Ilkka Rautiainen

Tiedonlouhinnan ja koneoppimisen menetelmät verkkohyökkäysten havaitsemisessa

Tietotekniikan pro gradu -tutkielma

23. marraskuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Ilkka Rautiainen

Yhteystiedot: ilkka.t.rautiainen@student.jyu.fi

Ohjaaja: Timo Hämäläinen

Työn nimi: Tiedonlouhinnan ja koneoppimisen menetelmät verkkohyökkäysten havaitsemisessa

Title in English: Methods of data mining and machine learning in network intrusion detection

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 119+15

Tiivistelmä: Tietokoneverkoissa toimivat hyökkääjät yrittävät jatkuvasti ohittaa käytössä olevia turvajärjestelmiä, ja pyrkivät kehittämään uusia tapoja kohteidensa vahingoittamiseen. Näitä hyökkäyksiä voidaan havaita tunkeilijan havaitsemisjärjestelmällä (engl. *intrusion detection system* eli *IDS*). Yleisesti käytettyjen väärinkäytöspohjaisten menetelmien lisäksi hyökkäyksiä voidaan havaita anomaliapohjaisilla eli tiedonlouhinnan ja koneoppimisen menetelmillä. Nämä menetelmät kykenevät teoriassa havaitsemaan myös aiemmin tuntemattomat hyökkäykset. Tiedonlouhinnan ja koneoppisen menetelmien hyödyntäminen IDS-järjestelmissä on laajalti tutkittu alue, mutta useat ongelmat ovat vielä vailla ratkaisua. Tässä tutkielmassa esitellään alan taustoja, menetelmiä ja ongelmia. Lopuksi tarkastellaan ja vertaillaan *k*-means-klusterointia, tukivektorikonetta ja keinotekoisia neuroverkkoja käytännössä. Käytännön osiossa hyödynnetään vuonna 2015 julkaistua synteettistä UNSW-NB15-verkkodataa. Tutkielman tulokset vahvistavat käsitystä, jonka mukaan yksittäinen menetelmä ei voi tehokkaasti löytää kaikkien ryhmien verkkohyökkäyksiä. Parhaimmat tulokset saavutetaankin erilaisten menetelmien tehokkaalla yhdistämisellä.

Avainsanat: tiedonlouhinta, koneoppiminen, IDS-järjestelmät, anomaliat, klusterointi, tukivektorikone, neuroverkot

Abstract: Attackers in computer networks are constantly trying to bypass existing network security systems and developing new ways to harm their targets. An intrusion detection system (IDS) can be used to detect these attacks. In addition to commonly used signature-based detection it is also possible to implement anomaly-based methods of data mining and machine learning to detect the attacks. In theory, these methods should be able to detect also previously unknown attacks. The field of anomaly-based detection has been widely studied, but some problems remain unsolved. This thesis discusses the background, methods and problems present in the field of study. It also examines and compares k-means clustering, support vector machine and artificial neural networks empirically. The methods are implemented with an aim to find the malicious traffic from a public UNSW-NB15 dataset released in 2015. The thesis confirms that a single anomaly-based method is not able to detect all types of network attacks reliably. An ideal solution for intrusion detection would be to efficiently combine several different detection methods.

Keywords: data mining, machine learning, intrusion detection, anomaly detection, clustering, support vector machine, neural network

Termiluettelo

APT	*Advanced persistent threat*, tarkkaan kohdistettu pitkäkestoinen verkkohyökkäys
AUC	*Area under curve*, ROC-käyrän alle jäävä pinta-ala
C-SVC	*C-support vector classification*, joustavan marginaalin luokitin tukivektorikoneessa
CIFE	*Conditional informative feature extraction*, eräs suodatinmallin mukainen ominaisuuksien valinnan algoritmi
CMIM	*Conditional mutual info maximisation*, eräs suodatinmallin mukainen ominaisuuksien valinnan algoritmi
CVE	*Common vulnerabilities and exposures*, tunnetut haavoittuvuudet sisältävä tietokanta
DoS	*Denial of service*, palvelunestohyökkäys
FPR	*False positive rate*, virheellisesti normaaliksi luokiteltujen haitallisen liikenteen havaintojen osuus
IDS	*Intrusion detection system*, tunkeilijan havaitsemisjärjestelmä
IPS	*Intrusion prevention system*, tunkeilijan estojärjestelmä
JMI	*Joint mutual information*, eräs suodatinmallin mukainen ominaisuuksien valinnan algoritmi
MSE	*Mean squared error*, keskineliövirhe
NIDS	*Network-based IDS*, verkkopohjainen IDS
PCA	*Principal component analysis*, pääkomponenttianalyysi
RMSE	*Root mean squared error*, keskineliövirheen neliöjuuri
ROC	*Receiver operating characteristic*, luokittelijan toimivuutta kuvaava käyrä
SVM	*Support vector machine*, tukivektorikone
TCP	*Transmission control protocol*, yhteydellinen verkkoprotokolla
TNR	*True negative rate*, oikein luokiteltujen normaalin liikenteen havaintojen osuus
UDP	*User datagram protocol*, yhteydetön verkkoprotokolla

Kuviot

Taulukot

Sisältö

1 Johdanto

IP-protokollaa käyttävien tietoverkkojen ja niitä hyödyntävien laitteiden räjähdysmäinen leviäminen 1990-luvulta lähtien on tuonut mukanaan erinäisiä ongelmia. Uuteen toimintaympäristöön ovat löytäneet tiensä myös monenlaiset laittomia keinoja hyödyntävät toimijat.

Tietojärjestelmien osittainen avoimuus takaa, että lailliset käyttäjät voivat niitä verkon yli mahdollisimman vaivattomasti hyödyntää. Tämä avoimuus avaa samalla mahdollisuuden myös verkossa toimivalle hyökkääjälle. Tietojärjestelmään hyökkääjää voi motivoida esimerkiksi näyttämisenhalu, ideologia tai raha (Day 2013, luku 8). Pahimmillaan hyökkäyksen takana voi olla järjestäytynyt taho, esimerkiksi valtiollinen toimija (West 2013). Osaavalla hyökkääjällä on mahdollisuus saada huomattavaa tuhoa aikaan, sillä nyky-yhteiskunta on paljolti rakennettu tietoverkkojen ja -järjestelmien varaan. Näitä hyökkäyksiä pyritään estämään tunkeilijan havaitsemisjärjestelmillä (*intrusion detection system* tai *IDS*).

Verkkopohjainen IDS-järjestelmä (*network-based intrusion detection system* tai *NIDS*) tutkii verkossa liikkuvia paketteja (Debar, Dacier ja Wespi 1999), esimerkiksi organisaation verkkoon tulevaa että sieltä lähtevää verkkoliikennettä. Järjestelmän tarkoitus on löytää tämän datan joukosta hyökkäykset ja muu haitallinen liikenne. Verkossa liikkuvat datamäärät voivat kuitenkin olla valtavia, ja jokaisen datapaketin analysointi on kallista. IDS-järjestelmät ovat myös vain yksi osa toimivaa tietoturvaa. Omalta osaltaan tietojärjestelmän turvallisuudesta pitävät edelleen huolen esimerkiksi palomuurit sekä ihmisylläpitäjät.

Anomalioiden havaitsemiseen (*anomaly detection*) käytetyillä tiedonlouhinnan (*data mining*) ja koneoppimisen (*machine learning*) menetelmillä on lähes rajattomasti eri käyttökohteita, joista verkkohyökkäysten löytäminen on vain yksi sovellus. Perusidea anomaliapohjaisessa hyökkäysten havaitsemisessa on, että järjestelmä tietää mikä on niin sanotusti normaalia liikennettä. Normaalista verkkoliikenteestä tarpeeksi poikkeavaa liikennettä pidetään potentiaalisesti haitallisena (Om ja Gupta 2015, luku 2.4).

Käytännössä edelleen luotetuin ja käytetyin tapa havaita hyökkäykset ovat kuitenkin väärinkäytöspohjaiset (*misuse-based* tai *signature-based*, myös *knowledge-based*) järjestelmät, jotka etsivät verkkoliikenteestä tunnettuja kuvioita (Sommer ja Paxson 2010; Om ja Gupta

2015).Väärinkäytöspohjaisten järjestelmien hyvä puoli on alhainen väärien hälytysten määrä, mutta ne eivät kykene havaitsemaan uusia tai riittävästi muunneltuja hyökkäyksiä. Anomaliapohjaiset järjestelmät voivat ainakin teoriassa havaita myös tällaisen liikenteen. Nämä kaksi tapaa havaita haitallista liikennettä eivät sulje toisiaan pois, vaan ne voivat toimia IDS-järjestelmässä rinnakkain.

IDS-järjestelmien ja menetelmien keskinäinen vertailu on osoittautunut haastavaksi. Universaalisti toimivaa ja yksiselitteisesti parasta menetelmää ei ole olemassa, vaan paras lähestymistapa ongelmaan riippuu aina useasta tekijästä. Todellista verkkoliikennettä vastaavan merkityn datan kerääminen verkosta on hankalaa. Tätä dataa vaaditaan järjestelmän opettamiseen sekä testaamiseen. Alan tutkimuksissa käytetäänkin edelleen yleisesti muutamia vanhoja yleisesti saatavilla olevia datajoukkoja (*data set*), joita ovat mm. KDD Cup 99 (“KDD cup 1999 data” 1999) ja vuosien 1998–2000 DARPA-joukot (“DARPA intrusion detection data sets” 2014).

Yleisesti saatavilla olevia datajoukkoja on kuitenkin kritisoitu muun muassa siitä, että ne vastaavat huonosti oikeaa verkossa tapahtuvaa liikennettä (ks. esim. Mahoney ja Chan 2003). Tutkimukset eivät useinkaan ole keskenään suoraan vertailukelpoisia, sillä vaikka käytetty testausdatajoukko olisikin sama, niin muun muassa testausdatasta valittu osajoukko vaihtelee tutkimuksesta toiseen. Tällaisella testausdatalla saatu mittaustulos ei todennäköisesti kerro paljoakaan menetelmän suorituskyvystä tositilanteessa. Myös alan vertaisarvioitujen tutkimusten raportointikäytänteitä ovat kritisoineet olennaisten tietojen selostamatta jättämisestä esimerkiksi Tavallaee, Stakhanova ja Ghorbani (2010) sekä Weller-Fahy, Borghetti ja Sodemann (2015). Raportoinnin puutteet taas johtavat yleensä siihen, että muiden tutkijoiden on lähes mahdotonta toistaa tehtyjä kokeita. Gates ja Taylor (2007) taas ovat kritisoineet alan tutkimuksessa usein pohjalla vaikuttavia oletuksia. Yksi näistä on se, että hyökkäykset tai haitallinen verkkoliikenne on poikkeus, jolloin suurin osa verkkoliikenteestä olisi niin sanottua normaalia liikennettä. Lisäksi Sommer ja Paxson (2010) ovat esittäneet, että anomaliapohjaiset menetelmät eivät sovellu verkkohyökkäysten havaitsemiseen yhtä hyvin kuin muihin aloihin.

Tässä tutkielmassa tarkastellaan tiedonlouhinnan ja koneoppimisen menetelmien hyödyntämistä IDS-järjestelmien toteutuksessa. Tutkielma sisältää empiirisen osion, jossa esiteltyjä

menetelmiä hyödynnetään käytännössä. Empiirisessä osiossa eteen tulevat valinnat, kuten parametrien arvot, pyritään perustelemaan kattavasti. Mittaustulokset esitetään usealla alalla yleisesti käytössä olevalla mittarilla, ja ne perustuvat lähes aina useasta simulointikerrasta saatuihin tuloksiin. Tulokset eri hyökkäysryhmille esitetään kunkin menetelmän kohdalla myös erikseen. Empiirisen osion datajoukkona käytetään vuonna 2015 julkaistua UNSW-NB15-dataa (Moustafa ja Slay 2015).

Tutkielman luvussa 2 esitellään IDS-järjestelmien jaottelua ja niissä käytettyjä tekniikoita. Luvussa 3 esitellään verkkohyökkäysten jaottelua. Luku 4 esittelee anomaliapohjaiset menetelmät ja anomalioiden havaitsemisessa huomioitavia asioita, joita ovat esimerkiksi erilaiset anomalioiden ja datan tyypit. Luvussa 5 esitellään tutkielmassa käytettävä datajoukko, tarkastellaan menetelmien vertailun ongelmia ja mittareita sekä valitaan tutkielmassa implementoitavat menetelmät. Luvuissa 6, 7 ja 8 toteutetaan valitut menetelmät käytännössä. Ensimmäinen edellä mainituista toteutusluvuista käsittelee k-means-klusterointia, toinen tukivektorikonetta ja kolmas keinotekoisia neuroverkkoja. Lopuksi luvussa 9 esitetään tutkielmasta tehdyt johtopäätökset.

2 IDS-järjestelmien jaottelu ja tekniikkaa

Tunkeilijan havaitsemis- eli IDS-järjestelmiä on tutkittu ja kehitetty viimeisimpinä vuosikymmeninä. Ensimmäisten joukossa idean mainitsi Anderson (1980), ja varsinaisen lähtölaukauksen alalle antoi malli, jonka Denning (1987) esitteli.

2.1 IDS-järjestelmien pääjaottelu

IDS-järjestelmät voidaan jakaa kahteen pääryhmään sen perusteella, mitä ne valvovat:

Isäntäpohjainen IDS (*host-based IDS* tai *HIDS*) valvoo suoraan suojattavaa järjestelmää, esimerkiksi yksittäistä tietokonetta. HIDS voi valvoa tarkemmin esimerkiksi järjestelmässä käynnissä olevia prosesseja. HIDS:n eräänä hyvänä puolena on mahdollisuus tarkkailla paremmin myös salatusti siirrettyä, vasta kohteessa purettua, verkkoliikennettä. Isäntäpohjaisen IDS:n suorittama valvonta vie kuitenkin aina resursseja pois järjestelmän varsinaiselta käytöltä (Day 2013, luku 14; Bhuyan, Bhattacharyya ja Kalita 2014).

Verkkopohjainen IDS (*network-based IDS* tai *NIDS*) sijoitetaan yleensä verkon solmukohtaan, toisin sanoen kohtaan josta suurin osa valvottavan organisaation verkkoliikenteestä kulkee. Toisin kuin isäntäpohjainen IDS, verkkopohjainen IDS valvoo vain siis sen kautta kulkevia IP-paketteja. Verkkopohjaisen IDS:n eräs suurimmista ongelmista on runsasliikenteisen verkon valvonta, jossa on lähes mahdotonta tarkastaa yksityiskohtaisesti jokaista verkossa liikkuvaa pakettia (Day 2013, luku 12; Pandya 2013, luku 11).

Koska yllä mainituissa kahdessa erityyppisessä IDS-järjestelmässä on molemmissa hyvät ja huonot puolensa, voivat ne käytännön verkkoympäristössä hyvin täydentää toisiaan. IDS-järjestelmän määritelmään kuuluu myös, että se ainoastaan valvoo ja ilmoittaa havaitsemastaan mahdollisesti haitallisesta toiminnasta, mutta ei estä sitä. Tällaisen toiminnan estäminen kuuluu tunkeilijan esto- eli IPS-järjestelmälle (*intrusion prevention system* tai *IPS*). Havaitsemis- ja estämistoiminnot on kuitenkin usein käytännössä yhdistetty samaan järjestelmään. Kolmas aihepiiriin liittyvä käsite on IRS (*intrusion response system*). Tällaisen järjestelmän tarkoituksena on reagoida jollain tavalla havaittuun tunkeutumiseen. IRS voi esi-

merkiksi muokata automaattisesti palomuurin sääntöjä tai lähettää hälytyksen verkon ylläpitäjälle (Stakhanova, Basu ja Wong 2007).

Edelleen IDS-järjestelmät voidaan jakaa kolmeen eri ryhmään sen mukaan, kuinka ne suorittavat järjestelmien valvontaa:

Väärinkäytöspohjaiset (*misuse-based* tai *signature-based*, myös *knowledge-based*) järjestelmät käyttävät haitallisen toiminnan tunnistamiseen etukäteen määriteltyjä valmiita sääntöjä tai tunnusmerkkejä (*signatures*). Menetelmä tunnistaa hyökkäyksen, jos toiminta noudattaa tarkoin jotain aiemmin havaittua ja tunnettua kaavaa. Väärinkäytöspohjainen järjestelmä ei kuitenkaan osaa tunnistaa hiukankin muunneltua hyökkäystä, ellei sen tunnusmerkkejä ole aiemmin määritelty tiedetyt hyökkäykset sisältävään tietokantaan (Bhuyan, Bhattacharyya ja Kalita 2014).

Anomaliapohjaiset (*anomaly-based*, myös *behaviour-based*) järjestelmät vaativat alussa niin sanotun normaalin toiminnan profiilin luontia. Perusoletuksena haitallinen toiminta on anomaalista, poikkeus normaaliin tilaan. Kun normaalin toiminnan profiili on selvillä, teoriassa on helppo erottaa haitallinen toiminta normaalin toiminnan joukosta, sillä tarpeeksi normaalista poikkeava toiminta voidaan luokitella haitalliseksi. Käytännössä asia ei kuitenkaan ole näin yksinkertaista. Jo normaalin toiminnan profiilin luominen tuottaa ongelmia. Ei voida käytännössä helposti esimerkiksi taata, että profiilin luomisen aikaan valvottavassa ympäristössä ei tapahdu haitallista toimintaa. Anomaliapohjainen järjestelmä usein tasapainoileekin liiallisten väärien hälytysten ja normaaliksi toiminnaksi virheellisesti tunnistetun haitallisen toiminnan välillä. Lisäksi anomaliapohjainen järjestelmä on esimerkiksi runsasliikenteisessä verkossa paljon laskentaresursseja vievä, ja voi muodostua eräänlaiseksi pullonkaulaksi (Bhuyan, Bhattacharyya ja Kalita 2014). Ringberg, Roughan ja Rexford (2008) jakavat anomaliapohjaiset IDS-järjestelmät kahteen ryhmään. Ensimmäisen ryhmän järjestelmät keskittyvät löytämään täsmällisesti määritellyt tietyntyyppiset hyökkäykset, esimerkiksi palvelunestohyökkäykset tai madot. Toisen ryhmän järjestelmät pyrkivät taas löytämään kaiken ”normaalista käyttäytymisestä poikkeavan”.

Hybridit (*hybrid*) järjestelmät käyttävät valvontaan sekä väärinkäytöspohjaisia että anomaliapohjaisia menetelmiä. Näin saadaan hyödynnettyä molempien menetelmien hyvät puolet.

2.2 Verkkoliikenteen tarkkailu reaaliaikaisesti ja jälkikäteen

IDS-järjestelmä voi toimia joko erillisessä (*offline*) tai suorassa (*online*) eli reaaliaikaisessa tilassa. Erilliskäytössä oleva IDS-järjestelmä analysoi verkkoliikennettä jälkikäteen, ja sen tuottamissa havainnoissa voi olla suurikin viive. Suorakäytössä toimiva IDS-järjestelmä sen sijaan analysoi verkkoliikennettä reaaliaikaisesti, ja se kykenee luokittelemaan liikennettä nopeasti, korkeintaan muutaman minuutin viiveellä (Sangkatsanee, Wattanapongsakorn ja Charnsripinyo 2011).

Reaaliaikaisen järjestelmän hyödyt näyttävät varsin selviltä, sillä parhaimmillaan se kykenee havaitsemaan tunkeutumisen lähes välittömästi. Kuitenkin erilliskäytössä toimiva IDS-järjestelmä tuo mukanaan joitakin etuja, joita reaaliaikaisesti toimiva IDS-järjestelmä ei voi kunnolla hyödyntää. Ensinnäkin verkosta kaapattu verkkopakettien tiedot sisältävä Tcpdump-raakadata voidaan koostaa tiedostoiksi, joissa jokainen rivi sisältää yhden havaitun yhteyden tiedot (Kayacık, Zincir-Heywood ja Heywood 2004). Yksi yhteys on tarkkaan määritellyllä protokollalla ja aikavälillä tapahtuva TCP-verkkopakettien sarja, jonka aikana data siirtyy lähde-IP:stä kohde-IP:hen (Stolfo ym. 1999).

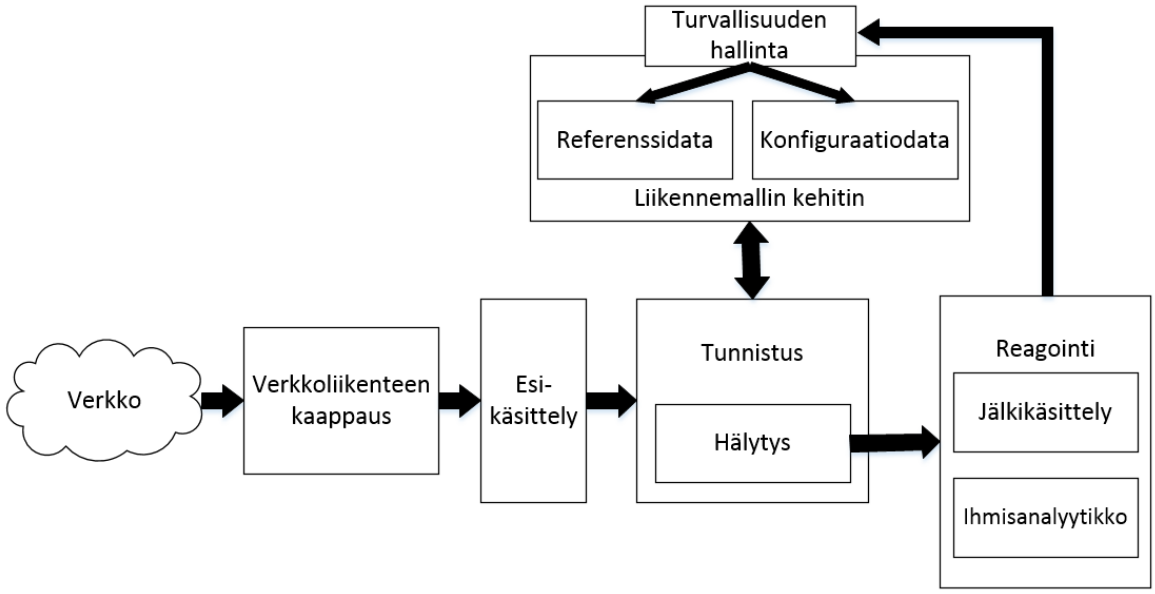
Yhteydestä saadaan tällä tavalla koostetuksi perusominaisuudet (*basic features*), kuten yhteyden kesto sekä yhteyden aikana lähetettyjen ja vastaanotettujen pakettien yhteenlasketut koot. Lisäksi voidaan laskea erilaisia johdannaisominaisuuksia (*derived features*), jotka voidaan esimerkiksi KDD Cup 99 -datajoukossa (“KDD cup 1999 data” 1999) jakaa kolmeen kategoriaan. Ensimmäisen kategorian johdannaisominaisuudet, sisältöominaisuudet (*content features*), käyttävät hyväkseen verkkopakettien hyötykuormien sisältöä. Toinen kategoria, aikaperusteiset liikenteen ominaisuudet (*time-based traffic features*), pyrkii saamaan esille yksittäisen yhteyden aikaikkunan ulkopuoliset liikenteen piirteet, esimerkiksi yhteyksien määrän samaan isäntään. Kolmas kategoria, isäntäpohjaiset liikenteen ominaisuudet (*host-based traffic features*), pyrkii yhteyksien määrän perusteella saamaan esiin pidemmällä aikavälillä tapahtuvat hyökkäykset (Kayacık, Zincir-Heywood ja Heywood 2004).

