691.html>.

[50] Sachinopoulou A., *Multidimensional Visualization*, VTT Electronics, Espoo, 2001, <URL: http://www.vtt.fi/inf/pdf/tiedotteet/2001/T2114.pdf>.

[51] Srikant R. ja Agrawal R., *Mining Generalized Association Rules*, In Proc. of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland, September 1995, <URL: http://citeseer.ist.psu.edu/srikant95mining.html>.

[52] Srikant R. ja Agrawal R., *Mining quantitative association rules in large relational tables*, In Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada, June 1996, <URL: http://citeseer.ist.psu.edu/srikant96mining.html>.

[53] Srikant R. ja Agrawal R., *Mining Sequential Patterns: Generalizations And Performance Improvements*, Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT). Avignon, France, 1996, <URL: http://citeseer.ist.psu.edu/189.html>.

[54] Srikant R., Vu Q. ja Agrawal R., *Mining association rules with item constraints*, Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, 1997, <URL: http://citeseer.ist.psu.edu/761.html>.

[55] Tan P., Kumar V. ja Srivastava J., *Selecting the right interestingness measure for association patterns*, In Proceedings of the Eight A CM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 2002, <URL: http://citeseer.ist.psu.edu/tan02selecting.html.

[56] The MathWorks, Inc., *MATLAB® - The Language of Technical Computing*, <URL: http://www.mathworks.com/>

[57] Thomas S., Bodagala S., Alsabti K. ja Ranka S., *An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases*, In Proceedings of the 3rd International conference on Knowledge Discovery and Data Mining (KDD 97), New Port Beach, California, 1997 <URL: http://citeseer.ist.psu.edu/thomas97efficient.html>.

[58] Tsur D., Ullman J. D., Abitboul S., Clifton C., Motwani R. ja Nestorov S., *Query flocks: A generalization of association-rule mining*, In Proc. ACM-SIGMOD, 1998, <URL: http://citeseer.ist.psu.edu/tsur98query.html>

[59] Vasko K., *Tausta: Mitä on tiedon louhinta (data mining)?*, CSC, Selvitykset 2001, <URL: http://www.csc.fi/selvitykset2001/datamining/dm_kuvaus.phtml.fi>.

[60] Vasko K., *Tiedon louhinnasta helpotusta infoähkyyn*, CSC, Tietoyhteys 1/2001, s. 15–17. <URL: http://www.csc.fi/lehdet/tietoyhteys/TY1_2001.pdf>

[61] Vesanto J., *Data Mining Techniques Based on the Self-Organizing Map*, Pro gradu -tutkielma, Helsingin yliopisto, 1997. <URL: http://www.cis.hut.fi/projects/ide/publications/html/mastersJV97/.

[62] Vesanto J., *Data Exploration Process Based on the Self-Organizing Map*, Väitöskirja, Helsingin yliopisto, 2002. <URL: http://lib.tkk.fi/Diss/2002/isbn9512258978/.

[63] Vesanto J., Himberg J., Alhoniemi E. ja Parhankangas J., *SOM Toolbox for Matlab 5*, SOM Toolbox Team, Helsinki University of Technology, ISBN 951-22-4951-0, 2000, <URL: http://www.cis.hut.fi/projects/somtoolbox/package/papers/techrep.pdf>.

[64] Wong P. C., Whitney P. ja Thomas J., *Visualizing Association Rules for Text Mining*. Proceedings IEEE Information Visualization 99, San Francisco, CA, 1999, <URL: http://citeseer.ist.psu.edu/wong99visualizing.html>

[65] Wong P. C., Cowley W., Foote H., Jurrus E., ja Thomas J., *Visualizing Sequential Patterns for Text Mining*, Proceedings IEEE Information Visualization 2000, Salt Lake City, Utah, 2000, <URL: http://citeseer.ist.psu.edu/wong00visualizing.html>

[66] Yan X., Han J. ja Afshar R., *CloSpan: Mining Closed Sequential Patterns in Large Datasets*, Proc., 2003, <URL: http://citeseer.ist.psu.edu/yan03clospan.html>.