Erilliskäytössä olevalla IDS-järjestelmällä on reaaliaikaista järjestelmää väljemmät nopeusvaatimukset, joten se voi käyttää hyökkäysten löytämiseen aikavaatimuksiltaan raskaampia tiedonlouhinta-algoritmeja (Brugger 2004; Pharate ym. 2015). Virvilis, Serrano ja Dandu-

rand (2014) esittävät, että reaaliaikaisella liikenteen tarkkailulla ei kyetä lainkaan havaitsemaan edistyneempiä verkkohyökkäyksiä. Kohdistetut ja pitkäkestoiset APT (*advanced persistent threat*) -hyökkäykset (ks. luku 3.5) on heidän mukaansa mahdollista havaita vain analysoimalla kerättyä verkkodataa jälkikäteen edistyneillä tiedonlouhinta-algoritmeilla. Esimerkiksi Suomen ulkoministeriössä vuonna 2013 havaittu verkkovakoilu ehti kestää useita vuosia, jopa neljä vuotta (Halminen ym. 2013; Leppänen 2013). Brugger (2004) ehdottaakin, että IDS-järjestelmä voi hyödyntää samaan aikaan sekä erilliskäyttöä että reaaliaikaista tunnistusta. Tällöin molempien keinojen hyvät puolet voivat täydentää toisiaan.

2.3 Anomaliapohjaisen NIDS-järjestelmän arkkitehtuuri

Anomalioiden tunnistukseen pohjautuvaa verkkopohjaista IDS-järjestelmää kutsutaan yleisesti lyhenteellä ANIDS tai A-NIDS. NIDS-järjestelmän arkkitehtuurista on olemassa useita hieman toisistaan eroavia yleistettyjä malleja. Axelsson (1999) esitteli ns. tyypillisen IDS-järjestelmän mallin, ja siinä nähdyt moduulit esiintyvät pääpiirteissään myös muiden myöhemmin esittelemissä malleissa. Seuraavat kuviossa 1 esitetyt arkkitehtuurin osat ovat maininneet sekä Catania ja Garino (2012) että Bhuyan, Bhattacharyya ja Kalita (2014).



Kuvio 1: Yleinen anomaliapohjaisen NIDS-järjestelmän arkkitehtuuri.

Verkkoliikenteen kaappaus (*Traffic data acquisition* (Catania ja Garino 2012) tai *Captu-*

ring traffic (Bhuyan, Bhattacharyya ja Kalita 2014)) -moduuli kerää järjestelmän tarvitseman raakadatan. Tämä data voi muodoltaan olla kaapattuja erillisiä verkkopaketteja tai vuopohjaista (*flow-based*) koottua tietoa verkkoliikenteestä. Sama IDS-järjestelmä voi myös kerätä ja käyttää hyväkseen yhtä aikaa sekä tietoa verkkopaketeista että vuopohjaista dataa (Sperotto ym. 2010). Verkkopaketeista voidaan kerätä pelkät otsikkotiedot (*header*) tai ottaa mukaan myös paketin hyötykuorma (*payload*).

Esikäsittely (*Traffic features generator* tai *Preprocessing*) -moduuli erottelee kaapatusta liikenteestä varsinaiseen analyysiin päätyvät osat. Moduulin toimintaa esitellään tarkemmin luvussa 4.5.

Tunnistus (*Incident detector* tai *Matching mechanism & Alarm*) -moduuli pyrkii löytämään valitusta datasta mahdollisen haitallisen verkkoliikenteen. Kun moduuli löytää epäilyttävää liikennettä, se suorittaa hälytyksen. Bhuyan, Bhattacharyya ja Kalita (2014) jakavat mallissaan tämän moduulin kahteen osaan. Siinä hälytyksen tekevä osa on omana moduulinaan.

Liikennemallin kehitin (*Traffic model generator* tai *Reference data & Configuration data*) -moduuli sisältää normaalin liikenteen profiilin eli referenssidatan, jota Tunnistus-moduuli käyttää hyväkseen tutkiessaan järjestelmään tulevaa liikennettä. Bhuyan, Bhattacharyya ja Kalita (2014) jakavat mallissaan moduulin kahteen tai kolmeen osaan, joista Referenssidata (*Reference data*) sisältää valmiin referenssidatan ja Konfiguraatiodata (*Configuration data*) vasta osittain luodun keskeneräisen referenssidatan. Turvallisuuden hallinta (*Security manager*) -moduuli voidaan myös liittää tähän kokonaisuuteen. Moduulin tehtävänä on päivittää tietoa tarvittaessa sekä Referenssidata- että Konfiguraatiodata-moduulissa.

Reagointi (*Response management*) -moduuli reagoi Tunnistus-moduulin tekemiin hälytyksiin. Hälytykseen reagointi voi olla kokonaan ihmisten käsissä, täysin automatisoitua tai jotain siltä väliltä. Bhuyan, Bhattacharyya ja Kalita (2014) jakavat mallissaan tämän moduulin tehtävät kahteen erilliseen osaan. Nämä osat ovat Ihmisanalyytikko (*Human analyst*) ja Jälkikäsittely (*Post-processing*). Ihmisanalyytikko toimii hälytykseen sopivaksi katsomallaan tavalla saatujen tietojen pohjalta ja Jälkikäsittely hoitaa hälytyksen jälkeiset toimenpiteet.

2.4 Luvun yhteenveto

Luvussa esiteltiin IDS-järjestelmien pääjaottelu, vertailtiin reaaliaikaisesti ja jälkikäteen tapahtuvaa verkkoliikenteen tarkkailua sekä kuvailtiin anomaliapohjaisen IDS-järjestelmän arkkitehtuuri. Luvun pääsisältö voidaan tiivistää seuraavasti:

- IDS-järjestelmät voidaan jakaa kahteen pääryhmään, jotka ovat isäntäpohjainen IDS (HIDS) ja verkkopohjainen IDS (NIDS).
- Väärinkäytöspohjaiset IDS-järjestelmät tunnistavat haittaliikenteen ennalta määriteltyjen sääntöjen avulla.
- Anomaliapohjaiset IDS-järjestelmät tunnistavat haittaliikenteen vertaamalla verkkoliikennettä aiemmin määriteltyyn profiiliin.
- Reaaliaikaisessa tilassa toimiva IDS-järjestelmä pyrkii tunnistamaan haittaliikenteen suoraan, joko välittömästi tai pienellä viivellä.
- Erilliskäytössä oleva IDS-järjestelmä koostaa tietoa verkkoliikenteestä pidemmällä aikaikkunalla, ja pyrkii tunnistamaan haittaliikenteen analysoimalla tätä tietoa jälkikäteen.
- Anomaliapohjaisen IDS-järjestelmän arkkitehtuuriin kuuluvat seuraavat osat: verkkoliikenteen kaappaus, esikäsittely, tunnistus, liikennemallin kehitin ja reagointi.

3 Verkkohyökkäysten lajit

Verkkohyökkäys pyrkii jollakin tavalla horjuttamaan yhtä tai useampaa tietoturvan perusperiaatetta (Kotzanikolaou ja Douligeris 2007, luku 1.1):

- **Luottamuksellisuus** (*confidentiality*) tarkoittaa, että luvattomilta käyttäjiltä estetään pääsy informaatiosisältöön.
- **Eheys** (*integrity*) tarkoittaa, että luvattomat käyttäjät eivät pääse muokkaamaan informaatiosisältöä.
- **Saatavuus** (*availability*) varmistaa, että informaatiosisältö ei luvattomasti joudu väliaikaisesti tai pysyvästi saavuttamattomaksi.
- **Todentaminen** (*authentication*) varmistaa tietojärjestelmää käyttävän kohteen identiteetin. Kohde voi olla esimerkiksi ihminen, tietokone tai tietokoneohjelma.
- **Kiistämättömyys** (*nonrepudiation*) tarkoittaa, että toimijat eivät voi jälkikäteen kiistää toimintaansa tietojärjestelmässä.

Kohdennetun verkkohyökkäyksen voidaan usein katsoa koostuvan kolmesta eri vaiheesta. Ensimmäinen vaihe on tiedustelu (*reconnaissance*), jonka tavoitteena on kerätä tietoa hyökkäyksen kohteesta ja sen heikkouksista. Toisessa vaiheessa suoritetaan varsinainen hyökkäys (*compromise*) ja kolmannessa vaiheessa pyritään peittämään hyökkäyksen jäljet (*cover-up*) (Chen ja Walsh 2013). Bhuyan, Bhattacharyya ja Kalita (2011) lisäävät ensimmäisen vaiheen jälkeen vielä suojattomuuden arvioinnin (*assessing vulnerability*), jossa hyökkääjä testaa hyökkäystapaansa joihinkin verkossa oleviin kohteisiin ennen varsinaista hyökkäystä. Seuraavan vaiheen varsinainen hyökkäys suoritetaan näiden murrettujen kohteiden kautta. Kohdentamattomat hyökkäykset, joihin kuuluvat esimerkiksi verkkomadot ja hyökkäyskoodia suorittamaan pyrkivät verkkosivustot, eivät kuitenkaan noudata näitä vaiheita (Chen ja Walsh 2013). Joissakin kohdennetuissa hyökkäyksissä tunkeutuja voi myös haluta, että kohde havaitsee tunkeutumisen. Tällöin hyökkääjä voi esimerkiksi näkyvästi turmella yrityksen verkkosivuston (West 2013).

Eri tavoin muodostettuja verkkohyökkäysten ryhmittelyjä on tehty lukuisia (Lazarevic, Kumar ja Srivastava 2005). Kendall (1999), jonka ryhmittely on kenties käytetyin, jakoi verkko-

hyökkäykset neljään pääryhmään: palvelunestohyökkäykset (luku 3.1), tiedon urkinta (3.2), R2L (3.3) ja U2R (3.4). Lisäksi on olemassa muita hyökkäyksiä (3.5), jotka eivät kunnolla mahdu Kendallin ryhmittelyn alle.

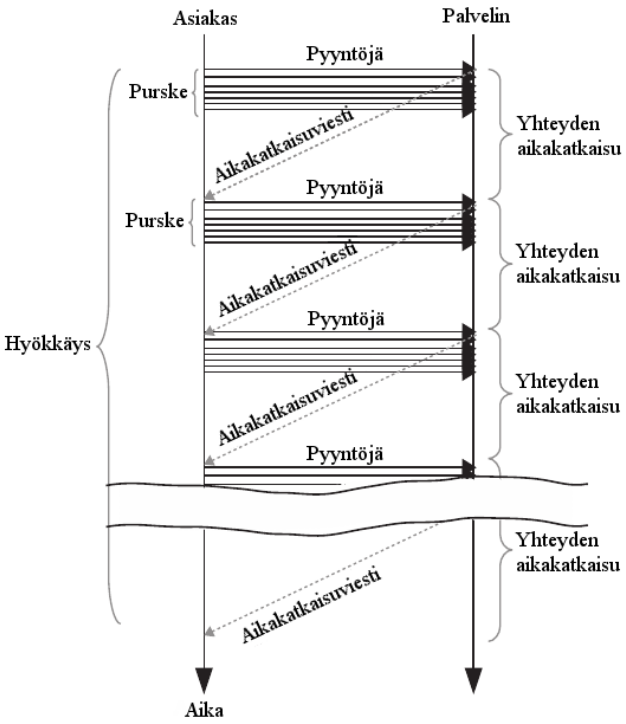
3.1 Palvelunestohyökkäykset

Palvelunestohyökkäyksen (*denial of service* eli *DoS*) tavoitteena on tukahduttaa järjestelmän resurssit ja estää sen normaali käyttö. Kohteena oleva järjestelmän resurssi voi olla verkon kaista, järjestelmän laskentaresurssit tai käyttöjärjestelmän tietorakenteet (Zargar, Joshi ja Tipper 2013). Hyökkäykset voidaan karkeasti jakaa kahteen ryhmään. Ensimmäisen ryhmän hyökkäykset pyrkivät hyödyntämään järjestelmän olemassa olevia ohjelmistojen haavoittuvuuksia tai ominaisuuksia. Toisen ryhmän hyökkäykset taas perustuvat valtavaan pakettimäärään, jolla kohde pyritään ”hukuttamaan” (*flooding*) (Hussain, Heidemann ja Papadopoulos 2003). Palvelunestohyökkäyksen voi suorittaa monella eri tavalla, ja useaa eri protokollaa käyttäen. Käytetyimpiä hyökkäyksissä ovat TCP-, UDP-, ICMP- ja DNS-protokollien mukaiset verkkopaketit (Zargar, Joshi ja Tipper 2013).

Hajautettu palvelunestohyökkäys (*distributed denial of service* eli *DDoS*) käyttää hyökkäykseen useiden koneiden muodostamaa verkkoa. Tämä verkko koostuu useimmiten niin sanotuista zombikoneista, jotka on kaapattu hyökkäyskäyttöön mahdollisesti jopa käyttäjän huomaamatta. Nämä kaapatut koneet muodostavat bottiverkon (*botnet*), jonka haltija voi vapaasti määrätä hajautetun palvelunestohyökkäyksen kohteen. Ensimmäinen ilmoitettu DDoS-hyökkäys tapahtui vuonna 1999, mistä lähtien suurin osa palvelunestohyökkäyksistä on ollut toteutustavaltaan hajautettuja (Zargar, Joshi ja Tipper 2013; Mirkovic ja Reiher 2004).

Puolustautuminen laajan bottiverkon suorittamaa hajautettua palvelunestohyökkäystä vastaan on haastavaa. Puolustautujan päällimmäisenä tavoitteena onkin hyökkäyksen mahdollisimman nopea havaitseminen, ja usein hyökkäyksen kohde voikin olla parasta vain irrottaa verkosta. Ideaalinen tilanne puolustautujalle on saada katkaistua hyökkäys mahdollisimman läheltä sen alkupistettä (Zargar, Joshi ja Tipper 2013). Niin sanotut alhaisen nopeuden (*low-rate*) (ks. kuvio 2) ja hitaan nopeuden (*slow-rate*) hyökkäykset on vaikeampi tunnistaa. Nämä hyökkäykset eivät pyri aiheuttamaan suurta verkkoliikennekuormaa, vaan käyttävät

hyväkseen palvelinten sisäänrakennettuja ominaisuuksia. Tällaiset hyökkäykset pyrkivät pitämään palvelimen resurssit varattuna hyökkääjän tekemille väärille pyynnöille, jolloin oikeiden käyttäjien pääsy palveluun estyy (Gonzalez ym. 2015).



Kuvio 2: Alhaisen nopeuden palvelunestohyökkäyksen malli (Gonzalez ym. 2015).

3.2 Tiedon urkinta

Tiedon urkinnan (*probing*, *surveillance* tai *reconnaissance*) voidaan katsoa olevan aiemmin mainitun kolmivaiheisen verkkohyökkäyksen ensimmäinen, tiedusteluun liittyvä, vaihe. Hyökkääjän tavoitteena on siis kerätä tietoa hyökkäyksen kohteesta. Hyökkääjää voi kiinnostaa muun muassa luettelo verkossa olevista koneista, koneiden avoimet portit ja koneille asennetut ohjelmat. Vaikka näillä hyökkäyksillä ei aiheuteta suoraa vahinkoa, niin tiedon urkinta mahdollistaa verkkohyökkäyksen seuraavat vaiheet (Ahmed, Mahmood ja Hu 2016; Al-Jarrah ja Arafat 2014). Urkinta voi olla joko aktiivista tai passiivista. Aktiivinen urkinta tapahtuu lähettämällä verkkopaketteja kohteisiin ja seuraamalla kohteiden vastauksia. Passiivista urkintaa taas suoritetaan seuraamalla verkkoliikennettä jossakin havaintopisteessä. Tämä voidaan toteuttaa esimerkiksi peilaamalla kaikki reitittimen kautta kulkeva liikenne

johonkin seurattuun porttiin (Bou-Harb, Debbabi ja Assi 2014).

Urkintaan on olemassa runsaasti vapaasti saatavilla olevia helppokäyttöisiä ja ilmaisia työkaluja. Eräs tunnetuimmista on Nmap, jonka kotisivuilla ohjelmaa markkinoidaan omassa hallinnassa olevan verkon auditointityökaluna ("Nmap: the network mapper - Free security scanner" 2016). Ohjelman ohjeistus vihjaa kuitenkin yhdeksi monista käyttötavoista käyttäjän manipuloinnin (*social engineering*). Kun ohjelman avulla on saatu lista koneelle asennetuista ohjelmista, voi hyökkääjä ohjeen mukaan soittaa suoraan koneen käyttäjälle ja tekeytyä jonkin koneelle asennetun ohjelman tukihenkilöksi. Tukihenkilö väittää ohjelman vaativan tietoturvapäivityksen. Jos kaikki menee hyökkääjän suunnittelemalla tavalla, niin koneelle asennetaan hyökkääjän valitsema haittaohjelma ("Nmap: Remote OS detection" 2016).

3.3 R2L

Remote to local eli *R2L*, myös *remote to user* eli *R2U* on etähyökkäys, jossa hyökkääjä onnistuu – käyttäen hyväkseen jotakin kohdekoneen haavoittuvuutta – saamaan pääsyn kohdeverkossa olevalle tietokoneelle (Hoque ym. 2014). Kendall (1999) käyttää jaottelussaan termiä R2U, ja painottaa hyökkääjän saavan pääsyn vain käyttäjätasoiselle koneen tilille. R2L-hyökkäyksen määrittelyissä (esim. Bhuyan, Bhattacharyya ja Kalita 2014) voidaan korostaa, että hyökkäyksellä saadaan pääsy joko käyttäjätasoiseen tai järjestelmänvalvojatasoiseen paikalliseen käyttäjätiliin. Tällaisen hyökkäyksen voi toteuttaa esimerkiksi lähettämällä kohteelle troijalaisen sisältävän sähköpostin. Troijalainen taas asentaa koneelle takaoven (*backdoor*), jonka avulla hyökkääjä saa pääsyn koneelle, ja samalla käytännössä paikallisen järjestelmänvalvojan oikeudet (Hoque ym. 2014).

Hyökkäyksessä auttavat suuresti aiemmin urkintavaiheessa kerätyt tiedot koneella olevista ohjelmista ja niiden versioista, sillä haavoittuvuudet koskevat usein vain ohjelmien tiettyjä versioita.

3.4 U2R

User to root tai *U2R* on hyökkäys, jossa etähyökkääjä on jo jotakin kautta saanut pääsyn käyttäjätasoiselle koneen tilille. Tämän tilin kautta hyökkääjä pyrkii saamaan itselleen järjestelmänvalvojatasoiset käyttäjäoikeudet (Kendall 1999). Tähän hyökkääjä pyrkii hyödyntämällä käyttöjärjestelmän tai ohjelmistojen haavoittuvuuksia tai vikoja (Hoque ym. 2014).

Jos kohdekoneessa on ohjelmistoja jotka eivät ole ajan tasalla, niin hyökkääjä voi etsiä sopivan tiedossa olevan haavoittuvuuden ja hyödyntää sitä. Hyvä aika iskeä on myös välittömästi haavoittuvuuden julkistamisen jälkeen, jolloin ohjelmistoja ei aina olla ehditty päivittää. Osaavammalla hyökkääjällä voi olla tiedossaan niin sanottuja nollapäivähaavoittuvuuksia (*zero-day vulnerability*). Ne hyödyntävät niitä ohjelmistojen haavoittuvuuksia joita ei ole julkistettu, ja joihin ei ole julkaistu korjauspäivityksiä (Bilge ja Dumitras 2012). Jos hyökkääjällä ei ole osaamista tai resursseja etsiä uusia haavoittuvuuksia itse, niitä voi myös ostaa internetistä (Danchev 2008). Mitä laajemmin uutta haavoittuvuutta hyödynnetään, sitä todennäköisemmäksi sen julkistaminen ja korjaaminen tulee. Tämän vuoksi hyökkääjän kannattaa usein käyttää tällaisia haavoittuvuuksia vain tarkoin rajatulla tavalla.

3.5 Muut hyökkäykset

Muiden muassa Lazarevic, Kumar ja Srivastava (2005, luku 2) lisäsivät yllä esiteltyihin hyökkäystyyppeihin virukset, madot sekä troijalaiset.

Virus on ohjelma, joka luvatta kopioi itseään kiinnittymällä toisiin ohjelmiin. Samalla ohjelmat joihin virus kiinnittyy saastuvat. Yleensä virus vaatii joitakin käyttäjän toimenpiteitä levitäkseen. Se voi esimerkiksi levitä verkossa, kun joku käyttäjä selailee saastuneen koneen verkkolevyn tiedostoja (Lazarevic, Kumar ja Srivastava 2005; Bhuyan, Bhattacharyya ja Kalita 2014).

Mato (*worm*) on itseään kopioiva ohjelma, joka pyrkii käyttämään levittäytymiseensä tietokoneesta löytyviä verkkotoimintoja. Madot voivat käyttää leviämisessä hyväkseen esimerkiksi sähköpostiohjelmien tai verkkojakojen ominaisuuksia. Madon tehtävänä voi olla esimerkiksi saada kone osaksi bottiverkkoa, jolloin konetta voidaan käyttää hajautetussa palve-

lunestohyökkäyksessä (Lazarevic, Kumar ja Srivastava 2005).

Troijalainen (*Trojan*) on hyödyllisen näköiseksi naamioitunut ohjelma, joka sisältää todellisuudessa jonkin haitallisen ominaisuuden. Troijalainen voi sisältää esimerkiksi takaportin jolla hyökkääjä pääsee ohjaamaan kohdekonetta. Toisin kuin virus tai mato, troijalainen ei kopioi itseään, vaan se luottaa leviämisessä ihmisten aktiivisiin asennustoimiin (Bhuyan, Bhattacharyya ja Kalita 2014).

Lisäksi viimeisimpien vuosien aikana on otettu käyttöön termi **APT** (*advanced persistent threat*). Termillä viitataan edistyneisiin hyökkäyksiin, jotka pyrkivät väistämään olemassa olevat suojausmekanismit joko hyödyntämällä nollapäivähaavoittuvuuksia tai muokkaamalla hyökkäyksiä niin, että mekanismit eivät niitä tunnista. APT-hyökkäys suoritetaan hitaasti jaoteltuna pitkälle aikavälille, jotta sen tunnistaminen olisi vaikeampaa. Hyökkääjällä on myös runsaasti resursseja käytössään, mikä tarkoittaa useimmiten valtiollista toimijaa. Tällaisen hyökkäyksen kohde tai kohteet valitaan hyvin tarkoin, eikä tarkoituksena ole levittää hyökkäystä mahdollisimman laajalle. Kohde voi olla esimerkiksi sellainen organisaation osasto, jolta puuttuu teknistä osaamista, ja joka siten vähemmän todennäköisesti huomaa hyökkäyksen (Virvilis, Serrano ja Dandurand 2014).

3.6 Luvun yhteenveto

Luvussa esiteltiin tietoturvan perusperiaatteet ja verkkohyökkäysten lajit. Luvun pääsisältö voidaan tiivistää seuraavasti:

- Verkkohyökkäys pyrkii horjuttamaan yhtä tai useampaa tietoturvan perusperiaatetta, jotka ovat luottamuksellisuus, eheys, saatavuus, todentaminen ja kiistämättömyys.
- Kohdennettu verkkohyökkäys koostuu vähintään kolmesta vaiheesta, jotka ovat tiedustelu, hyökkäys ja peittely.
- Verkkohyökkäykset voidaan jakaa neljään pääryhmään, jotka ovat palvelunestohyökkäykset, tiedon urkinta, R2L ja U2R. Muita verkkohyökkäyksiä ovat virukset, madot, troijalaiset sekä APT-hyökkäykset.

4 Anomalioiden havaitseminen

Järjestelmän toimivuuden arvioinnissa jokainen yksittäinen käsitelty havainto voidaan jakaa neljään ryhmään järjestelmän tekemän luokittelun perusteella (Witten ja Frank 2005, luku 5.7):

True negative (TN)

Oikein luokiteltu normaalin liikenteen havainto.

True positive (TP)

Oikein luokiteltu haitallisen liikenteen havainto.

False negative (FN)

Virheellisesti haitalliseksi luokiteltu normaalin liikenteen havainto.

False positive (FP)

Virheellisesti normaaliksi luokiteltu haitallisen liikenteen havainto.

Mitä vähemmän virheellisiä luokitteluja järjestelmä tekee, sitä käyttökelpoisempi se luonnollisesti on. Muun muassa Gates ja Taylor (2007) toteavat kuitenkin, että alan tutkimuksissa raportoidaan usein hyvin korkeita tunnistusprosentteja, vaikka samalla väärien hälytysten määrä tekee järjestelmän käytännössä käyttökelvottomaksi.

4.1 Anomalian määrittely

Sanakirjan mukaan anomalia on ”jotakin epätavallista tai odottamatonta” tai ”poikkeama yleisestä säännöstä” (“Definition of anomaly by Merriam-Webster” 2015). Anomaliapohjaisten IDS-järjestelmien tutkimuksessa tehdään joidenkin tutkijoiden mukaan usein oletuksia, jotka eivät välttämättä pidä paikkaansa. Gates ja Taylor (2007) esittävät tällaisina oletuksina muun muassa sen, että järjestelmään tuleva haitallinen liikenne tai hyökkäys on tunnistettavissa oleva anomalia. Toinen heidän mukaansa usein nähty oletus on edellinen käänteisenä, eli että anomalia on haitallista liikennettä. Taitava hyökkääjä osaa kuitenkin naamioida hyökkäyksensä niin, että se ”hukkuu” normaalin liikenteen sekaan, eikä erotu lainkaan tunnistettavissa olevana anomaliana. Järjestelmän anomaliaksi – ja samalla siis myös epäilyttäväksi – näkemä tapahtuma voi taas hyvin olla aiemmin näkemätöntä harmitonta liikennettä.

Tavallaee, Stakhanova ja Ghorbani (2010) painottavat, että jokaisen tutkimuksen kohdalla on tärkeää määritellä mitä anomalia tutkimuksessa tarkoittaa. Tämä on tarpeen, koska toisen olemassa olevan tulkinnan mukaan anomalia tarkoittaa automaattisesti haitallista liikennettä. Uusissakin tutkimuksissa (ks. esim. Mazel, Fontugne ja Fukuda 2014) anomalia-käsitteellä saatetaan viitata suoraan haittaliikenteeseen. Tässä tutkielmassa anomalialla pyritään kuitenkin viittamaan sanakirjan mukaiseen määritelmään, eikä käsite ole synonyymi verkkohyökkäykselle tai haitalliselle liikenteelle.

4.2 Anomaliapohjaisten menetelmien jaottelu

Anomaliapohjaisten menetelmien taustalla ovat tiedonlouhinnan (*data mining*) ja koneoppimisen (*machine learning*) tutkimusalat. Witten ja Frank (2005, luku 1.1) kuvailevat tiedonlouhintaa käyttökelpoisten kuvioiden tai mallien löytämiseksi suuresta tietomäärästä, joko automaattisesti tai puoliautomaattisesti. Koneoppimisen määritelmään (Alpaydın 2004, luku 1.1; Witten ja Frank 2005) taas liittyy vahvasti ajatus siitä, että tietokoneohjelma kykenee itsenäisesti ja jatkuvasti parantamaan suorituskykyään muuttamalla käyttäytymistään aiempien havaintojensa pohjalta.

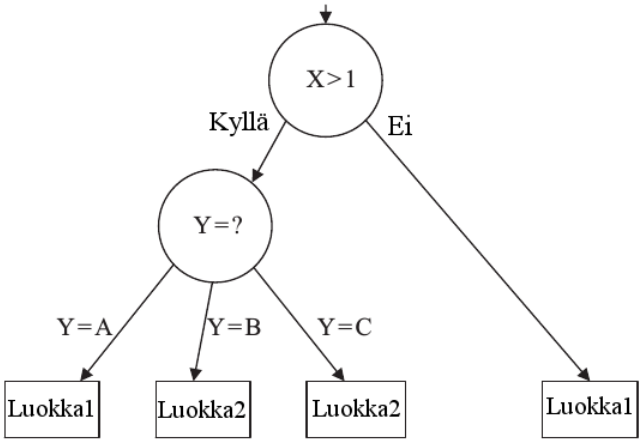
Erilaisia anomaliapohjaisia menetelmiä on olemassa runsaasti, ja niitä kehitellään koko ajan lisää. Menetelmien jaottelu erillisiin pääryhmiinkään ei ole yksiselitteistä, sillä moni menetelmä sopii hyvin usean eri pääotsikon alle. Viime vuosina jaottelua IDS-järjestelmien näkökulmasta ovat tehneet muun muassa Patcha ja Park (2007), Liao ym. (2013) ja Bhuyan, Bhattacharyya ja Kalita (2014).

Yhteistä näille jaotteluille on, että tilastollisille menetelmille on kaikissa varattu oma pääryhmänsä. Selkein ero näkyy pääryhmien määrässä. Patcha ja Park (2007) jakavat menetelmät kolmeen eri pääryhmään ja Liao ym. (2013) kasvattavat määrää neljään. Bhuyan, Bhattacharyya ja Kalita (2014) jakavat menetelmät peräti kuuteen pääryhmään:

Tilastolliset menetelmät (*statistical methods*) voivat perustua esimerkiksi määriteltyjen kynnysarvojen ylityksiin tai alituksiin, keskiarvoihin, keskihajontoihin ja todennäköisyyksiin (Liao ym. 2013). Esimerkkejä tilastollisista menetelmistä ovat Bayes-verkot (*Bayesian networks*), n-gram-analyysi ja assosiaatiosääntöjen louhinta (*association rule mining*) (Bhuyan,

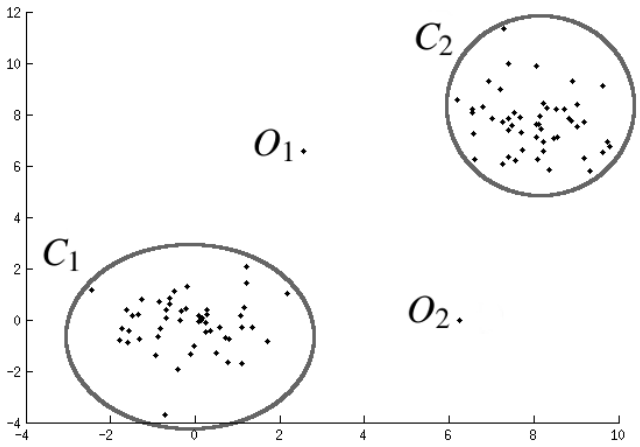
Bhattacharyya ja Kalita 2014).

Luokittelupohjaiset (*classification-based*) menetelmät pyrkivät jakamaan järjestelmään tulevan datan eri kategorioihin. Onnistunut jako edellyttää kuitenkin hyvälaatuista kategorisoitua opetusdataa. Esimerkkejä luokittelupohjaisista menetelmistä ovat tukivektorikoneet (*support vector machine* eli *SVM*), *k*:n lähimmän naapurin menetelmä (*k-nearest neighbour* eli *kNN*) ja päätöspuut (*decision tree*) (Bhuyan, Bhattacharyya ja Kalita 2014). Esimerkki yksinkertaisesta päätöspuusta on esitetty kuviossa 3. Tämä päätöspuu jakaa dataa kahteen luokkaan muuttujien X ja Y arvojen perusteella.



Kuvio 3: Yksinkertainen päätöspuu (Kantardzic 2011).

Klusterointi- ja poikkeava havainto -pohjaiset (*clustering and outlier-based*) menetelmät pyrkivät jakamaan järjestelmään tulevan datan klustereihin. Yhteen ja samaan klusteriin kuuluva data vaikuttaa järjestelmän näkökulmasta enemmän tai vähemmän samantyyppiseltä verrattuna muissa klustereissa olevaan dataan. Kantardzic (2011, luku 9) jakaa klusterointitekniikat kolmeen pääryhmään, jotka ovat hierarkkinen (*hierarchical*), ositteleva (*partitional*) ja inkrementaalinen (*incremental*). Tekniikoiden erot ilmenevät klustereiden muodostustavassa, ja siinä miten suurelle datamassalle tekniikka soveltuu käytettäväksi. Poikkeaviksi havainnoiksi (*outliers*) kutsutaan dataa, joka erottuu selvästi muusta joukosta. Klusteroinnin kontekstissa poikkeava havainto voi sijaita esimerkiksi kahden klusterin välissä. Tällaisen poikkeavan havainnon ei voida katsoa kuuluvan kumpaankaan klusteriin (Bhuyan, Bhattacharyya ja Kalita 2014; Kantardzic 2011, luku 2.6). Kuviossa 4 on esitetty esimerkki kahdesta klusterista C_1 ja C_2, sekä niiden välissä sijaitsevista poikkeavista havainnoista O_1 ja O_2.



Kuvio 4: Esimerkki kahdesta klusterista ja kahdesta poikkeavasta havainnosta.

Pehmeän laskennan (*soft computing*) menetelmät sisältävät tekniikat, jotka eivät anna ongelmiin niin sanottuja tarkkoja ratkaisuja. Näihin kuuluvat ainakin geneettiset algoritmit (*genetic algorithms*), keinotekoiset neuroverkot (*artificial neural networks* eli *ANN*), sumeat joukot (*fuzzy sets*), karkeat joukot (*rough sets*), muurahaiskolonnaoptimointi (*ant colony optimization*) ja keinotekoiset immuunijärjestelmät (*artificial immune systems* eli *AIS*) (Bhuyan, Bhattacharyya ja Kalita 2014).

Tietämyspohjaiset (*knowledge-based*) menetelmät käyttävät tunnistukseen ennalta määriteltyjä haitallisen liikenteen kuvaavia sääntöjä. Järjestelmään tulevaa liikennettä voidaan verrata tiedettyyn hyökkäykseen, jolloin tiedetyn hyökkäyksen tavoin käyttäytyvä liikenne on suurella todennäköisyydellä haitallista. Esimerkkejä tietämyspohjaisista menetelmistä ovat sääntöpohjaiset (*rule-based*), ontologiapohjaiset (*ontology-based*), logiikkapohjaiset (*logic-based*) menetelmät ja tilasiirtymäanalyysi (*state-transition analysis*) (Bhuyan, Bhattacharyya ja Kalita 2014).

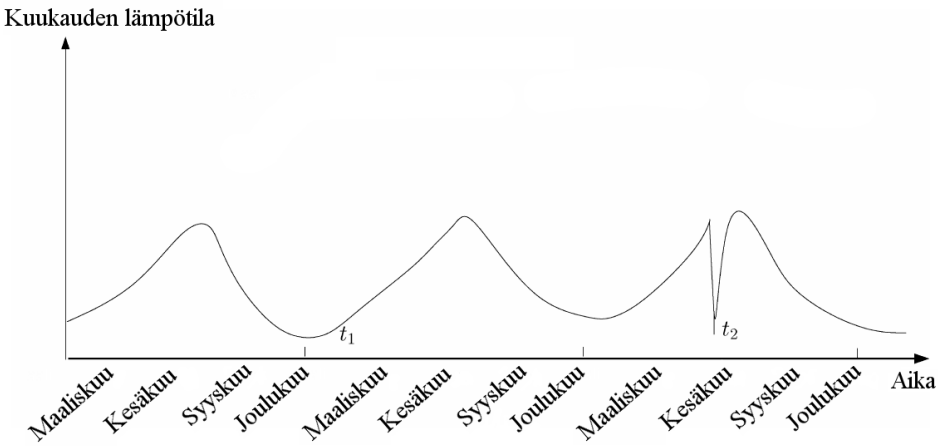
Yhdistelevät (*combination learner*) menetelmät käyttävät hyväkseen kahta tai useampaa erillistä anomaliapohjaista menetelmää. Koska kaikissa menetelmissä on omat hyvät ja huonot puolensa, sopivilla yhdistelmillä voidaan saada aikaan hyvin suoriutuvia järjestelmiä.

4.3 Anomalioiden ryhmittely ja havaitsemistekniikat

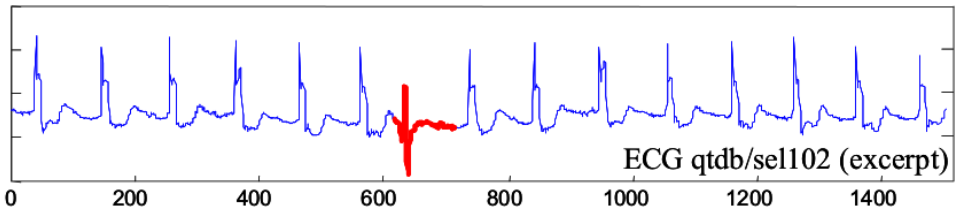
Datassa esiintyvät anomaliat voidaan jakaa kolmeen ryhmään (Chandola, Banerjee ja Kumar 2009; Ahmed, Mahmood ja Hu 2016):

- **Pistemäisen** (*point*) anomalian voidaan katsoa olevan poikkeava suhteessa muuhun dataan. Esimerkkejä pistemäisistä anomalioista ovat kuviossa 4 esitetyt pisteet O_1 ja O_2. Ahmed, Mahmood ja Hu (2016) pyrkivät jakamaan eri verkkohyökkäystyypit (ks. luku 3) anomaliaryhmiin. Heidän mukaansa pistemäisiä anomalioita edustavat R2L- (luku 3.3) ja U2R-hyökkäykset (luku 3.4).
- **Kontekstuaalinen** (*contextual*) anomalia on poikkeava omassa kontekstissaan. Kuviossa 5 on esitetty lämpötilakeskiarvoja aikasarjana kolmen vuoden ajalta. Kolmannen vuoden kesäkuun aikana on mitattu keskiarvoksi poikkeuksellisen alhainen lämpötila, joka kuitenkin olisi normaali talvikuukausina. Kuviossa 5 on t_2 siis kontekstuaalinen anomalia. Ahmed, Mahmood ja Hu (2016) ryhmittelevät verkkohyökkäyksistä tiedon urkinnat (luku 3.2) kontekstuaalisiksi anomalioiksi.
- **Kollektiiviseksi** (*collective*) anomaliaksi kutsutaan toisiinsa liittyvien havaintojen joukkoa, joka on poikkeuksellinen suhteessa muuhun dataan. Kuviossa 6 on esitetty sydänsähkökäyrä eli elektrokardiogrammi, jossa kollektiivinen anomalia on esitetty paksumpana viivana ja punaisella värillä. Tässä tapauksessa anomalia viittaa sydämen lisälyöntisyyteen (*premature ventricular contraction*) (Lin ym. 2005). Verkkohyökkäysten kontekstissa kollektiivisia anomalioita edustavat ainakin palvelunestohyökkäykset (luku 3.1) (Ahmed, Mahmood ja Hu 2016).

Anomaliapohjaisen IDS-järjestelmän kouluttaminen vaatii datajoukon, joka sisältää sekä normaalia liikennettä että hyökkäyksiä tai muuta haitallista liikennettä. Datajoukon instanssit voivat sisältää tiedon siitä, millaista liikennettä instanssi edustaa. Tätä tietoa kutsutaan nimiöksi (*label*). Datajoukon nimiöt voivat yksinkertaisimmillaan olla esimerkiksi *normaali* ja *hyökkäys*, mutta liikennettä on mahdollista yksilöidä tarkemminkin. Tarkemmin nimiöidyssä datajoukossa voidaan esimerkiksi nähdä, mitkä instanssit edustavat jotakin tiettyä hyökkäystyyppiä. Käytetyn datajoukon nimiöimistavan perusteella voidaan anomaliapohjaiset menetelmät jakaa kolmeen ryhmään (Chandola, Banerjee ja Kumar 2009):



Kuvio 5: Esimerkki kontekstuaalisesta anomaliasta (Chandola, Banerjee ja Kumar 2009).



Kuvio 6: Esimerkki kollektiivisesta anomaliasta (Lin ym. 2005).

- **Ohjatussa** (*supervised*) tekniikassa datajoukko on kokonaan nimiöity ja sisältää tiedon sekä normaalin että haitallisen liikenteen instansseista. Haitallisen liikenteen nimiöinti sisältää tiedon hyökkäystyypistä. Näin tarkkaan nimiöidyn käyttökelpoisen datajoukon saaminen käyttöön voi kuitenkin olla hankalaa, ja simuloidut datajoukot ovat käytännössä ainoa vaihtoehto.
- **Puoliohjatussa** (*semi-supervised*) tekniikassa vain datajoukon normaali liikenne on nimiöity, ja tarkempaa tietoa haitallisesta liikenteestä ei ole saatavilla.
- **Ohjaamattomassa** (*unsupervised*) tekniikassa datajoukko ei sisällä nimiöintiä lainkaan. Tällaista dataa on samalla kaikkein yksinkertaisinta saada käyttöön.

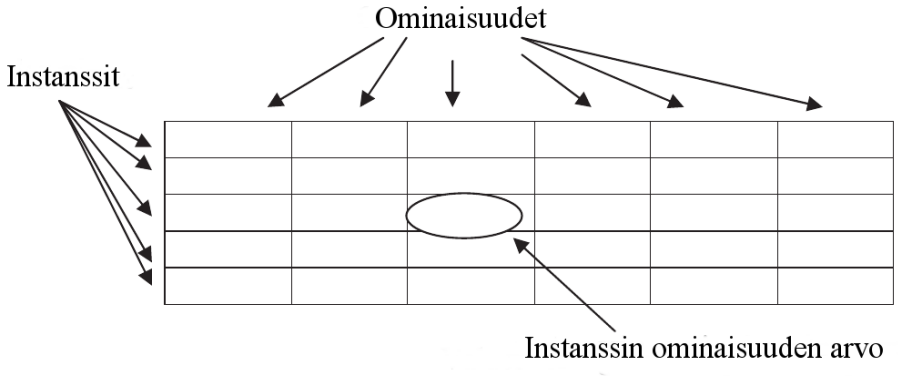
4.4 Datajoukot ja datan lajit

Kantardzic (2011, luku 1.4) jakaa analysoinnin kohteena oleva datan rakenteelliseen, puolirakenteelliseen tai rakenteettomaan dataan:

- **Rakenteellinen** (*structured*) data tarkoittaa kaikkea sitä tietoa, joka on tarkkaan määritellyllä tavalla numeerisesti tai sanallisesti datajoukossa ilmaistu. Tällaisen tiedon voi mielekkäästi esittää kuvion 7 mukaisessa perinteisessä taulukossa.
- **Puolirakenteellinen** (*semi-structured*) data tarkoittaa tietoa, jolla on jokin sisäinen rakenne, mutta joka datajoukossa esitetään vain yksittäisenä elementtinä tai tiedostona. Tällaista tietoa ovat esimerkiksi erilaiset sähköisessä muodossa olevat asiakirjat.
- **Rakenteeton** (*unstructured*) data tarkoittaa tietoa, jolla ei ole helposti määriteltävissä olevaa sisäistäkään rakennetta. Tällaista tietoa on esimerkiksi videokuva.

Datajoukko voi sisältää kaikkia edellä mainittuja. Verkkoliikennedataa toimittavat työkalut esittävät tiedon yleensä tarkkaan määritellyssä muodossa, toisin sanoen data on rakenteellista. Verkkopakettien hyötykuormassa sen sijaan voi liikkua mitä tahansa tietoa. Jos data siis sisältää pakettien otsikkotietojen lisäksi hyötykuormatietoa, on tuo osuus datasta vain harvoin rakenteellista. Verkkoliikennedatan voi usein myös katsoa olevan temporaalista (*temporal*), sillä jokainen instanssi sisältää aikaleiman, jonka mukaan instanssit on järjestetty (Kantardzic 2011, luku 12.2).