[67] Zaki M. J., Parthasarathy S., Ogihara M. ja Li W., *New algorithms for fast discovery of association rules*, Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), 1997, <URL: http://citeseer.ist.psu.edu/zaki97new.html>.

[68] Zaki M. J., *SPADE: An Efficient Algorithm for Mining Frequent Sequences*, In Proc. of Machine Learning Journal, special issue on Unsupervised Learning (Doug Fisher, ed.), Vol. 42 Nos. 1/2, pages 31-60, Jan/Feb 2001. <URL: http://citeseer.ist.psu.edu/zaki00spade.html>

[69] Zhao K., Liu B., *Visual Analysis of the Behavior of Discovered Rules*, Workshop Notes in ACM SIGKDD-2001 Workshop on Visual Data Mining, San Francisco, CA, 2001, <URL: http://www.cs.uic.edu/ liub/publications/kdd2001-wksp.pdf>.

A Kiinnostavuuden mittarit säännöissä

#	Mittari	Kaava
1	ϕ-coefficient	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_k P(A_j,B_k)-\max_j P(A_j-\max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(A,\overline{B})P(\overline{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\overline{AB})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{AB})+P(A,\overline{B})P(\overline{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\overline{AB})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{AB})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
7	Mutual Information (M)	$\frac{\sum_i \sum_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i), -\sum_j P(B_j)\log P(B_j))}$
8	J-Measure (J)	$\max(P(A,B)\log(\frac{P(B\|A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),$ $P(A,B)\log(\frac{P(A\|B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})}))$
9	Gini index (G)	$\max(P(A)[P(B \mid A)^2 + P(\overline{B} \mid A)^2]+$ $P(\overline{A})[P(B \mid \overline{A})^2 + P(\overline{B} \mid \overline{A})^2] - P(B)^2 - P(\overline{B})^2,$ $P(B)[P(A \mid B)^2 + P(\overline{A} \mid B)^2]+$ $P(\overline{B})[P(A \mid \overline{B})^2 + P(\overline{A} \mid \overline{B})^2] - P(A)^2 - P(\overline{A})^2)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B \mid A), P(A \mid B))$
12	Laplace (L)	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
13	Conviction (V)	$\max(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})})$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	Cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B) - P(A)P(B)$
17	Certainty factor (F)	$\max(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)})$
18	Added Value (AV)	$\max(P(B \mid A) - P(B), P(A \mid B) - P(A))$
19	Collective strenght (S)	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
20	Jaccard (ζ)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
21	Klosgen (K)	$\sqrt{P(A,B)}\max(P(A \mid B) - P(B), P(B \mid A) - P(A))$

B Apriori-algoritmi toteutuksen parametrit

```
                Find association rules with the apriori algorithm
                version 4.27 (2005.06.20) (c) 1996-2005 Christian Borgelt
usage:          apriori [options] infile outfile [appfile]
-t#             target type (default: association rules)
                (s: item sets, c: closed item sets, m: maximal item sets,
                r: association rules, h: association hyperedges)
-m#             minimal number of items per set/rule/hyperedge (default: 1)
-n#             maximal number of items per set/rule/hyperedge (default: 5)
-s#             minimal support of a set/rule/hyperedge (default: 10%)
-S#             maximal support of a set/rule/hyperedge (default: 100%)
-c#             minimal confidence of a rule/hyperedge (default: 80%)
-o              use original definition of the support of a rule (body & head)
-k#             item separator for output (default: " ")
-p#             output format for support/confidence (default: "%.1f")
-x              extended support output (print both rule support types)
-a              print absolute support (number of transactions)
-y              print lift value (confidence divided by prior)
-e#             additional evaluation measure (default: none)
-!              print a list of additional evaluation measures
-d#             minimal value of additional evaluation measure (default: 10%)
-v              print value of additional rule evaluation measure
-g              write output in scanable form (quote certain characters)
-l              do not load transactions into memory (work on input file)
-q#             sort items w.r.t. their frequency (default: 2)
                (1: ascending, -1: descending, 0: do not sort,
                2: ascending, -2: descending w.r.t. transaction size sum)
-u#             filter unused items from transactions (default: 0.1)
                (0: do not filter items w.r.t. usage in sets,
                <0: fraction of removed items for filtering,
                >0: take execution times ratio into account)
-h              do not organize transactions as a prefix tree
-j              use quicksort to sort the transactions (default: heapsort)
-z              minimize memory usage (default: maximize speed)
-i#             ignore records starting with a character in the given string
-b/f/r#         blank characters, field and record separators
                (default: " \t \r", " \t", "\n")
infile          file to read transactions from
outfile         file to write item sets/association rules/hyperedges to
appfile         file stating item appearances (optional)
```