Rakenteellista dataa sisältävä datajoukko voidaan esittää kuvion 7 mukaisena taulukkona, jossa kukin kerätty instanssi (kutsutaan myös nimillä näyte tai havainto, engl. *instance*, *object*, *record*, *point*, *vector*, *pattern*, *event*, *case*, *sample*, *observation* tai *entity*) (Tan, Steinbach ja Kumar 2006, luku 2.1) on omalla vaakarivillään. Tällöin taulukon pystyriveillä esitetään kunkin instanssin ominaisuudet (*features*) eli muuttujat (*attributes* tai *variables*). Yksittäinen taulukon solu viittaa siis yhden instanssin yhteen ominaisuuteen. Lisäksi ominaisuudella eli muuttujalla on vielä jokin arvo (*value*) (Kantardzic 2011, luku 1.4). Muuttujien tyyppien tunnistaminen käytetyssä datajoukossa on tärkeää, sillä jo esikäsittelyvaiheessa on tiedettävä muuttujille tehtävissä olevat lailliset operaatiot. Yleinen tapa jaotella datajoukon muuttujat on jakaa ne kategorisiin ja numeerisiin muuttujiin (ks. luku 4.4.1). Vaihtoehtoisesti muuttujat voidaan jakaa myös diskreetteihin ja jatkuviin muuttujiin (ks. luku 4.4.2).



Kuvio 7: Datajoukon esitys taulukkomuodossa (Kantardzic 2011).

4.4.1 Kategoriset ja numeeriset muuttujat

Kategorisille (*categorical*) muuttujille yhteistä on, että niiltä puuttuu useimmat numeroiden ominaisuudet. Nämä muuttujat ovat datassa usein sanallisessa muodossa, mutta ne voidaan esittää myös numeroin. Kategoriset muuttujat voidaan jakaa kahteen ryhmään, jotka ovat nominaalinen ja ordinaalinen:

- **Nominaalisen** (*nominal*) muuttujan arvoa tulee käsitellä pelkkänä sanana, jota voi verrata muihin datajoukon arvoihin ainoastaan sen osalta, onko arvo sama vai erilainen. Toisin sanoen nominaalisten muuttujien käsittelyssä käytössä ovat ainoastaan operaatiot $=$ ja $\neq$, ja mitään järjestystä arvojen välillä ei voida saada aikaan (Tan, Steinbach ja Kumar 2006, luku 2.1). Verkkoliikenteen kontekstissa tällainen muuttuja voi olla esimerkiksi liikennöintiin käytetty protokolla.
- **Ordinaalisen** (*ordinal*) muuttujan arvot voidaan järjestää keskinäisen vertailun perusteella. Esimerkiksi muuttujan mahdolliset arvot joukosta $\{hyvä, parempi, paras\}$ voidaan mielekkäällä tavalla laittaa järjestykseen. Ordinaalisten muuttujien käsittelyssä käytössä ovat siten operaatiot $=$, $\neq$, $<$, $\leq$, $>$ ja $\geq$ (Tan, Steinbach ja Kumar 2006; Kantardzic 2011).

Numeerisille (*numeric*) muuttujille yhteistä on, että ne nimensä mukaisesti esitetään aina numeroin, joko kokonaislukuina tai reaalilukuina, ja niillä on useimmat numeroiden ominaisuudet (Tan, Steinbach ja Kumar 2006, luku 2.1). Numeeriset muuttujat voidaan jakaa kahteen ryhmään, joista ensimmäiseen kuuluvat käyttävät välimatka-asteikkoa ja toiseen kuulu-

vat suhdeasteikkoa:

- **Välimatka- eli intervalliasteikkoa** (*interval scale*) käyttävien muuttujien arvon ”0” voidaan sanoa olevan mielivaltaisesti määritelty. Nollapiste ei tällä asteikolla tarkoita absoluuttista pienintä mahdollista arvoa. Yleisin esimerkki tällaisesta muuttujasta on lämpötilan ilmaisu celsiusasteina, missä 2 °C ei tarkoita kahta kertaa niin suurta lämpötilaa kuin 1 °C (Kantardzic 2011; Tan, Steinbach ja Kumar 2006). Toinen esimerkki on kalenteri, jossa lasketaan päiviä (“Välimatka- eli intervalliasteikko” 2016). Välimatka-asteikkoa käyttävien muuttujien käsittelyssä käytössä ovat siten operaatiot $=, \neq, <, \leq, >, \geq, +$ ja $-$ (Tan, Steinbach ja Kumar 2006).
- **Suhdeasteikkoa** (*ratio scale*) käyttävillä muuttujilla sitä vastoin on absoluuttinen nollapiste (Kantardzic 2011). Esimerkiksi jos verkkopaketin koko on ilmaistu tavuina, niin 50 tavun paketti on kooltaan kaksi kertaa 25 tavun pakettia suurempi. Tähän ryhmään kuuluvat useimmat numeeriset muuttujat verkkoliikenteen kontekstissa. Suhdeasteikkoa käyttävien muuttujien käsittelyssä käytössä ovat siten operaatiot $=, \neq, <, \leq, >, \geq, +, -, *$ ja $/$ (Tan, Steinbach ja Kumar 2006).

4.4.2 Diskreetit ja jatkuvat muuttujat

Toinen yleinen tapa on jaotella muuttujat diskreetteihin ja jatkuviin muuttujiin. Jatkuvat muuttujat ovat aina myös numeerisia, mutta diskreetit muuttujat voivat olla joko kategorisia tai numeerisia:

- **Diskreetti** (*discrete*) muuttuja voi saada arvoja vain äärellisestä joukosta tai numeroituvasta äärettömästä joukosta (Tan, Steinbach ja Kumar 2006). Eräs esimerkki äärellisestä joukosta on aiemmin mainittu $\{hyvä, parempi, paras\}$. Numeroituvasta äärettömästä joukosta taas arvoja saa esimerkiksi kokonaislukuina tapahtumien lukumäärästä kirjaa pitävä laskuri, jolla ei ole teoreettista ylärajaa. Myös kaksiarvoinen binäärinen (*binary*) muuttuja on tyypiltään diskreetti (Tan, Steinbach ja Kumar 2006; Kantardzic 2011).
- **Jatkuvan** (*continuous*) muuttujan arvot ilmaistaan reaalilukuina, mikä teoriassa sallii mittaustulosten ilmaisun äärettömällä tarkkuudella (Tan, Steinbach ja Kumar 2006;

Kantardzic 2011). Eräs esimerkki jatkuvasta muuttujasta on verkkoyhteyden kesto. Jatkuvia muuttujia on mahdollista muuntaa diskreeteiksi. Tätä toimenpidettä kutsutaan diskretisoinniksi (*discretisation*) (ks. luku 4.5.2) (Han ja Kamber 2006, luku 2.6).

4.5 Datan esikäsittely

Datan esikäsittely (*preprocessing*) sisältää kaikki ne toimet, joilla analysoitava data saatetaan valmiiksi varsinaisen analyysin suorittavalle järjestelmän osalle eli tunnistusmoduulille, ts. tiedonlouhinta-algoritmille. Esikäsittely on olennainen osa anomaliapohjaisen NIDS-järjestelmän toimintaa, ja sen aikana tehdyt toimet vaikuttavat suoraan järjestelmän suorituskykyyn (Davis ja Clark 2011). Vaadittavat toimenpiteet riippuvat analysoitavasta datasta. Paljon käsinsyötettyä tietoa sisältävässä perinteisessä tietokannassa on hyvin todennäköisesti virheellisesti tai moneen kertaan syötettyä tietoa, ja osa tiedosta on väistämättä puutteellista. Näitä puutteita ja virheitä korjaavia toimenpiteitä kutsutaan **datan siivoukseksi** (*data cleaning*) (Han ja Kamber 2006, luku 2.3). Verkkoliikennedata on kuitenkin aina jonkin ohjelman automaattisesti tuottamaa, joten se harvoin vaatii tällaista siivousta. Käsiteltävä data voi myös olla jakautuneena useammassa eri lähteessä. Tällöin eri lähteissä oleva data on yhdistettävä samassa muodossa olevaksi kokonaisuudeksi. Tätä toimenpidettä kutsutaan **datan yhdistämiseksi** (*data integration*) (Han ja Kamber 2006, luku 2.4.1; Kantardzic 2011, luku 1.5).

Muut esikäsittelyn aikana tehtävät toimenpiteet voidaan jakaa kahteen pääryhmään, jotka ovat datan vähentäminen (ks. luku 4.5.1) sekä datamuunnokset (luku 4.5.2).

4.5.1 Datan vähentäminen

Datan vähentäminen (*reduction*) tarkoittaa niitä toimia, joilla analysoitavan datan määrää pyritään vähentämään analysoinnin nopeuttamiseksi. Vähentäminen pyritään tekemään niin, että mahdollisimman suuri osa alkuperäisen datan olennaisesta informaatiosta säilyy (Kantardzic 2011, luku 3). Datan vähentämisen menetelmiä ovat muun muassa ominaisuuksien valinta, ulottuvuuksien vähentäminen ja näytteenotto.

Ominaisuuksien valinnan (*feature selection*) eli tarkemmin ominaisuuksien osajoukon va-

linnan (*feature subset selection*) aikana karsitaan datajoukosta varsinaisen analyysin kannalta epäolennaiset ja toisteiset ominaisuudet (Han ja Kamber 2006, luku 2.5.2; Tan, Steinbach ja Kumar 2006, luku 2.3.4). Vaikka kaikki datajoukon ominaisuudet olisivatkin sinänsä tärkeitä, resurssien rajallisuuden vuoksi kaikkien ominaisuuksien huomioinen on yleensä epäkäytännöllistä (Om ja Gupta 2015, luku 2.6). Valinnan aikana tarkkailtavia parametreja ovat ainakin laskenta-aika ja osuvuus. Lisäksi voidaan kiinnittää huomiota siihen, että analysoitavan datajoukon ymmärrettävyys ja esitettävyys paranee tiedon karsimisen ansiosta (Kantardzic 2011, luku 3.1). Samalla on kuitenkin pidettävä mielessä, että analyysin kannalta olennaiset ominaisuudet voivat vaihdella sen mukaan mitä datasta kulloinkin ollaan etsimässä (Blum ja Langley 1997). Verkkohyökkäysten kontekstissa parhaaseen tunnistustarkkuuteen saatetaan näin päästä kouluttamalla järjestelmä erityyppisten hyökkäysten tunnistukseen käyttäen useita erilaisia ominaisuuksien osajoukkoja.

Ominaisuuksien osajoukon valintaan on olemassa useita erilaisia malleja. Ensinnäkin **sisäänrakennetussa** (*embedded*) mallissa ominaisuuksien valinta on kiinteänä osana varsinaista tiedonlouhinta-algoritmia eli anomaliapohjaista menetelmää (ks. luku 4.2). Sisäänrakennettuna ominaisuuksien valinta on esimerkiksi monissa luokittelupohjaisissa päätöspuita käyttävissä menetelmissä (Blum ja Langley 1997; Tan, Steinbach ja Kumar 2006, luku 2.3.4). Ominaisuuksien valinta on järjestelmässä erillisenä osana seuraavissa malleissa:

- **Suodatinmallin** (*filter model*, *method* tai *approach*) (ks. kuvio 8) mukaisessa järjestelmässä ominaisuuksien valinnan suorittaa kokonaan erillinen esikäsittelijä, joka ei valinnan aikana ole vuorovaikutuksessa järjestelmän tiedonlouhinta-algoritmin kanssa. Suodatin käyttää valinnassa hyväkseen ainoastaan datajoukon sisäisiä ominaispiirteitä. Valinnan jälkeen esikäsittelijä lähettää suodattamansa ominaisuuksien osajoukon analyysin suorittavalle tiedonlouhinta-algoritmille (Blum ja Langley 1997).

  Eräs esimerkki suodatinmallin ominaisuuksien valinnan menetelmästä on JMI (*joint mutual information*), jossa pistemäärä J ominaisuudelle X_k määritellään informaatioteorian käsitteistöllä ilmaistuna (Brown ym. 2012)

  $$J_{jmi}(X_k) = \sum_{X_j \in S} I(X_k X_j; Y), \tag{4.1}$$

  missä S on valittujen ominaisuuksien joukko ja $I(X_k X_j; Y)$ kohteiden Y sekä yhteis-

muuttuja $X_k X_j$:n välinen informaatio. Ehdokkaana oleva ominaisuus X_k siis liitetään menetelmässä yhteen jokaisen aiemmin valitun ominaisuuden X_j kanssa.

Toinen esimerkki on CMIM (*conditional mutual info maximisation*), joka määritellään (Brown ym. 2012)

$$J_{cmim}(X_k) = \min_{X_j \in S} \left[I(X_k; Y | X_j) \right]. \tag{4.2}$$

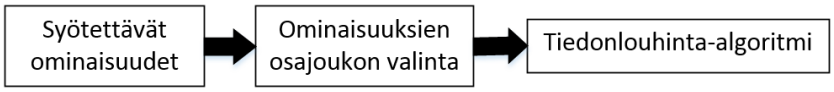
CMIM-algoritmi pyrkii välttämään keskenään samankaltaista informaatiota sisältäviä ominaisuuksia. Vaikka ehdokkaana oleva ominaisuus X_k itsenäisenä olisikin käyttökelpoinen, sitä ei valita mukaan mikäli se ei tuo riittävästi lisäinformaatiota valittuun ominaisuusjoukkoon S (Fleuret 2004).

Kolmantena esimerkkinä mainittakoon CIFE (*conditional informative feature extraction*). Se määritellään (Brown ym. 2012)

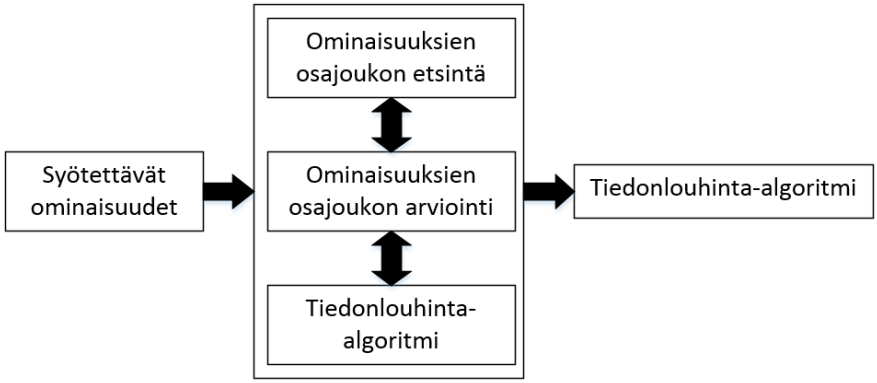
$$J_{cife}(X_k) = I(X_k; Y) - \sum_{j \in S} I(X_j; X_k) + \sum_{j \in S} I(X_j; X_k | Y). \tag{4.3}$$

CIFE-algoritmin kahden ominaisuuden välisessä vertailussa on olemassa luokittelulle merkityksellinen ja merkityksetön osa. Näistä huomioidaan ainoastaan luokittelulle merkityksellinen osa, josta vielä vähennetään kahden ominaisuuden välinen luokittelulle merkityksellinen redundanssi (*class-relevant redundancy*) (Lin ja Tang 2006).

- **Kääremallin** (*wrapper*) (ks. kuvio 9) mukaisessa ominaisuuksien valinnassa tiedonlouhinta-algoritmi ja ominaisuuksien valintaa suorittava esikäsittelijä vaihtavat keskenään tietoja koko valintaprosessin ajan (John, Kohavi ja Pfleger 1994). Käytännössä esikäsittelijä valitsee osajoukon ja lähettää sen arvioitavaksi tiedonlouhinta-algoritmille. Tätä jatketaan kunnes tarpeeksi hyvä osajoukko on löytynyt. Huonona puolena tällaisen mallin mukainen ominaisuuksien valinta voi viedä paljon enemmän aikaa, sillä tiedonlouhinta-algoritmi joutuu arvioimaan useamman osajoukon erikseen (Blum ja Langley 1997).
- **Hybridimallissa** (*hybrid*) on sekä suodatin- että kääremallin piirteitä. Tyypillisesti hybridimalli valitsee ensin monta ominaisuuksien osajoukkoa käyttäen ensin suodatinmallin mukaista ominaisuuksien valintaa. Nämä osajoukot ovat kandidaatteja jotka tiedonlouhinta-algoritmi käy läpi. Algoritmi valitsee tämän jälkeen testaamistaan osajoukoista sopivimman (Chen ym. 2006).



Kuvio 8: Suodatinmallin mukainen ominaisuuksien valinta (John, Kohavi ja Pfleger 1994).



Kuvio 9: Kääremallin mukainen ominaisuuksien valinta (John, Kohavi ja Pfleger 1994).

Ulottuvuuksien vähentämisellä (*dimensionality reduction*) viitataan menetelmiin joilla vähennetään datajoukon ulottuvuuksien, ts. ominaisuuksien, määrää. Yhteistä näille menetelmille on, että ne luovat kokonaan uusia ominaisuuksia yhdistelemällä alkuperäisessä datajoukossa olevaa tietoa, samalla vähentäen käytettyjen ominaisuuksien lukumäärää (Tan, Steinbach ja Kumar 2006, luku 2.3.3; Kantardzic 2011, luku 3.2.2). Eräs käytetyimmistä ulottuvuuksien vähentämisen menetelmistä on **pääkomponenttianalyysi** (*principal component analysis* eli *PCA*). PCA luo kokonaan uuden joukon muuttujia, joihin se pyrkii projisoimaan alkuperäisen datajoukon olennaiset ominaisuudet. Uusi muuttujajoukko pyrkii olemaan kooltaan alkuperäistä datajoukkoa pienempi (Kantardzic 2011, luku 2.5.3). PCA etsii alkuperäisten ominaisuuksien joukosta ne lineaarikombinaatiot, joilla saadaan aikaan suurin varianssi. Pääkomponenttianalyysin tuottama muuttujajoukko järjestetään varianssin mukaan. Tällöin ensimmäinen muodostettu pääkomponentti sisältää suurimman osan alkuperäisen datajoukon informaatiosta, toinen pääkomponentti toiseksi eniten informaatiota ja niin edelleen. Myöhemmässä analyysissä voidaan siis usein huomioida vain ensimmäiset pääkomponenttianalyysin komponenteista, sillä näiden jälkeen tulevat komponentit eivät tuo analyysiin enää merkittävästi lisää informaatiota (Wang ja Battiti 2006).

Wang ja Battiti (2006) määrittelevät pääkomponenttianalyysin tarkemmin seuraavalla taval-

la. Merkitään matriisilla $X_{n\times m}$ datajoukkoa, jossa on n instanssia ja m ominaisuutta:

$$X_{n\times m} = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \cdots & \cdots & x_{3m} \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}. \tag{4.4}$$

Pääkomponenttianalyysin laskennan pohjana käytetään kovarianssimatriisia C, joka määritellään

$$C = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T, \tag{4.5}$$

missä μ on keskiarvo $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$. Seuraavaksi kovarianssimatriisille C lasketaan ominaisarvot (*eigenvalues*) sekä ominaisvektorit (*eigenvectors*), mikä voidaan tehdä pääakselihajotelman (*singular value decomposition*) avulla. Oletetaan että kovarianssimatriista C laskettu $(\lambda_1, \mathrm{u}_1), (\lambda_2, \mathrm{u}_2), \cdots, (\lambda_m, \mathrm{u}_m)$ sisältää m kappaletta ominaisarvo- ja ominaisvektoripareja. Tällöin valitaan k kappaletta suurimmat ominaisarvot sisältävää ominaisvektoria. Aliavaruuden k ulottuvuuksien määrä määräytyy tällöin

$$\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{m}\lambda_i} \geq \alpha, \tag{4.6}$$

missä α on aliavaruuden variaation ja alkuperäisen avaruuden variaation suhde. Tämän jälkeen muodostetaan matriisi $U_{m\times k}$, ja projisoidaan data k-ulotteiseen aliavaruuteen seuraavasti:

$$y_i = U^T(x_i - \mu). \tag{4.7}$$

Näytteenotto (*sampling*) on datan vähentämisen menetelmä, jolla datajoukosta valitaan osajoukko analyysia varten. (Tan, Steinbach ja Kumar 2006, luku 2.3.2). Verkkoliikenteestä voidaan ottaa näytteitä joko verkkopaketeista tai vuopohjaisella tasolla (Bartos ja Rehak 2012). Yksinkertaisimmillaan näytteenotto verkkopaketeista tai -voista voidaan suorittaa täysin satunnaisesti (*random packet* ja *random flow sampling*). Muitakin metodeja hyödyntäviä verkkoliikenteen näytteenoton menetelmiä on kehitelty, mutta monet näistä menetelmistä eivät kunnolla sovellu tälle tutkimusalalle (Mai ym. 2006; Androulidakis ja Papavassiliou 2008).

Eräs esimerkki tutkimusalalle käyttökelpoisesta näytteenottomenetelmästä on kuitenkin valikoiva näytteenotto (*selective sampling*), joka keskittyy ottamaan näytteitä kooltaan pienistä verkkoliikennevoista (Bartos ja Rehak 2012; Androulidakis ja Papavassiliou 2008).

4.5.2 Datamuunnokset

Datamuunnoksilla (*data transformations*) datajoukon muuttujat valmistellaan sopivaan muotoon analyysia varten (Han ja Kamber 2006, luku 2.4.2). Datamuunnos-käsitteen alle voidaan sijoittaa muun muassa normalisointi, diskretisointi, aggregointi ja ominaisuuksien luonti.

Normalisoinnilla (*normalisation*) muuttujien arvot skaalataan tietylle vaihteluvälille (Kantardzic 2011, luku 2.3.1). Monet anomaliapohjaiset menetelmät käytännössä vaativat datajoukon muuttujien normalisoinnin ennen analyysiä, koska ilman normalisointia suuret muuttujan arvot saavat analyysissä tarpeettoman suuren painoarvon (Tan, Steinbach ja Kumar 2006, luku 2.3.7). Seuraavassa esitellään lyhyesti yleisimpiä IDS-järjestelmien tutkimusalalla käytettyjä normalisointimenetelmiä, joissa x_i tarkoittaa normalisoitua muuttujan arvoa ja v_i alkuperäistä muuttujan arvoa:

- **Minimi-maksimi** (*MinMax*) -menetelmää voidaan käyttää, kun tiedossa on muuttujajoukon pienin ja suurin arvo. Menetelmä muuntaa muuttujien arvot välille $[0,1]$ ja se määritellään
$$x_i = \frac{v_i - \min(v_i)}{\max(v_i) - \min(v_i)}, \tag{4.8}$$
missä $\min(v_i)$ on muuttujajoukon pienin arvo ja $\max(v_i)$ muuttujajoukon suurin arvo (Wang ym. 2009; Said ym. 2011).
- **Suuruusjärjestykseen perustuva** (*ordinal*) menetelmä antaa ensin kaikille muuttujan arvoille sijaluvun niiden suuruuden mukaan. Muuttujajoukon kaikkien arvojen pienin arvo saa sijaluvun 1, toiseksi pienin arvo sijaluvun 2 ja niin edelleen. Menetelmä määritellään
$$x_i = \frac{r_i - 1}{\max(r) - 1}, \tag{4.9}$$
missä r_i on muuttujan arvon sijaluku ja $\max(r)$ korkeimman sijaluvun arvo. Menetelmä siis muuntaa muuttujien arvot välille $[0,1]$ (Wang ym. 2009).
- **Suhteellinen** (*frequency*) menetelmä määrittää muuttujalle arvon suhteessa siihen, mi-

ten suuren osan yksittäinen arvo muodostaa koko n muuttujaa sisältävän muuttujajoukon yhteenlasketuista arvoista. Menetelmä määritellään

$$x_i = \frac{v_i}{\sum_{i=1}^{n} v_i} \tag{4.10}$$

ja se muuntaa muuttujien arvot välille $[0,1]$ (Wang ym. 2009).

- **Z-piste** (*z-score*) -menetelmän käytön edellytyksenä mainitaan, että muuttujajoukon arvojen tulisi noudattaa normaalijakaumaa. Menetelmällä voidaan muuntaa mikä tahansa tällainen muuttujajoukko noudattamaan standardoitua normaalijakaumaa. Muuttujajoukon arvojen keskiarvo on täten muunnoksen jälkeen 0 ja keskihajonta 1. Muunnoksen jälkeen siis suurin osa muuttujien arvoista on korkeintaan keskihajonnan päässä keskiarvosta. Menetelmä määritellään

$$x_i = \frac{v_i - \mu}{\sigma}, \tag{4.11}$$

missä μ on n:stä kappaleesta muuttujia koostuvan muuttujajoukon arvojen keskiarvo $\mu = \frac{1}{n}\sum_{i=1}^{n} v_i$ ja σ niiden keskihajonta $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - \mu)}$ (Wang ym. 2009; Said ym. 2011). Normalisointimenetelmien vertailussa, jonka Wang ym. (2009) tekivät KDD Cup 99 -datajoukkoa käyttäen, pärjäsi z-piste-menetelmä parhaiten. Muut vertailussa olleet menetelmät olivat minimi-maksimi-menetelmä, suuruusjärjestykseen perustuva menetelmä sekä suhteellinen normalisointimenetelmä.

- **Logaritminen** menetelmä (Said ym. 2011) on yksinkertainen logaritmia hyödyntävä normalisointimenetelmä. Se määritellään

$$x_i = \log(1 + v_i). \tag{4.12}$$

Said ym. (2011) havaitsivat normalisointimenetelmien vertailussaan logaritmisen menetelmän toimivan parhaiten. Muut vertailussa olleet menetelmät olivat z-piste-menetelmä sekä minimi-maksimi-menetelmä. He käyttivät etäisyyspohjaisia menetelmiä löytääkseen haitallisen liikenteen KDD Cup 99 -datajoukosta.

Diskretisoinnilla (*discretisation*) muunnetaan jatkuvat muuttujien arvot (ks. luku 4.4.2) diskreeteiksi arvoiksi. Arvojen muuntaminen diskreeteiksi voi hyödyttää prosessia monella tavalla. Ensinnäkin monet luokittelijoihin perustuvat anomaliapohjaiset menetelmät voivat hyödyntää ainoastaan diskreettiä dataa. Muunnoksessa data saadaan myös usein helpommin

esitettävään ja ymmärrettävään muotoon. Tämä taas johtaa siihen, että analyysissä saadut tulokset ovat yleensä helpommin tulkittavia ja käyttökelpoisempia (Liu ym. 2002). Myös suodatinmallin mukainen ominaisuuksien valinta (ks. luku 4.5.1) toimii tehokkaimmin diskreetillä datalla (Bolón-Canedo, Sánchez-Maroño ja Alonso-Betanzos 2009). Diskretisointivaiheessa kuitenkin häviää lähes väistämättä osa alkuperäisen datan informaatiosisällöstä. Päämääränä onkin minimoida muunnoksessa aiheutuva informaatiohäviö (García ym. 2013).

García ym. (2013) listasivat diskretisointimenetelmien jaottelussaan 87 menetelmää, joista he vertailivat 30:tä. Diskretisointimenetelmät voidaan heidän mukaansa ryhmitellä staattisiin tai dynaamisiin, yksi- tai moniulotteisiin, ohjattuihin tai ohjaamattomiin, jakaviin tai yhdistäviin, globaaleihin tai paikallisiin ja suoraviivaisiin tai inkrementaalisiin:

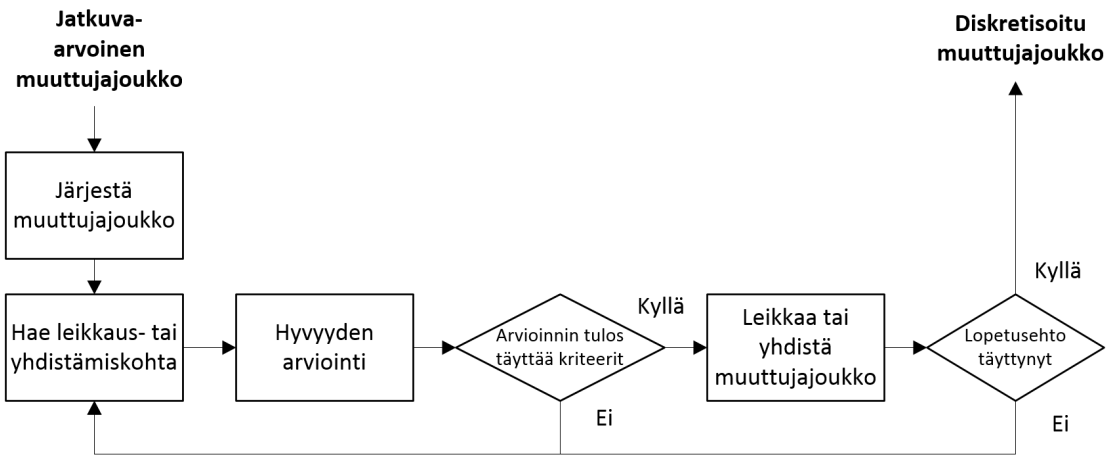
- **Staattinen** (*static*) ja **dynaaminen** (*dynamic*): Staattinen menetelmä diskretisoi muuttujien arvot itsenäisesti, eli ei hyödynnä valinnan aikana millään tavalla varsinaista tiedonlouhinta-algoritmia. Staattisen diskretisoinnin jälkeen data toimitetaan tiedonlouhinta-algoritmille, joka ei voi enää tehdä muutoksia diskretisointiin. Dynaaminen diskretisointi sen sijaan pyrkii hyödyntämään diskretisointiprosessin aikana tiedonlouhinta-algoritmia. Dynaamisia diskretisointimenetelmiä on olemassa melko vähän, ja ne ovat silloinkin usein kiinteänä osana tiedonlouhinta-algoritmeja (García ym. 2013; Liu ym. 2002).
- **Yksiulotteinen** (*univariate*) ja **moniulotteinen** (*multivariate*): Yksiulotteinen menetelmä ottaa huomioon ainoastaan yhden muuttujan arvon kerrallaan, ja jo käsiteltyyn muuttujaan ei enää missään vaiheessa palata. Sen sijaan moniulotteinen menetelmä huomioi jokaisen muuttujan arvon kohdalla muutkin datajoukon muuttujien arvot (García ym. 2013; Liu ym. 2002).
- **Ohjattu** (*supervised*) ja **ohjaamaton** (*unsupervised*): Ohjatussa diskretisoinnissa käytetään datajoukon nimiöintiä (ks. luku 4.3) apuna datan jakamisessa sopiviin ryhmiin. Ohjaamattomalla menetelmällä voidaan diskretisoida myös nimiöimätöntä dataa. Suurin osa olemassa olevista diskretisointimenetelmistä on ohjattuja (García ym. 2013; Liu ym. 2002).
- **Jakava** (*splitting*) ja **yhdistävä** (*merging*): Jakavat diskretisointimenetelmät aloittavat diskretisoinnin tilanteesta, jossa yhtään leikkauskohtaa datan jakamista varten ei ole

määritelty. Menetelmä laskee datan pohjalta aina yhden tai useamman leikkauskohdan lisää, ja lopettaa jakamisen kun määritelty lopetusehto täyttyy. Yhdistävät diskretisointimenetelmät aloittavat diskretisoinnin päinvastaisesta tilanteesta, jossa leikkauskohtia on määritelty maksimimäärä. Leikkauskohtia poistetaan tämän jälkeen vastaavasti yksi tai useampi kerrallaan, kunnes lopetusehto täyttyy (García ym. 2013).

- **Globaali** (*global*) ja **paikallinen** (*local*): Globaali menetelmä ottaa diskretisoinnin aikana jatkuvasti huomioon kaikki muuttujat leikkaus- tai yhdistämiskohtia ratkaistessaan. Paikallinen diskretisointi taas huomioi ainoastaan yhden muuttujan kerrallaan (Chmielewski ja Grzymala-Busse 1996).
- **Suoraviivainen** (*direct*) ja **inkrementaalinen** (*incremental*): Suoraviivainen diskretisointi vaatii jollakin tavalla etukäteen määritellyn parametrin k, joka määrää kuinka moneen ryhmään diskretisoitava data jaetaan. Menetelmä jakaa datan parametrin määrittelyn jälkeen samanaikaisesti ryhmiin, joita on k kappaletta. Inkrementaalinen menetelmä sitä vastoin tuottaa datasta ensin pelkistetyn diskretisoinnin, jota parannellaan seuraavilla diskretisointikierroksilla (García ym. 2013; Liu ym. 2002).

Tyypillisesti diskretisointiprosessi (ks. kuvio 10) alkaa ominaisuuden jatkuvien muuttujien arvojen järjestämisellä pienimmästä arvosta suurimpaan tai päinvastoin. Riippuen siitä onko diskretisointimenetelmä jakava vai yhdistävä, valitaan seuraavaksi ehdokas joko leikkaus- tai yhdistämiskohdaksi. Tämän jälkeen ehdokkaan hyvyys arvioidaan menetelmän määrittelemällä tavalla. Jos ehdokas ei täytä vaadittuja ehtoja, valitaan uusi ehdokas. Kun ehdokas on arvioitu riittävän hyväksi, suoritetaan datan leikkaus tai yhdistäminen aiemmin valitun ja hyväksytyn ehdotuksen pohjalta. Tämän jälkeen tarkistetaan, onko diskretisointiprosessin lopetusehto täyttynyt. Yksinkertaisimmillaan lopetusehto voi esimerkiksi määritellä, että diskretisointi lopetetaan kun kategorioita on muodostettu k kappaletta. Ellei lopetusehto ole täyttynyt, siirrytään prosessin alkuun etsimään uutta leikkaus- tai yhdistämiskohtaa (Liu ym. 2002).

Diskretisoinnin hyvyyttä voidaan siis arvioida jatkuvasti myös prosessin aikana. Arvioinnissa voidaan huomioida ainakin neljä ominaisuutta: kategorioiden määrä, epäjohdonmukaisuus (*inconsistency*), osuvuus ja nopeus. Diskretisoinnin tavoitteena on sisällyttää alkuperäisen datan informaatiosisältö mahdollisimman pieneen määrään kategorioita. Epäjohdon-



Kuvio 10: Tyypillinen diskretisointiprosessi (Liu ym. 2002).

mukaisuudella tarkoitetaan tilannetta, jossa kahdella datajoukon instanssilla on täsmälleen samat muuttujien arvot, mutta nimiöinnin perusteella ne kuitenkin edustavat eri ryhmiä. Tällaiset tilanteet pyritään diskretisointivaiheessa minimoimaan. Diskretisointi on tarkkuudeltaan ihanteellinen, kun tiedonlouhinta-algoritmin suorituskykyä vertaillaan diskretisoidulla sekä ennen diskretisointia olevalla datalla, ja tarkkuudet ovat lähellä toisiaan. Myös diskretisoinnin aikavaatimukset on usein syytä ottaa huomioon. (García ym. 2013; Liu ym. 2002).

Aggregointi (*aggregation*) on datamuunnos, jolla pyritään yhdistämään kahden tai useamman muuttujan olennainen informaatiosisältö yhteen muuttujaan. Muunnoksella voidaan saada useita eri hyötyjä. Ensinnäkin muunnoksella saadaan datajoukkoa pienennettyä. Aggregointi vaikuttaa myös datan esitystapaan, jota voidaan muunnoksella saada selkeämmäksi, hieman samoin kuin diskretisointi saattaa selkeyttää datan esitystä (Tan, Steinbach ja Kumar 2006, luku 2.3.1).

Ominaisuuksien luonti (*feature construction* tai *feature generation*) on läheisesti aggregointia muistuttava datamuunnos. Siinä luodaan uusi ominaisuus yhden tai useamman ominaisuuden pohjalta (Tan, Steinbach ja Kumar 2006, luku 2.3.5). Yksinkertaisimmillaan luonti voi olla esimerkiksi nominaalisten muuttujien muuttaminen binäärisiksi (Weller-Fahy, Borghetti ja Sodemann 2015).

4.6 Etäisyyden ja samankaltaisuuden mittarit

Anomaliapohjaiset NIDS-järjestelmät perustuvat tavalla tai toisella siihen, että tulevaa verkkoliikennettä verrataan aiemmin luotuihin profiileihin. Tässä vertailussa olennaisena osana ovat mittarit, joilla määritellään kuinka samankaltaista analysoitava data on aiemmin nähdyn verkkoliikenteen kanssa (Weller-Fahy, Borghetti ja Sodemann 2015). Mittarin käyttökelpoisuus vaihtelee tilanteen mukaan. Eräs mittarien valintaa rajoittava tekijä on datan muuttujien tyyppi, sillä numeeriselle ja kategoriselle datalle on olemassa omat mittarinsa. Lisäksi jotkut mittarit on suunniteltu käytettäväksi molempia datatyyppejä sisältävään dataan (Bhuyan, Bhattacharyya ja Kalita 2014; Boriah, Chandola ja Kumar 2008)

Etäisyyden ja samankaltaisuuden mittareita on hyödynnetty NIDS-järjestelmissä yleisesti kahdessa eri vaiheessa. Ensimmäinen, harvemmin nähty, käyttötapa on hyödyntää todennäköisyyksiin liittyviä mittareita optimaalisten ominaisuuksien etsinnässä ominaisuuksien valinnan (ks. luku 4.5.1) aikana. Yleisimmin mittareita kuitenkin käytetään datan luokittelussa ja klusteroinnissa. Luokittelu on luonteeltaan ohjattua (ks. luku 4.3), eli se vaatii etukäteistiedon olemassa olevista luokista. Klusterointia sen sijaan voidaan käyttää ohjaamattomasti, ja sillä voidaan saada selville tietoa datan ominaisuuksista, esimerkiksi luokkien määrä ja kunkin luokan ominaispiirteet (Weller-Fahy, Borghetti ja Sodemann 2015).

Deza ja Deza (2009) kokosivat ja jaottelivat teoksessaan etäisyyksiin liittyvää terminologiaa. Teoksen pohjalta jakoivat Weller-Fahy, Borghetti ja Sodemann (2015) anomaliapohjaisten NIDS-järjestelmien kannalta olennaisimmat mittarit neljään pääryhmään, jotka ovat (p,r)-potensseihin perustuvat, todennäköisyysjakaumille perustuvat, korrelaatiopohjaiset sekä muut samankaltaisuuden ja etäisyyden mittarit:

- **(p,r)-potensseihin perustuvien** (*power distances*) mittarien perusmuoto on kahta instanssia eli vektoria vertailtaessa

$$d = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{r}}, \tag{4.13}$$

  missä n on instanssien ominaisuuksien määrä eli vektorin ulottuvuus, x ensimmäinen instanssi ja y toinen instanssi. Parametreja p ja r säätämällä määritellään varsinainen käytettävä mittari. Esimerkiksi kun $p = r = 2$, saadaan eräs yleisimmistä etäisyysmit-

tareista, Euklidinen etäisyys (*Euclidean distance*), joka siis määritellään (Cha 2007)

$$d_{euc} = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}. \tag{4.14}$$

Euklidinen etäisyys toimii yleensä parhaiten kaksi- tai kolmiulotteisella datalla, mutta sitä voidaan käyttää myös moniulotteiselle datalle. Ulottuvuuksien määrän kasvaessa muut mittarit saattavat toimia paremmin. Esimerkkinä tällaisesta mittarista Weller-Fahy, Borghetti ja Sodemann (2015) sekä Deza ja Deza (2009, luku 17.2) mainitsevat *fractional* l_p -etäisyyden, jolloin kaavassa 4.13 on $0 < p = r < 1$.

- **Todennäköisyysjakaumille perustuvat** (*distances on distribution laws*) mittarit sisältävät useimmat entropiapohjaiset mittarit sekä ehdollisille todennäköisyysjakaumille perustuvat mittarit.
- **Korrelaatiopohjaiset** (*correlation similarities*) samankaltaisuuden ja etäisyyden mittarit pyrkivät luonnehtimaan kahden datajoukon välisen korrelaation, ja käyttämään sitä mittarina etäisyyksissä tai samankaltaisuuksissa. Esimerkkejä korrelaatiopohjaisista etäisyysmittareista ovat kosinietäisyys (*cosine distance*), joka määritellään (Deza ja Deza 2009, luku 17.4; Weller-Fahy, Borghetti ja Sodemann 2015)

$$d_{cos} = 1 - cos(\theta) = 1 - \frac{x \cdot y}{\|x\| \, \|y\|} = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{4.15}$$

sekä Pearsonin korrelaatioetäisyys (*Pearson correlation distance*), joka määritellään (Deza ja Deza 2009, luku 17.4; Weller-Fahy, Borghetti ja Sodemann 2015)

$$d_{cor} = 1 - \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{(\sum_{i=1}^{n} (x_i - \bar{x})^2)(\sum_{i=1}^{n} (y_i - \bar{y})^2)}}, \tag{4.16}$$

missä $\bar{x}$ on ensimmäisen instanssin ominaisuuden arvojen keskiarvo ja $\bar{y}$ vastaavasti toisen instanssin ominaisuuden arvojen keskiarvo.

- **Muut**-ryhmän alle Weller-Fahy, Borghetti ja Sodemann (2015) laskevat kuuluvaksi mittarit, jotka eivät sovellu yllä esiteltyihin ryhmiin, mutta joille löytyy käyttötarkoituksia tällä tutkimusalueella. Yhtenä esimerkkinä tällaisesta on Dice-koeffisientti eli -etäisyys, joka käyttää laskennan pohjana n-gram-analyysiä.

4.7 Anomaliapohjaisten menetelmien soveltuvuus tutkimusalalle

Anomaliapohjaisten menetelmien soveltuvuutta tutkimusalalle voidaan käsitellä monesta näkökulmasta. Ensinnäkin tutkimusalan taustalla vaikuttavia perimmäisiä haasteita esitellään luvussa 4.7.1. Toisenlainen ongelma ovat tutkimusalalla vallitsevat tutkimus- ja raportointikäytänteet, joiden ongelmia esitellään luvussa 4.7.2. Tutkimusalalla käytettyihin datajoukkoihin liittyviä haasteita esitellään luvussa 4.7.3.

4.7.1 Perimmäiset haasteet

Axelsson (2000) osoitti, että anomaliapohjaisen IDS-järjestelmän suorituskykyä rajoittava tekijä on järjestelmän kykenemättömyys tukahduttaa väärät hälytykset. Tämän osoittamiseen Axelsson käytti Bayesiläisen tilastotieteen esiintyvyysharha (*base rate fallacy*) -käsitettä. Kun hälytysten tulkinta jätetään niin sanotun ihmisanalyytikon harteille, on tehokkaasti käsiteltävissä oleva hälytysten määrä hyvin alhainen. Axelsson viittaa prosessiautomaation ja tehtaanhallinnan aloilla tehtyihin tutkimuksiin, joissa tutkittiin työntekijöiden reagointia näiden alojen järjestelmien tekemiin automaattisiin hälytyksiin. Näiden tutkimusten mukaan jo tilanne, jossa joka toinen hälytys on aiheeton lannistaa suuren osan ihmisistä. Tällöin kaikkiin tuleviin hälytyksiin suhtaudutaan hyvin skeptisesti. Asiantuntijankin on usein vaikea päätellä mistä järjestelmän tekemä hälytys johtuu, ja tulisiko hälytykseen reagoida. Havaitsemattomalla verkkohyökkäyksellä voi kuitenkin olla vakavat seuraukset.

Sommer ja Paxson (2010) vertailevat väärän luokittelun seurauksia muilla käyttöalueilla. Vastaavia tiedonlouhinnan tai koneoppimisen menetelmiä hyödynnetään kaupallisesti muun muassa verkkokauppojen tuotesuositteluissa, tekstintunnistuksessa (*optical character recognition* eli *OCR*) ja roskapostin tunnistuksessa. Väärä tuotesuositus tai oikean sähköpostin luokittelu roskapostiksi ei kuitenkaan tee näitä järjestelmiä käyttökelvottomiksi. Tekstintunnistuksessa voidaan taas väärin tunnistettu teksti usein korjata oikeaksi automaattisilla oikolukutyökaluilla. Verkkoliikenteen tunnistuksen käyttöalueella ongelmat ovat suurempia datan luonteen vuoksi. Virheellisiä luokitteluja voi olla paljon vaikeampi tulkita ja hälytysten läpikäynnin automatisointi on hankalaa.