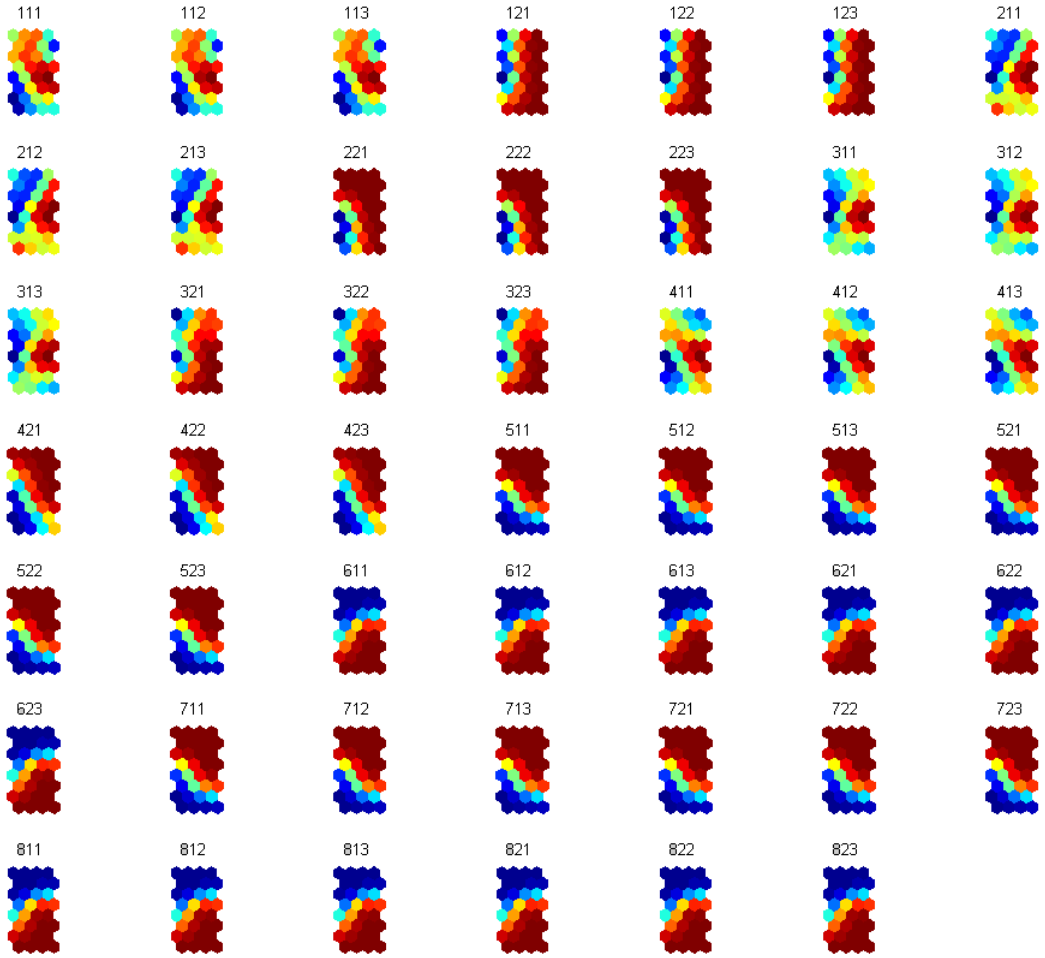
C Apriori-algoritmi pseudokoodina [10]

```
function apriori(I, T, s_min, c_min, k_max)          (* apriori algorithm for association rules *)
begin
    k := 1;                                          (* – find frequent item sets *)
    C_k := ⋃_{i∈I} {i};                              (* start with single element sets *)
    F_k := prune(C_k, T, s_min);                     (* and determine the frequent ones *)
    while F_k ≠ ∅ and k ≤ k_max do begin             (* while there are frequent item sets *)
        C_{k+1} := candidates(F_k);                  (* create item sets with one item more *)
        F_{k+1} := prune(C_{k+1}, T, s_min);         (* and determine the frequent ones *)
        k := k + 1;                                  (* increment the item counter *)
    end;
    R := ∅;                                          (* – generate association rules *)
    forall f ∈ ⋃_{j=2}^{k} F_j do begin              (* traverse the frequent item sets *)
        m := 1;                                      (* start with rule heads (consequents) *)
        H_m := ⋃_{i∈f} {i};                          (* that contain only one item *)
        repeat                                       (* traverse rule heads of increasing size *)
            forall h ∈ H_m do                        (* traverse the possible rule heads *)
                if s(f)/s(f−h) ≥ c_min               (* if the confidence of the rule *)
                then R := R ∪ {[(f − h) → h]};       (* is high enough, add it to the result, *)
                else H_m := H_m − {h};               (* otherwise discard the rule head *)
            H_{m+1} := candidates(H_m);              (* create rule heads with one item more *)
            m := m + 1;                              (* increment the head item counter *)
        until H_m = ∅ or m ≥ |f|;                    (* until there are no more rule heads *)
    end;                                             (* or the antecedent would become empty *)
    return R;                                        (* return the rules found *)
end                                                  (* apriori *)

function candidates(F_k)                             (* generate candidates with k + 1 items *)