Sommer ja Paxson (2010) esittävät myös, että koneoppimisen menetelmien vahvuudet ovat

aiemmin nähdyn toiminnan kaltaisen uuden toiminnan havaitseminen. Järjestelmä tulisi heidän mukaansa kouluttaa tunnistamaan kaikki sellainen toiminta, jota sen halutaan löytävän. IDS-järjestelmän tapauksessa tämä tarkoittaisi kouluttamista kaikilla tiedetyillä hyökkäyksillä. Samalla tämä tosin tarkoittaa, että täysin uudenlainen toiminta – jonka havaitsemista yleisesti pidetään anomaliapohjaisen järjestelmän kenties suurimpana vahvuutena – jää todennäköisesti järjestelmältä huomaamatta. Tällaista lähestymistapaa nähdään alan tutkimuksissa heidän mukaansa kuitenkin liian harvoin, sillä useimmiten pyritään kirjaimellisesti löytämään anomaliat; normaalin profiilin ulkopuolella oleva toiminta. He näkevätkin tärkeimpänä askeleena anomaliapohjaisten IDS-järjestelmien suorituskyvyn parantamisessa järjestelmien käyttöalueen tarkemman rajaamisen. Samalla tulisi heidän mukaansa hyväksyä, että kaikkia mahdollisia hyökkäyksiä ei ole edes mahdollista havaita.

4.7.2 Tutkimus- ja raportointikäytänteet

Tavallaee, Stakhanova ja Ghorbani (2010) havaitsivat, että suurin osa heidän tutkimuksessaan mukana olleista vuosina 2000–2008 julkaistuista artikkeleista ei täyttänyt tieteellisen tarkkuuden vaatimuksia. Puutteita oli ensinnäkin datan esikäsittelyn (ks. luku 4.5) raportoinnissa. Lähes neljä viidestä tutkimuksessa mukana olleista artikkeleista jätti selostamatta käytetyt normalisoinnin toimenpiteet. Automatisoitua ominaisuuksien valintaa käytettiin vain alle neljänneksessä artikkeleista, ja näistäkin vain alle puolet selosti käytetyn ominaisuuksien valinnan mekanismin. Jopa 42 % artikkeleista ei suorittanut minkäänlaista ominaisuuksien valintaa, vaan käytti kaikkia ominaisuuksia. Läheskään kaikki artikkelit eivät nimenneet tutkimukseen lopulta päätyneitä datajoukon ominaisuuksia.

Toinen ongelmallinen vaihe, jonka Tavallaee, Stakhanova ja Ghorbani (2010) havaitsivat, oli varsinaisen kokeen asettelu. Tässä tutkimuksen vaiheessa tulisi heidän mukaansa asianmukaisesti määritellä, mitä datajoukon osajoukkoja käytetään tutkimuksessa opetusdatana sekä testausdatana. Olisi hyvä kiinnittää huomiota myös testausdatan normaalin ja haitallisen liikenteen suhteeseen. Vain noin viidesosa tutkimuksen artikkeleista ilmoitti tämän suhteen, ja vaihtelevuus suhdeluvuissa tutkimusten välillä oli suuri. Suhteen ilmoittaneista artikkeleista reilusti yli puolet käytti testausdatana datajoukkoa, jossa haitallisen liikenteen määrä oli peräti 80–82 %. Toiseksi suurin joukko, noin neljännes, artikkeleista käytti testausdatana da-

tajoukkoa, jossa haitallisen liikenteen määrä oli vain 1–2 %. Voidaan melko varmasti todeta, että kumpikaan näistä ääripäistä ei edusta keskimääräisen, normaalisti toimivan, verkkoympäristön liikennejakaumaa. Kokeen asetteluvaiheessa yleinen ongelma oli myös valmiiden työkalujen ja algoritmien alkuparametrien raportoimatta jättäminen. Kokeen parametrien valintaa ylipäänsä pyrittiin perustelemaan vain reilussa kymmenesosassa artikkeleista.

Muita tutkimuksessa havaittuja puutteita olivat kokeiden tulosten validointi, esittäminen ja menetelmän suorituskyvyn arviointi. Tulosten validointi yleensä tarkoittaa, että koe toistetaan asiaankuuluvalla tavalla tarpeeksi monta kertaa, jolloin lopullisiksi tuloksiksi saadaan näiden kokeiden keskiarvot. Tätä menetelmää kutsutaan koneoppimisessa ristiinvalidoinniksi (*cross-validation*). Kuitenkin vain viidesosa artikkeleista käsitteli tulosten luotettavuuden varmistusta ja validointia. Tulosten esittämisessä on tärkeää ilmoittaa selkeällä tavalla sellaiset mittaustulokset, jotka edesauttavat tutkimuksen tulosten yksiselitteistä tulkintaa. Lisäksi mittaustulokset tulisi ilmoittaa jokaisen haittaliikennetyypin kohdalla erikseen, sillä tutkimuksessa käytetty menetelmä voi havaita täydellisesti tietyn tyypin haittaliikenteen, mutta siltä voi mennä täysin ohi muu haittaliikenne. Suorituskyvyn arvioinnissa, jota suoritettiin vain alle viidesosassa tutkimuksista, tulisi huomioida menetelmän järjestelmälle aiheuttama yleisrasite ja aikavaatimukset.

Weller-Fahy, Borghetti ja Sodemann (2015) valitsivat etäisyyden ja samankaltaisuuden mittareita (ks. luku 4.6) käsittelevään tutkimukseensa sata satunnaista NIDS-järjestelmiin liittyvää tieteellistä artikkelia. Otoksen pohjalta ensimmäinen ongelma heidän mukaansa on useimpien artikkeleiden tapa jättää käytetty etäisyysmittari kokonaan nimeämättä. Vain 40 prosenttia tutkimuksen artikkeleista nimesi käytetyn etäisyysmittarin. Vielä harvempi artikkeli esitti matemaattisesti muotoillun määritelmän käytetylle etäisyysmittarille. Joissakin artikkeleissa oli käytetty ilman tarkempaa määrittelyä myös algoritmien oletusasetuksia, joiden parametrit voivat vaihdella versiosta toiseen, ja jotka voivat myös soveltua huonosti tarkasteltavaan ongelmaan. Useimmat artikkelit jättivät myös perustelematta miksi juuri kyseistä etäisyysmittaria oli käytetty tutkimuksessa. Viimeisenä ongelmana he mainitsevat vaihtoehtoisten etäisyysmittareiden kokonaan tarkastelematta jättämisen. Vain kaksi artikkelia tutkituista huomioi riittävällä tavalla vaihtoehtoiset etäisyysmittarit. Mikäli tutkimuksessa käytetään ainoastaan yhtä etäisyysmittaria, heidän mukaansa olisi hyvä tutkia vähintään vaihtoeh-

toisia parametreja ja esittää niiden vaikutukset tutkimustuloksissa.

Edellä mainitut raportoinnin, datan esikäsittelyvaiheen, kokeen asettelun, tulosten validoinnin ja esityksen sekä suorituskyvyn arvioinnin puutteet vaikuttavat suoraan tutkimuksen luotettavuuteen ja pätevyyteen. Puutteellisen raportoinnin vuoksi on myös kokeen toistaminen ulkopuolisten toimesta usein lähes mahdotonta.

4.7.3 Datajoukkojen haasteet

Tutkimusalalla on pitkään ollut ongelmana menetelmien harjoittamiseen ja testaukseen sopivien datajoukkojen puute (Abt ja Baier 2014b). Aidosta tuotantokäytössä olevasta verkosta on toki mahdollista kaapata dataa tällaiseen käyttöön – ainakin jos lainsäädännön asettamat rajoitteet sivuutetaan – mutta ongelmaksi tulee tällöinkin niin kutsutun perustotuuden (*ground truth*) saavuttaminen. Perustotuus tarkoittaa, että datajoukon sisältämä haittaliikenne on täysin aukottomasti tiedossa. Jos varmuutta perustotuudesta ei ole, on datajoukolla mahdotonta luotettavasti ja kattavasti mitata menetelmien toimivuutta. Perustotuuden saavuttaminen aidosta tuotantoympäristöstä kaapatulle datalle vaatisi siis asiantuntijan suorittamaa manuaalista datan läpikäyntiä, ja lisäksi tämä asiantuntija ei saisi tehdä yhtäkään virhettä datan luokittelun aikana. Ellei kyseessä ole hyvin pieni datajoukko, on perustotuuden saavuttaminen tällä tavalla käytännössä mahdotonta (Ringberg, Roughan ja Rexford 2008). Sommer ja Paxson (2010) pitävät varmuutta perustotuudesta niin tärkeänä, että ilman sitä on koko tutkimuksen jatkaminen kyseenalaista.

Tuotantokäytössä olevasta verkosta kaapatun datan käytössä on toinenkin ongelma. Datajoukkoa ei käytännössä voi jakaa suoraan muiden tutkijoiden käyttöön ainakaan ilman anonymisointia, jonka luotettava toteuttaminen taas on haastavaa ja riskialtista (Abt ja Baier 2014a). Abt ja Baier (2014b) kuitenkin myös huomauttavat, että tutkimuksen pohjana käytetyn datan jakamatta jättäminen muiden tutkijoiden käyttöön on tieteen perusperiaatteiden vastaista.

Aidon verkkoympäristön datajoukkojen haasteiden vuoksi tutkimusalalla käytetäänkin usein simuloituja eli synteettisiä datajoukkoja. Simuloitu datajoukko voidaan luoda eristetyssä ja täysin kontrolloidussa verkkoympäristössä, jossa perustotuuden saavuttaminen on periaat-

teessa vaivatonta. Tutkimusalalla edelleen käytetyimmät simuloidut datajoukot ovat jo yli 15 vuoden ikäiset DARPA- sekä KDD Cup 99 -datajoukot (Tavallaee, Stakhanova ja Ghorbani 2010; Ahmed, Mahmood ja Hu 2016). Simuloituja datajoukkoja, etenkin vanhoja ja käytetyimpiä, toisaalta taas kritisoidaan usein siitä, että ne eivät vastaa todellisen ja nykyaikaisen verkkoympäristön liikennettä, niissä käytetyt hyökkäykset ovat vanhentuneita sekä opetus- ja testausdatajoukon liikenteen jakaumat poikkeavat toisistaan (Thomas, Sharma ja Balakrishnan 2008; Tavallaee ym. 2009). Näitä ongelmia ovat myöhemmin julkaistut simuloidut datajoukot pyrkineet korjaamaan. Viimeisimpien joukossa tällaisen nykyaikaisen simuloidun datajoukon, nimeltään UNSW-NB15, ovat koonneet ja julkaisseet Moustafa ja Slay (2015).

4.8 Luvun yhteenveto

Luvussa käytiin läpi anomalian määrittelyä, esiteltiin anomaliapohjaisten menetelmien taustaa ja jaottelua, anomalioiden ryhmittelyä ja havaitsemistekniikoita, datajoukkoja ja datan lajeja, datan esikäsittelyä, etäisyyden ja samankaltaisuuden mittareita sekä pohdittiin anomaliapohjaisten menetelmien soveltuvuutta tutkimusalalle. Luvun pääsisältö voidaan tiivistää seuraavasti:

- Anomalia on määritelmällisesti "jotakin epätavallista tai odottamatonta"tai "poikkeama yleisestä säännöstä".
- Anomaliapohjaiset menetelmät voidaan jakaa kuuteen pääryhmään: tilastolliset, luokittelupohjaiset, klusterointi- ja poikkeava havainto -pohjaiset, pehmeän laskennan, tietämyspohjaiset sekä yhdistelevät menetelmät.
- Datassa esiintyvät anomaliat voidaan jakaa pistemäisiin, kontekstuaalisiin ja kollektiivisiin anomalioihin.
- Anomaliapohjaiset menetelmät voidaan edelleen jakaa ohjattuihin, puoliohjattuihin ja ohjaamattomiin tekniikoihin.
- Analysoinnin kohteena oleva data voi olla rakenteellista, puolirakenteellista tai rakenteetonta.
- Datassa olevat muuttujat voidaan jakaa kategorisiin ja numeerisiin muuttujiin, tai vaihtoehtoisesti diskreetteihin ja jatkuviin muuttujiin. Kategoriset muuttujat ovat joko nominaalisia tai ordinaalisia. Numeeriset muuttujat käyttävät joko välimatka-asteikkoa

tai suhdeasteikkoa.

- Datan esikäsittelyllä saatetaan analysoitava data valmiiksi analyysin suorittavalle tiedonlouhinta-algoritmille. Esikäsittelyn menetelmiä ovat datan siivous, yhdistäminen, vähentäminen sekä datamuunnokset. Datan vähentämisen menetelmiä ovat ominaisuuksien valinta, ulottuvuuksien vähentäminen sekä näytteenotto. Datamuunnoksia ovat normalisointi, diskretisointi, aggregointi ja ominaisuuksien luonti.
- Etäisyyden ja samankaltaisuuden mittareilla määritellään, kuinka samankaltaista analysoitava data on aiemmin nähdyn datan kanssa.
- Anomaliapohjaisen IDS-järjestelmän suorituskykyä rajoittava tekijä on järjestelmän kykenemättömyys tukahduttaa väärät hälytykset.
- Tutkimusalan tutkimus- ja raportointikäytänteissä on yleisesti ongelmia ja puutteita, jotka vaikuttavat tutkimusten luotettavuuteen ja pätevyyteen.
- Tutkimusalalla on puute tutkimukseen soveltuvista verkkoliikenteen datajoukoista, ja erityisesti haasteena on ns. perustotuuden saavuttaminen tuotantoympäristöstä kaapatulle datalle.

5 Menetelmien valinta ja käsiteltävä data

Luvussa 4.7.3 esiteltyjen olemassa olevien suosituimpien synteettisten ja aitojen datajoukkojen ongelmakohtien vuoksi tässä tutkielmassa päädyttiin käyttämään synteettistä vuonna 2015 julkaistua UNSW-NB15-datajoukkoa (Moustafa ja Slay 2015). Tämä datajoukko pyrkii korjaamaan aiemmissa datajoukoissa ilmenneitä ongelmia, joista yksi on ajan tasalla olevien hyökkäysten puute.

5.1 Datajoukon kuvaus

UNSW-NB15-datajoukon verkkoliikenne on luotu Australian kyberturvallisuuskeskuksessa (*Australian centre for cyber security* eli *ACCS*) IXIA PerfectStorm -testausalustan avulla. IXIA sisältää jatkuvasti päivittyvän tietokannan, johon uudet hyökkäykset lisätään ulkopuolisesta CVE (common vulnerabilities and exposures) -tietokannasta. Alustalla voidaan siis periaatteessa generoida sekä normaalia harmitonta verkkoliikennettä että nykyaikaista haittaliikennettä. Datajoukon tekijät kaappasivat Tcpdump-työkalulla IXIA-alustan generoiman verkkoliikenteen, jota jatkokäsiteltiin kolmella tavalla. Tcpdumpin luomista pcap-tiedostoista tekijät koostivat lopulliset ominaisuudet Argus- ja Bro-IDS-ohjelmien avulla. Lisäksi he loivat kaksitoista lisäominaisuutta itse luomillaan algoritmeilla (Moustafa ja Slay 2015).

Tässä tutkielmassa käytetään sivustolta Moustafa ja Slay (2016b) ladattuja opetusdatan ja testausdatan tiedostoja. Tiedostojen nimissä ja datajoukon tilastollisia ominaisuuksia käsittelevässä artikkelissa oli kuitenkin ristiriita, joka havaittiin vasta tutkielman teon loppuvaiheessa. Artikkelin mukaan nimittäin opetusdatajoukossa tulisi olla 175 341 instanssia ja testausdatajoukossa 82 332 instanssia. Tiedosto *UNSW_NB15_training-set.csv* kuitenkin sisälsi 82 332 instanssia ja tiedosto *UNSW_NB15_testing-set.csv* 175 341 instanssia[1]. Tätä tiedostojen alkuperäistä jakoa noudatetaan tässä tutkielmassa.

Yhteensä ominaisuuksia on datajoukossa 47. Lisäksi on kaksi nimiöintiin liittyvää ominai-

1. Tilanne tarkastettiin 15.11.2016, ja tuolloin sivustolta ladatut tiedostot noudattivat tätä jakoa. Moustafa vahvisti sähköpostiviestissään 22.11.2016, että datajoukon web-sivun tiedostot oli virheellisesti nimetty ristiin.

suutta, joista ensimmäinen ilmaisee mitä yhdeksästä hyökkäysryhmästä instanssi edustaa. Toinen ominaisuus sisältää yksinkertaisen nimiöinnin, jossa luku 0 tarkoittaa normaalin liikenteen instanssia ja luku 1 haittaliikennettä. Datajoukko sisältää myös 82 332 instanssia sisältävän valmiin opetusdatan tiedostossa *UNSW_NB15_training-set.csv* sekä 175 341 instanssia sisältävän testausdatan tiedostossa *UNSW_NB15_testing-set.csv* (Moustafa ja Slay 2015). Tässä tutkielmassa hyödynnetään vain näiden kahden tiedoston sisältämää dataa.

Datajoukon 47 ominaisuutta on jaettu viiteen ryhmään (Moustafa ja Slay 2016a, 2015):

- **Vuo-ominaisuudet** (*flow features*) sisältävät isäntien välisen liikenteen tunnistetiedot eli IP-osoitteet ja porttinumerot sekä liikennöinnissä käytetyn protokollan.
- **Perusominaisuuksissa** (*basic features*) on perustietoa yhteyksistä, esimerkiksi liikennöinnissä käytettyjen tavujen ja pakettien lukumäärät.
- **Sisältöominaisuudet** (*content features*) käsittävät TCP- ja HTTP-protokollien ominaisuuksia.
- **Aikaominaisuudet** (*time features*) sisältävät yhteyksien aikoihin liittyviä ominaisuuksia. Tällainen on esimerkiksi TCP-protokollan SYN- ja SYN_ACK-pakettien välissä kulunut aika.
- **Lisäominaisuudet** (*additional generated features*) on laskettu neljän aiemman ryhmän ominaisuuksien perusteella, ja tekijät ovat niihin pyrkineet koostamaan mahdollisesti analyysissä hyödynnettävää informaatiota. Lisäominaisuudet on edelleen jaettu kahteen ryhmään: yleisominaisuudet (*general purpose features*) ja yhteysominaisuudet (*connection features*).

Datajoukon instanssien jakauma kymmenen pääryhmän osalta on esitetty taulukossa 1. Testausdatassa normaalin liikenteen osuus on harjoitusdataa tuntuvasti vähäisempi, ja se sisältää enemmän eri hyökkäysryhmiin kuuluvaa liikennettä. Datajoukko siis sisältää normaalin verkkoliikenteen lisäksi haittaliikennettä. Tekijät ovat jakaneet haittaliikenteen yhdeksään pääryhmään, joihin tämän tutkielman toteutusvaiheessa viitataan niiden alkuperäisillä nimillä (Moustafa ja Slay 2015, 2016a):

- ***Analysis***-ryhmä sisältää porttiskannauksia (*port scanning*), roskapostia sekä HTML-pohjaisia hyökkäyksiä.

Taulukko 1: Datan jakauma UNSW-NB15-datajoukossa.

Ryhmä	**Osuus opetus-datassa (%)**	**Osuus testaus-datassa (%)**
Normal	44,94	31,94
Analysis	0,82	1,14
Backdoor	0,71	1,00
DoS	4,97	6,99
Exploits	13,52	19,04
Fuzzers	7,36	10,37
Generic	22,92	22,81
Reconnaissance	4,25	5,98
Shellcode	0,46	0,65
Worms	0,05	0,07

- **Takaovien** (*backdoors*) avulla hyökkääjä voi vaivihkaisesti ohittaa olemassa olevat turvamekanismit ja saada pääsyn kohdekoneeseen tai sen tiedostoihin.
- **Palvelunestohyökkäykset** (*DoS*), jotka on esitelty luvussa 3.1.
- **Haavoittuvuuksien** (*exploits*) hyödyntäminen vaatii hyökkääjältä tietoa käyttöjärjestelmän tai ohjelmien tietoturva-aukoista. Hyökkääjän hyödyntämä haavoittuvuus voi olla joko aiemmin tuntematon nollapäivähaavoittuvuus tai yleisesti tiedossa oleva haavoittuvuus.
- **Fuzzauspohjaiset** (*fuzzers*) hyökkäykset pyrkivät löytämään haavoittuvuuksia verkosta, käyttöjärjestelmästä tai ohjelmasta syöttämällä niihin satunnaisesti generoitua dataa.
- **Yleishyökkäykset** (*generic*) sisältävät muita eri ohjelmien haavoittuvuuksia hyödyntäviä hyökkäyksiä.
- **Tiedon urkinnat** (*reconnaissance*), jotka on esitelty luvussa 3.2.
- **Shellcode-hyökkäykset** ovat joukko käskyjä, jotka hyökkääjä ajaa varsinaisen haavoittuvuuden hyödyntämisen jälkeen. Shellcode-hyökkäyksen sisältö voi olla lähes

mitä tahansa, ja se voi esimerkiksi luoda uuden käyttäjätilin kohdekoneelle (Foster ja Price 2005, luku 8).

- **Madot** (*worms*), jotka on esitelty luvussa 3.5.

5.2 Menetelmien toimivuuden arviointi

Menetelmien toimivuutta ja tulosten luotettavuutta voidaan arvioida sekä vertailla erilaisilla keinoilla ja mittareilla. Luvussa 5.2.1 esitellään näistä ristiinvalidointi, luvussa 5.2.2 luottamusväli ja luvussa 5.2.3 varsinaiset arvioinnissa käytettävät mittarit.

5.2.1 Ristiinvalidointi

Ristiinvalidoinnin (*cross-validation*) perusajatuksena on saada selville, kuinka hyvin opetusdatan pohjalta muodostettu malli onnistuu sijoittamaan oikeisiin ryhmiin validointidatan instanssit joita se ei ole aiemmin nähnyt. Validointidatalla tarkoitetaan opetusdatasta irrotettua datan osajoukkoa, jota ei käytetä mallin opetusvaiheessa. Varsinainen testausdata on siten erillinen datajoukko, jota ei ristiinvalidointivaiheessa käytetä. Edelleen validoinnin tarkoituksena on varmistaa, että muodostettu malli toimii riittävällä tarkkuudella. Jotta ristiinvalidoinnilla saadut tulokset olisivat luotettavia, tulee opetus- ja validointidatana käytettyjen datajoukkojen olla itsenäisiä. Tämä tarkoittaa, että yksi datajoukon instanssi voi kuulua vain joko opetusdatajoukkoon tai validointidatajoukkoon, ei molempiin (Kantardzic 2011, luku 4.7). Ristiinvalidoinnin menetelmiä ovat:

- **Holdout**, jossa datajoukko jaetaan kahteen osaan. Ensimmäinen osa sisältää opetusdatana käytettävät instanssit ja toisen osan instansseja käytetään validointidatana. Opetus- ja validointidatan suhde voi olla mikä tahansa, mutta yleisin valinta on kaksi kolmasosaa opetusdataa sekä loput validointidataa (Kohavi 1995; Kantardzic 2011). Ristiinvalidoinniksi menetelmää voidaan kutsua vasta sitten, kun se on ajettu enemmän kuin yhden kerran, jolloin saadut tulokset ovat useamman ajon keskiarvoja (Arlot ja Celisse 2010).
- **K-kertainen ristiinvalidointi** (*k-fold cross-validation*), jossa datajoukko jaetaan k:hon likimain yhtä suureen osaan. Yhtä näistä osista käytetään validointidatajoukkona ja

loppuja opetusdatana. Validointi suoritetaan k kertaa, ja jokaista osaa käytetään validoinnin aikana vuorollaan kerran validointidatana (Kohavi 1995; Friedman, Hastie ja Tibshirani 2009, luku 7.10.1) .

- **Leave-one-out**, joka on k-fold-ristiinvalidoinnin erikoistapaus, jossa k:n arvoksi määritellään datajoukon instanssien lukumäärä. Tällöin jokainen instanssi muodostaa oman joukkonsa, ja jokaista instanssia käytetään vuorollaan validointidatana (Kantardzic 2011).

Datajoukon instanssien valinta voidaan suorittaa ositetusti (*stratified cross-validation*), jolloin eri ryhmiä edustavia instansseja on kussakin joukossa suhteessa saman verran kuin alkuperäisessä datajoukossa (Kohavi 1995). Tässä tutkielmassa ositettu otanta suoritetaan aina siten, että otannassa huomioidaan datajoukon kymmenen ryhmän jakauma.

5.2.2 Luottamusväli

Luottamusväli (*confidence interval*) ilmaisee kuinka suurella todennäköisyydellä vastaavanlaisen uuden otoksen keskiarvo on ilmoitettujen arvojen välillä. Luottamusvälin ala- ja yläarajat saadaan kaavasta

$$\bar{x} \pm z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}, \tag{5.1}$$

missä $\bar{x}$ on otoksen keskiarvo, $z_{\alpha/2}$ luottamustasoa vastaava kriittinen arvo, σ otoksen keskihajonta ja n havaintojen lukumäärä. Luottamustasoa vastaava kriittinen arvo voidaan laskea normaalijakaumasta. Esimerkiksi luottamustasoja 95 %, 99 % ja 99,9 % vastaavat kriittiset arvot ovat 1,96, 2,58 ja 3,30 (Nummenmaa, Holopainen ja Pulkkinen 2016, luku 7.1).

5.2.3 Arvioinnin mittarit

Menetelmien hyvyyttä voidaan arvioida eri mittareilla, joiden pohjana ovat luvun 4 alussa esitellyt neljä luokkaa: *TN*, *TP*, *FN* ja *FP*. Näiden neljän luokan arvoista voidaan laskea oikeiden negatiivisten osuus (*true negative rate* eli *TNR*), oikeiden positiivisten osuus (*true positive rate* eli *TPR* tai *recall*), väärien positiivisten osuus (*false positive rate* eli *FPR*),

tarkkuus (*precision*), osuvuus (*accuracy*) ja F-mitta (*F-measure*) seuraavasti (Fawcett 2006; Wu ja Banzhaf 2010):

$$TNR = \frac{TN}{TN+FP} \tag{5.2}$$

$$TPR = recall = \frac{TP}{TP+FN} \tag{5.3}$$

$$FPR = \frac{FP}{FP+TN} \tag{5.4}$$

$$precision = \frac{TP}{TP+FP} \tag{5.5}$$

$$accuracy = \frac{TN+TP}{TN+FP+TP+FN} \tag{5.6}$$

$$F\text{-}measure = \frac{2 \cdot TPR \cdot precision}{TPR + precision}. \tag{5.7}$$

Jatkossa näihin mittareihin viitataan tekstissä ja taulukoissa niiden englanninkielisillä nimillä, lukuun ottamatta osuvuutta (*accuracy*).

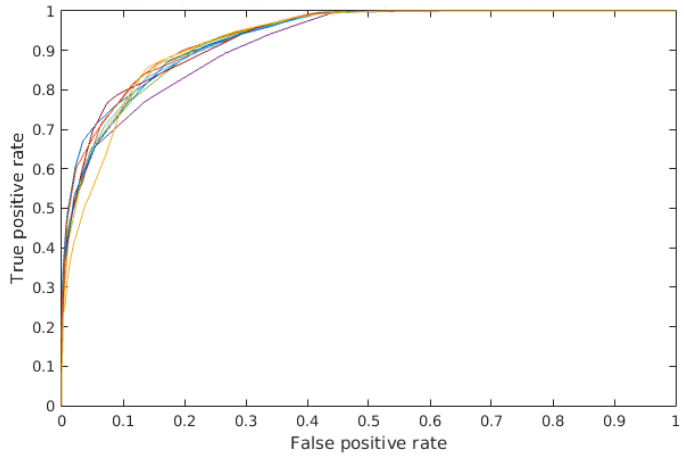
Menetelmän toimivuutta voidaan kuvata myös ROC-käyrän (*receiver operating characteristic*) avulla. ROC-käyrä muodostetaan siten, että kaksiulotteisen kuvaajan x-akselille sijoitetaan väärien positiivisten osuus eli *FPR* ja y-akselille oikeiden positiivisten osuus eli *TPR*. Kuviossa 11 on esitetty ositetulla 10-kertaisella ristiinvalidoinnilla aikaansaadut kymmenen ROC-käyrää. Nämä esimerkkikäyrät on saatu aikaan luvussa 6.4 käytetyillä asetuksilla, ja esitetyt käyrät kuvaavat taulukossa 2 esiintyvän minimi-maksimi-normalisointimenetelmän toimintaa.

Jokaiseen muodostettuun ROC-käyrään kuuluvat aina koordinaattipisteet $(0,0)$ ja $(1,1)$. Pisteessä $(0,0)$ luokitellaan kaikki instanssit haitallisiksi, ja pisteessä $(1,1)$ vastaavasti kaikki

instanssit luokitellaan normaaleiksi. Mallin optimaalinen toimintapiste löytyy luonnollisesti näiden ääripäiden väliltä, ja sen tulisi olla mahdollisimman lähellä pistettä $(0,1)$, jolloin väärien hälytysten määrä on mahdollisimman alhainen. Eräs tapa optimaalisen toimintapisteen valintaan on laskea ensin optimaalinen kulmakerroin S kaavalla (Metz 1978)

$$S = \frac{C_{FP} - C_{TN}}{C_{FN} - C_{TP}} \cdot \frac{N}{P}, \tag{5.8}$$

missä C on luokittelulle määritelty kustannus, N datajoukossa olevan normaalin liikenteen havaintojen lukumäärä ja P datajoukossa olevan haitallisen liikenteen havaintojen lukumäärä. Esimerkiksi Matlabissa väärien luokittelujen eli C_{FP}:n ja C_{FN}:n kustannuksiksi on määritelty 0,5, ja oikeiden luokittelujen eli C_{TN}:n ja C_{TP}:n kustannuksiksi 0 (“MathWorks documentation: Perfcurve” 2016). Optimaalisen kulmakertoimen laskemisen jälkeen etsitään käyrältä piste, jossa kulmakerroin on mahdollisimman lähellä laskettua. Tätä kautta saadaan selville mallissa käytettävä kynnysarvo (*threshold*). Kynnysarvo määrittää, milloin luokiteltava oleva instanssi sijoitetaan tiettyyn luokkaan. Esimerkiksi yhdessä klusterissa oleville instansseille on voitu laskea anomaliapistemäärä (*anomaly score*), joka kertoo millä todennäköisyydellä satunnainen klusterin instanssi edustaa tiettyä luokkaa, tässä tapauksessa haittaliikennettä. Tällöin kynnysarvon ylittävät instanssit luokitellaan haittaliikenteeksi (Patcha ja Park 2007).



Kuvio 11: Esimerkki kymmenestä ROC-käyrästä.

Huonointa mahdollista mallia edustava ROC-käyrä on ääripäiden pisteiden $(0,0)$ ja $(1,1)$ vä-

lille piirretty suora viiva. Tällainen luokittelija vastaa täysin sattumanvaraista valintaa kahden vaihtoehdon välillä (Fawcett 2006; Axelsson 2000). Muodostetulle ROC-käyrälle voidaan laskea käyrän alla oleva pinta-ala (*area under curve* eli *AUC*), jota muiden muassa Bradley (1997) ja Huang ja Ling (2005) ovat pitäneet yksinkertaisena ja toimivana mittarina koneoppimisen menetelmien vertailussa. Esimerkiksi kuviossa 11 esitettyjen kymmenen ROC-käyrän keskimääräinen AUC-arvo on 0,9322.

5.3 Toteutettavien menetelmien valinta ja käytettävät työkalut

Kantavana ideana oli toteuttaa anomaliapohjaiset menetelmät käytännössä siten, että ne kaikki edustaisivat eri pääryhmiä luvussa 4.2 esitellyssä jaottelussa. Toinen lähtökohta valinnoille oli, että valittujen menetelmien tulisi olla laajasti ja yleisesti käytettyjä tutkimusalalla, eli niissä on jotain tutkimusalalla toimivaksi havaittua. Kolmantena tekijänä valintaan vaikutti valmiiden työkalujen saatavuus menetelmien toteuttamiselle.

Toteutettavaksi valittiin edellä mainittujen kriteerien pohjalta

- luvussa 6 esiteltävä klusterointi- ja poikkeava havainto -ryhmää edustava *k*-means, jolle löytyy valmis toteutus Matlabin *Statistics and machine learning toolbox* -lisäosasta
- luvussa 7 esiteltävä luokittelupohjaisia menetelmiä edustava tukivektorikone, joka toteutetaan Matlabin kautta käytettävän LIBSVM-kirjaston avulla
- luvussa 8 esiteltävä pehmeän laskennan ryhmää edustava keinotekoiset neuroverkot, joka toteutetaan Matlabin *Neural network toolbox* -lisäosan avulla.

Menetelmät toteutettiin ja niihin liittyvät laskennat suoritettiin Matlabin R2015a-versiolla. Tietokoneen käyttöjärjestelmänä oli Kubuntu 16.04 LTS, ja koneessa oli neliytiminen Intel Core i3-2330M @ 2,20 GHz -suoritin sekä 4 GB keskusmuistia.

6 K-means

K-means on osittteleva klusterointimenetelmä, jossa k:lle käyttäjän antama arvo määrää muodostettavien klustereiden lukumäärän (Gan, Ma ja Wu 2007, luku 9.1).

6.1 Algoritmin kuvaus

Olkoon D klusteroitava datajoukko, jossa on n kappaletta instansseja. Merkitään D:n klustereita $C_1, C_2, \cdots, C_k$, jolloin k-meansin virhefunktio – jonka algoritmi pyrkii minimoimaan – määritellään (Gan, Ma ja Wu 2007)

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu(C_i)), \tag{6.1}$$

missä $\mu(C_i)$ on klusterin C_i keskipiste eli sentroidi ja $d(x, \mu(C_i))$ instanssin x etäisyys klusterin keskipisteestä.

K-means-algoritmin k-means++-muunnos on esitetty algoritmilistauksessa 1. Esitys pohjautuu listauksiin, jotka Han ja Kamber (2006, luku 7.4) sekä Arthur ja Vassilvitskii (2007) esittivät. K-means-algoritmista käytettiin tutkielmassa valmista Matlabin sisäistä toteutusta, jossa aloitusklustereiden valintaan käytetään oletuksena k-means++-algoritmissa esitettyä tapaa. K-means++-algoritmin mukainen aloitusklustereiden valinta on algoritmilistauksessa 1 esitetty riveillä 1–4. K-means++ pyrkii valitsemaan keskenään mahdollisimman erilaiset aloitusklusterit (Arthur ja Vassilvitskii 2007). Yksinkertaisempi tapa on valita aloitusklusterit datajoukosta täysin satunnaisesti. Heikosti valituilla aloitusklustereilla algoritmi voi kuitenkin helpommin päätyä epäedulliseen klusterointiratkaisuun minimoimalla virhefunktion 6.1 vain lokaalisti (Kantardzic 2011, luku 9.4). Maksimimääräksi algoritmin riveillä 5–8 esitetylle iterointivaiheelle määriteltiin Matlabissa 500 kierrosta.

K-means on käyttökelpoinen ainoastaan numeerisella datalla, jolla usean instanssin muodostamille ryhmille voidaan mielekkäästi laskea keskiarvot muuttujien arvojen perusteella (Gan, Ma ja Wu 2007). Algoritmin aikavaatimus on $O(nkt)$, missä n on datajoukon instanssien, k klustereiden ja t iterointikierrosten lukumäärä (Han ja Kamber 2006). Koska algoritmin aikavaatimus on lineaarisesti riippuvainen datajoukon koosta, on suurienkin datajoukkojen

Algoritmi 1 *K*-means++

Syötteet: Klustereiden lukumäärä k ja datajoukko D

1: valitse aloitusklusteri C_1 satunnaisesti datajoukosta D

2: **toista**

3: valitse seuraava klusterin keskipiste $C_i = x' \in D$ todennäköisyydellä $\frac{d(x')^2}{\sum_{x \in D} d(x)^2}$, missä $d(x)$ on pienin havaittu etäisyys instanssin x ja lähimmän klusterin keskipisteen välillä

4: **kunnes** k aloitusklusteria valittu

5: **toista**

6: Sijoita jokainen D:n instanssi sitä lähinnä olevaan klusterin keskipisteeseen. Klusterin keskipiste määräytyy klusterissa olevien instanssien keskiarvon mukaan.

7: laske uudelleen keskiarvot kullekin klusterille siihen kuuluvien instanssien perusteella.

8: **kunnes** ei muutoksia

9: **palauta** k klusteria ja tieto niihin kuuluvista D:n instansseista

klusterointi algoritmilla tehokasta (Gan, Ma ja Wu 2007).

6.2 Klustereiden määrän arviointi

Opetus- ja testausdatajoukoista poistettiin ensimmäiseksi rivit yksilöivä *id*-arvo. Myös kaksi nimiöintitiedot sisältävää ominaisuutta otettiin datajoukoista erilleen. Ensimmäisessä vaiheessa käytettäviksi ominaisuuksiksi valittiin kaikki mielekkäällä tavalla *k*-means-algoritmin kanssa yhteensopivat ominaisuudet. Tällaisia olivat kaikki numeerisia muuttujan arvoja sisältävät ominaisuudet. Ominaisuusjoukosta poistettiin näin ollen tässä vaiheessa kategorisia muuttujan arvoja sisältävät ominaisuudet.

Ensimmäisessä ominaisuuksien valinnassa otettiin siis mukaan seuraavat 36 ominaisuutta: *dur, spkts, dpkts, sbytes, dbytes, rate, sttl, dttl, sload, dload, sloss, dloss, sinpkt, dinpkt, sjit, djit, swin, stcpb, dtcpb, dwin, tcprtt, synack, ackdat, smean, dmean, trans_depth, response_body_len, ct_srv_src, ct_state_ttl, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_flw_http_mthd, ct_src_ltm, ct_srv_dst*.

K-meansin *k*-arvon eli klustereiden määrän arviointiin käytettiin L-menetelmän (*L method*)

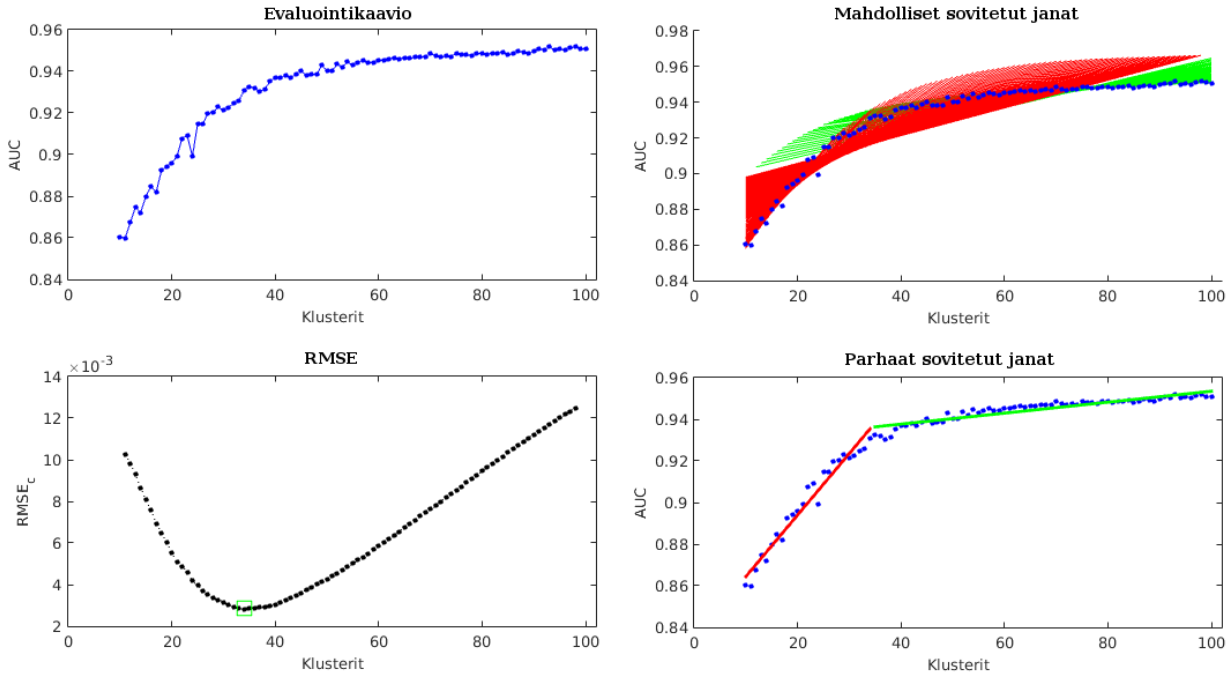
(Salvador ja Chan 2004) Matlab-toteutusta (Zagouras 2014; Zagouras ym. 2013; Zagouras, Inman ja Coimbra 2014). Käytetyssä datajoukossa oli yhteensä kymmenen ryhmää, joten klustereiden mahdolliseksi minimimääräksi määriteltiin kymmenen kappaletta. Datajoukon kymmenessä ryhmässä taas oli yhteensä 207 aliryhmää, joten klusterien maksimimääräksi pidettiin perusteltuna valita melko suuri arvo. Lopulliseksi klusterien mahdolliseksi maksimimäärän arvoksi valittiin noin puolet aliryhmien määrästä, eli sata kappaletta.

Alustavat etäisyysmittaritestit osoittivat kolmen etäisyysmittarin suorituskyvyn olevan melko lähellä toisiaan. Tarkemmin etäisyysmittareiden vaikutusta tarkastellaan luvussa 6.3. Etäisyysmittarina päädyttiin tässä vaiheessa käyttämään Pearsonin korrelaatioetäisyyttä (ks. kaava 4.16). Normalisointimenetelmänä käytettiin hyvin yleisesti käytössä olevaa z-piste-menetelmää (ks. kaava 4.11). Myös normalisointimenetelmien vaikutusta käsitellään tarkemmin myöhemmin luvussa 6.4.

K-means-algoritmi ajettiin UNSW-NB15-datajoukon *UNSW_NB15_training-set.csv*-tiedoston opetusdatalle ositetulla 10-kertaisella ristiinvalidoinnilla (*stratified 10-fold cross-validation*) jokaiselle *k*:n arvolle eli klusterimäärälle välillä 10–100. Jokainen näistä 91 ajosta tuotti näin ollen kymmenen ROC-käyrää, joista klusterimäärän hyvyyden mittariksi valittiin kymmenen ROC-käyrän keskimääräinen AUC-arvo. Tässä ja myöhemmissä opetusvaiheissa huomioitiin ainoastaan kaksi nimiöinnin pääluokkaa – normaali ja haittaliikenne – eli datajoukon tarkempaa kymmenen ryhmän jaottelua ei otettu huomioon.

L-menetelmässä kaksiulotteisen kuvaajan x-akselille sijoitetaan klustereiden määrä ja y-akselille valitun mittarin tuottama arvo (ks. evaluointikaavio kuviossa 12). Tähän kuvaajaan pyritään mittaustulosten päälle sovittamaan kaksi janaa (ks. mahdolliset sovitetut janat kuviossa 12) siten, että keskineliövirheen neliöjuuri (*root mean squared error* eli *RMSE*) on mahdollisimman pieni (ks. RMSE kuviossa 12). Tämän jälkeen kahden pisteille sovitetun janan yhtymäkohdasta (ks. parhaat sovitetut janat kuviossa 12) eli L-kirjaimen muotoisen ”polven” kohdalta saadaan menetelmän ehdottama klustereiden lukumäärä (Salvador ja Chan 2004). Tässä tapauksessa lukumääräksi saatiin 34 klusteria.

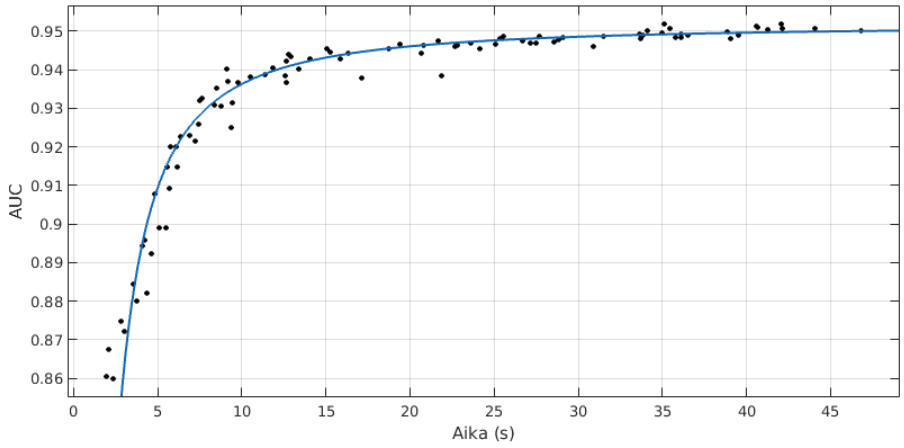
Klustereiden määrän valintaa voidaan tarkastella myös ajankäytön näkökulmasta. Mitä useampaan klusteriin data jaetaan, sitä enemmän laskentaresursseja algoritmi vie. Kuvion 12 eva-



Kuvio 12: *K*-means: klustereiden määrän arviointi L-menetelmällä.

luointikaaviosta voidaan kuitenkin nähdä, että tietyn klusterimäärän jälkeen algoritmin suorituskyky ei enää merkittävästi – jos ollenkaan – parane. Kuvioon 13 on sovitettu käyrä, jossa on esitetty käytetyn ajan suhde algoritmin suorituskykyyn. X-akselilla esitetyt ajat on saatu ositetulla 10-kertaisella ristiinvalidoinnilla jakamalla siihen kokonaisuudessaan käytetty aika kymmenellä, jolloin on saatu yksittäiseen *k*-means-ajoon kulunut keskimääräinen aika. Y-akselin AUC-arvot ovat samoin ositetun 10-kertaisen ristiinvalidoinnin tuottamia keskimääräisiä arvoja. Kuviosta voidaan siis edelleen päätellä, että tietyn rajan jälkeen klustereiden laskentaan käytetyllä ajalla ei ole enää vaikutusta suorituskykyyn.