begin
    C := ∅;                                          (* initialize the set of candidates *)
    forall f1, f2 ∈ F_k                              (* traverse all pairs of frequent item sets *)
    with f1 = {i1, ..., i_{k−1}, i_k}                (* that differ only in one item and *)
    and f2 = i1, ..., i_{k−1}, i'_k                  (* are in a lexicographic order *)
    and i_k < i'_k do begin                          (* (the order is arbitrary, but fixed) *)
        f := f1 ∪ f2 = i1, ..., i_{k−1}, i_k, i'_k;  (* the union of these sets has k + 1 items *)
        if ∀i ∈ f : f − {i} ∈ F_k                    (* only if all k element subsets are frequent, *)
        then C := C ∪ {f};                           (* add the new item set to the candidates *)
    end;                                             (* (otherwise it cannot be frequent) *)
    return C;                                        (* return the generated candidates *)
end                                                  (* candidates *)

function prune(C, T, s_min)                          (* prune infrequent candidates *)
begin
    forall c ∈ C do                                  (* initialize the support counters *)
        s(c) := 0;                                   (* of all candidates to be checked *)
    forall t ∈ T do                                  (* traverse the transactions *)
        forall c ∈ C do                              (* traverse the candidates *)
            if c ∈ t                                 (* if the transaction contains the candidate, *)
            then s(c) := s(c) + 1;                   (* increment the support counter *)
    F := ∅;                                          (* initialize the set of frequent candidates *)
    forall c ∈ C do                                  (* traverse the candidates *)
        if s(c) ≥ s_min                              (* if a candidate is frequent, *)
        then F := F ∪ {c};                           (* add it to the set of frequent candidates *)
    return F;                                        (* return the pruned set of candidates *)
end                                                  (* prune *)
```

D Laitteiden komponenttitasot



E Mittaushetkien komponettitasot