Tulevissa luvuissa 6.3 ja 6.4 tehtyjen testien ja niistä saatujen lisätietojen perusteella palattiin vielä kerran kokeilemaan klustereiden määrän arviointia L-menetelmällä. Tässä testissä käytettiin ominaisuuksien valinnan algoritmi JMI:tä valitsemaan ominaisuusjoukosta kuusi ominaisuutta, etäisyysmittariksi valittiin Manhattan-etäisyys ja normalisointimenetelmäksi minimi-maksimi-menetelmä. Klustereiden minimi- ja maksimimääriksi määriteltiin aiemmin käytetyt 10 ja 100 kappaletta. Näillä asetuksilla L-menetelmä ehdotti klustereiden määräksi 30 kappaletta, joka on edelleen melko lähellä aiemmin arvioitua 34 kappaletta.



Kuvio 13: *K*-means: käytetty aika suhteessa menetelmän suorituskykyyn.

6.3 Ulottuvuuksien vähentäminen ja ominaisuuksien valinta

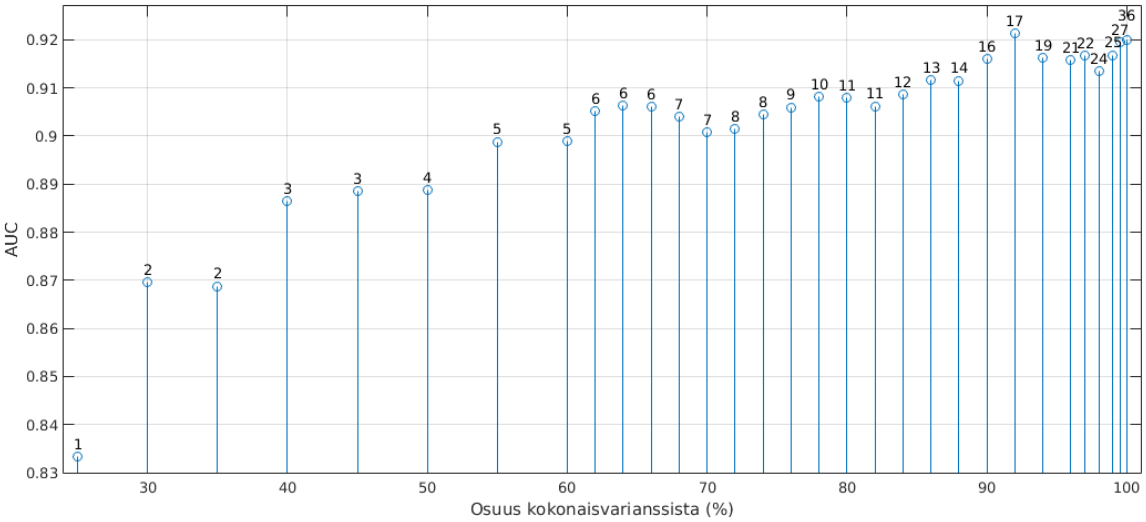
Laskennan nopeuttamisen ja suorituskyvyn parantamisen vuoksi kokeiltiin datan vähentämistä (ks. luku 4.5.1) kahdella eri menetelmällä. Ensimmäiseksi kokeiltiin ulottuvuuksien vähentämistä pääkomponenttianalyysin avulla, minkä jälkeen testattiin ominaisuuksien valintaa kolmella eri suodatinmallia edustavalla ominaisuuksien valinnan menetelmällä.

6.3.1 Ulottuvuuksien vähentäminen pääkomponenttianalyysillä

Pääkomponenttianalyysin tuottama muuttujajoukko on järjestetty varianssin mukaan merkityksellisimmästä vähämerkityksisimpään, joten osa viimeisistä ulottuvuuksista voidaan käytännössä jättää huomiotta. Valittujen ulottuvuuksien määrää hallittiin parametrilla, jossa määriteltiin kuinka monta prosenttia muuttujajoukon kokonaisvarianssista haluttiin huomioida. Käytännössä esimerkiksi määrittelemällä tämä parametri vaikkapa 90 prosenttiin karsiutui pois 36 ulottuvuudesta viimeiset noin 20 ulottuvuutta. Tällä tavalla algoritmin nopeus myös paranee huomattavasti ilman havaittavissa olevaa vaikutusta suorituskykyyn.

Sopivaa arvoa varianssiparametrille etsittiin kasvattamalla sitä vähitellen (ks. kuvio 14). Kuviossa jokaisen mittaustuloksen yllä on esitetty varianssiparametria vastaava ominaisuuksien määrä. Korkein AUC-arvo, 0,921, saavutettiin tässä mittauksessa kokonaisvarianssiparametrin ollessa 92%, jolloin ominaisuuksista otettiin mukaan laskentaan 17 ensimmäistä. Jokainen kuvion mittaustulos on ositetusti ja 10-kertaisesti ristiinvalidoitu viisi kertaa, eli esitetty

AUC-arvo on 50 erillisen ajon keskiarvo. Mittauksessa k-meansin k:n arvoksi määriteltiin luvussa 6.2 L-menetelmällä arvioitu 34 kappaletta, etäisyysmittarina käytettiin Manhattan-etäisyyttä ja normalisointimenetelmänä z-piste-menetelmää.



Kuvio 14: K-means PCA:lla: Huomioitavien ulottuvuuksien määrän arviointi.

6.3.2 Ominaisuuksien valinta

Ominaisuuksien valinnan ajaksi kaikki muuttujan arvot muutettiin diskreeteiksi. Diskretisointiin käytettiin Matlabissa *quantize*-funktiota (Schroedl 2010), joka jakaa muuttujajoukon arvot x:ään kvantiiliin. X:n arvoksi määriteltiin 10, jolloin diskretisoinnin jälkeen jokaisen ominaisuuden muuttujajoukossa oli uniikkeja arvoja enintään kymmenen kappaletta. Diskretisointia ei siis suoritettu niiden ominaisuuksien muuttujajoukoille, joissa erilaisia muuttujien arvoja oli alle kymmenen kappaletta. Diskretisoidut muuttujien arvot olivat käytössä ainoastaan ominaisuuksien valinnan ajan. Varsinaisen analyysin aikana käytettiin siis edelleen datajoukon alkuperäisiä arvoja.

Suodatinmallia käyttävistä ominaisuuksien valinnan algoritmeista kokeiltiin JMI- (määritelty kaavassa 4.1), CMIM- (4.2) ja CIFE (4.3) -algoritmeja. Brown ym. (2012) suorittivat ominaisuuksien valinnan algoritmien vertailun, jossa nämä kolme algoritmia pärjäsivät parhaiten. Ominaisuuksien valinnassa hyödynnettiin Matlabissa FEAST-lisäosaa (“FEAST: A feature selection toolbox for C and Matlab” 2011; Brown ym. 2012). Ominaisuuksien valin-

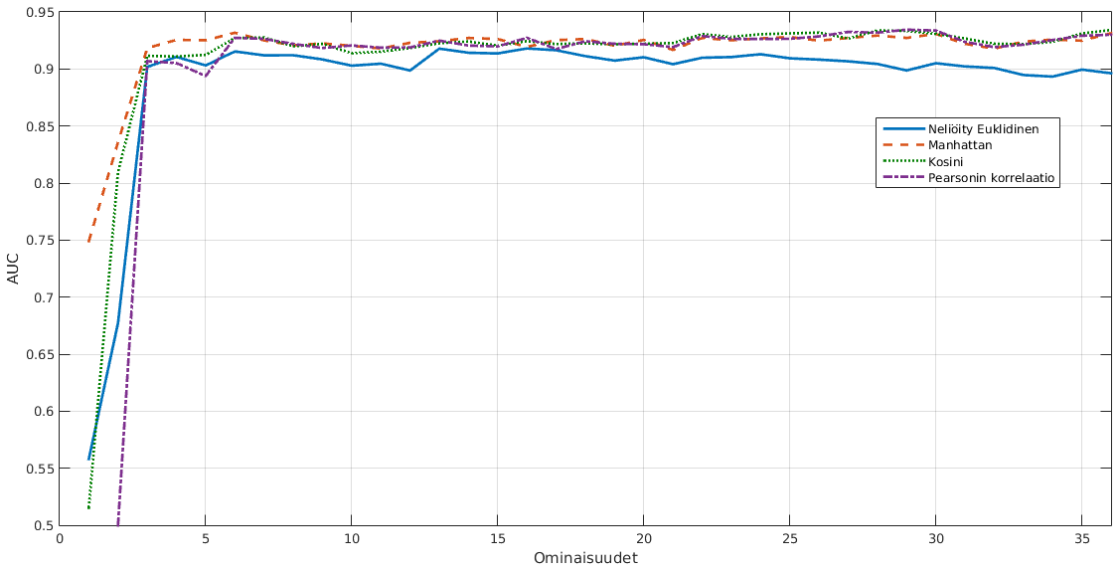
nan algoritmit vaativat toimiakseen datajoukon nimiöintitietoja, joiksi valittiin ryhmänumeroihin 0–9 instanssit ryhmittelevät nimiöintitiedot. Näiden kolmen ominaisuuksien valinnan algoritmin valitsemat ominaisuudet on listattu liitteen A taulukossa 18. Näistä listauksista puuttuvat *k*-means-toteutuksen alussa datajoukosta karsitut kategoriset ominaisuudet.

Ominaisuuksien valinnan algoritmien hakemia ominaisuusjoukkoja testattiin ensimmäiseksi neliöidyllä Euklidisella etäisyydellä (*squared Euclidean distance*), jolloin kaavassa 4.13 on $(p,r) = (2,1)$ (Deza ja Deza 2009, luku 17.2). Seuraavaksi käytettiin Manhattan- eli L_1-etäisyyttä, joka tunnetaan myös nimellä *city block* -etäisyys. Manhattan-etäisyyttä käytettäessä kaavassa 4.13 on $p = r = 1$ (Deza ja Deza 2009, luku 17.2). Kolmanneksi kokeiltiin kosinietäisyyttä, joka on määritelty kaavassa 4.15. Viimeiseksi testattiin vielä kaavassa 4.16 määriteltyä Pearsonin korrelaatioetäisyyttä. Testit ajettiin jokaiselle ominaisuusmäärälle välillä 1–36, lukuun ottamatta Pearsonin korrelaatioetäisyyttä, jota ei voinut käyttää ominaisuuksien määrän ollessa 1.

Tulokset JMI-algoritmille on esitetty kuviossa 15, CMIM-algoritmille kuviossa 16 ja CIFE-algoritmille kuviossa 17. *K*-meansin *k*:ksi eli klustereiden määräksi määriteltiin 34 kappaletta, ja normalisointimenetelmänä käytettiin z-piste-menetelmää. Testit ajettiin ositetulla 10-kertaisella ristiinvalidoinnilla, joten kuvioissa esiintyvät AUC-arvot ovat kymmenen ajon keskiarvoja.

JMI- ja CMIM-algoritmien valitsemissa ominaisuuksissa neliöidyn Euklidisen etäisyyden suorituskyky jää jälkeen muista ominaisuuksien eli ulottuvuuksien määrän kasvaessa. CIFE:n valitsemissa ominaisuuksissa neliöity Euklidinen etäisyys taas on selkeästi muista jäljessä koko ajan, ja CIFE kokonaisuudessaan ei tässä vertailussa pärjää kahdelle muulle algoritmille. Kokonaisuutena JMI-algoritmin valitsemat ominaisuudet näyttävät valitulla mittarilla toimivan parhaiten, sillä jo kolmella valitulla ominaisuudella saavutetaan kaikilla etäisyysmittareilla yli 0,9:n keskimääräinen AUC-arvo. Myös CMIM pärjää hyvin valittujen ominaisuuksien ylittäessä neljä kappaletta.

Kuviossa 18 on esitetty kunkin algoritmin ominaisuuksien valintaan käyttämä aika ominaisuuksien määrille 1–35. Esitetyt arvot ovat viiden ajon keskiarvoja. Tässä vertailussa parhaiten pärjäävät JMI- ja CMIM-algoritmit. Tosin CMIM käyttää suurilla ominaisuusmäärillä



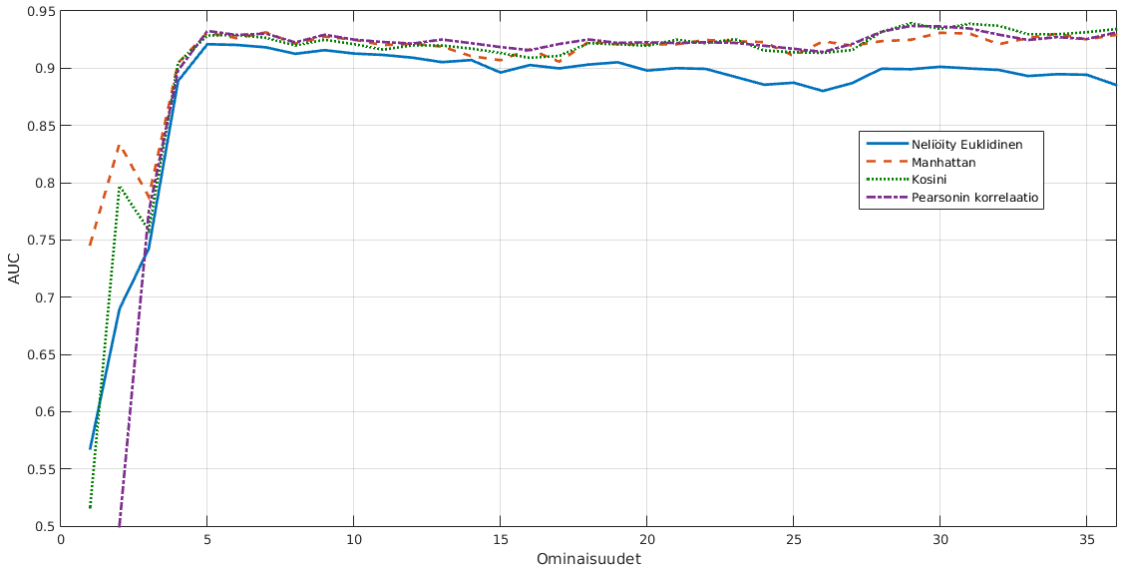
Kuvio 15: *K*-means: Valittavien ominaisuuksien määrän arviointi JMI-algoritmilla.

enemmän aikaa valintaan verrattuna JMI:hin.

Kuviossa 19 vertaillaan eri etäisyysmittareiden vaikutusta laskennan kestoon ominaisuuksien määrän kasvaessa. Vertailussa on käytetty JMI-algoritmin valitsemia ominaisuuksia, ja muina asetuksina olivat aiemmin tässä luvussa esitellyt parametrit. Jokainen kuviossa esitetty mittaustulos perustuu kymmenen kertaa ajettuun ositettuun 10-kertaiseen ristiinvalidointiin, jolloin y-akselilla ilmoitetut ajat ovat keskiarvoja yksittäisille algoritmin 1 suorituksille.

6.4 Normalisointimenetelmien vertailu

Z-piste- (määritelty kaavassa 4.11), minimi-maksimi- (kaava 4.8) ja logaritmisia (kaava 4.12) normalisointimenetelmiä vertailtiin opetusdatajoukolla ajamalla kukin normalisointimenetelmä ositetulla 10-kertaisella ristiinvalidoinnilla kymmenen kertaa. Tällöin jokaisen otoksen kooksi saatiin sata kappaletta. Etäisyysmittariksi valittiin luvussa 6.3 etenkin pienillä ulottuvuuksien määrällä AUC-mittarilla ja nopeutensa puolesta hyvin pärjännyt Manhattan-etäisyys. Muina asetuksina *k*-meansissa oli 34 klusteria, ja ominaisuuksien valinta suoritettiin JMI-algoritmilla, joka valitsi datajoukosta kuusi ominaisuutta. Tässä mittauksessa luottamusvälin luottamustasoksi valittiin 99,9 %. Mittauksen tulokset on esitetty taulukossa 2.

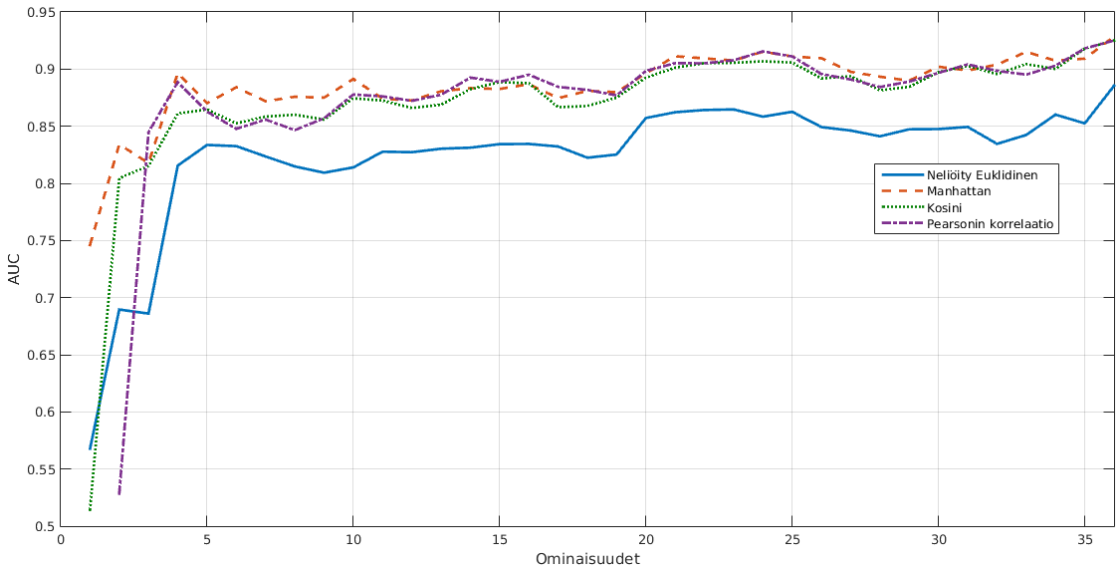


Kuvio 16: *K*-means: Valittavien ominaisuuksien määrän arviointi CMIM-algoritmilla.

Z-piste-menetelmä sijoittui vertailussa vasta kolmanneksi minimi-maksimi- ja logaritmisen menetelmän jälkeen. Z-piste-menetelmää sovellettiin siten, että opetusdatasta saatuja muuttujajoukkojen keskiarvoja ja keskihajontojen arvoja käytettiin validointidatan normalisointiin. Vastaavalla tavalla hyödynnettiin myös minimi-maksimi-menetelmää, eli validointidatan normalisointi toteutettiin opetusdatasta saatujen minimi- ja maksimiarvojen pohjalta. Mikäli validointidatassa havaittiin opetusdataa suurempi arvo, sen arvoksi määriteltiin 1. Vastaavasti opetusdataa pienemmälle arvolle määriteltiin arvoksi 0.

Logaritmisen menetelmän kantalukuna kokeiltiin myös Neperin lukua, mutta merkittävää eroa tuloksissa eri kantalukujen välillä ei havaittu. Sopivat ominaisuudet valitsemalla voi ilman normalisointiakin selvitä kohtuullisesti, mutta tässä vertailussa käytetyillä ominaisuuksilla jäi ilman normalisointia tehty testi selvästi muista jälkeen. Testi toteutettiin myös käyttäen kolmea muuta etäisyysmittaria, ja näiden vertailujen tulokset on esitetty liitteen B taulukoissa 19, 20 ja 21.

Lisäksi vastaavalla tavalla testattiin normalisointimenetelmiä vielä korvaamalla JMI-algoritmin suorittama ominaisuuksien valinta pääkomponenttianalyysin suorittamalla ulottuvuuksien vähentämisellä. PCA:n tuottamista ulottuvuuksista huomioitiin testeissä kuusi ensimmäistä, jolloin määrä oli periaatteessa vertailukelpoinen JMI:n valitseman ominaisuusjoukon kanssa.



Kuvio 17: *K*-means: Valittavien ominaisuuksien määrän arviointi CIFE-algoritmilla.

Taulukko 2: *K*-means: Normalisointimenetelmien vertailu JMI:llä ja Manhattan-etäisyydellä.

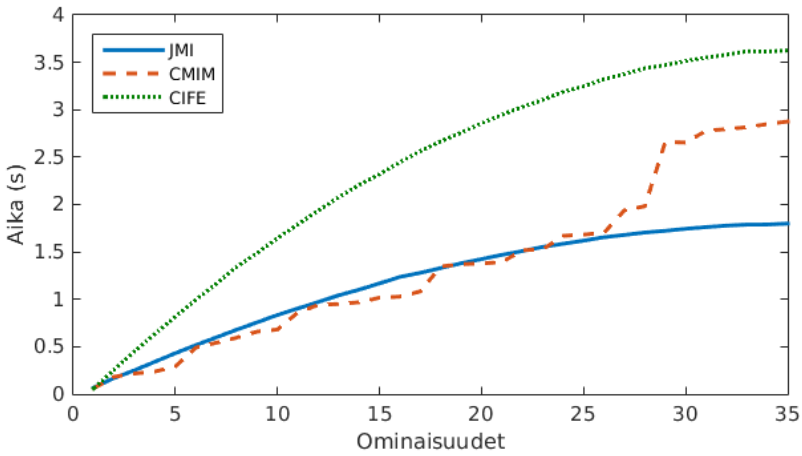
	AUC (keskiarvo)	**Keskihajonta**	**Luottamusväli (99,9 %)**
Z-piste	0,9273	0,0061	0,9253 – 0,9294
Minimi-maksimi	0,9343	0,0061	0,9322 – 0,9363
Logaritmi (kantaluku 10)	0,9315	0,0096	0,9282 – 0,9347
Ei normalisointia	0,7551	0,0401	0,7414 – 0,7687

Muut asetukset olivat samat kuin aiemmassa testissä.

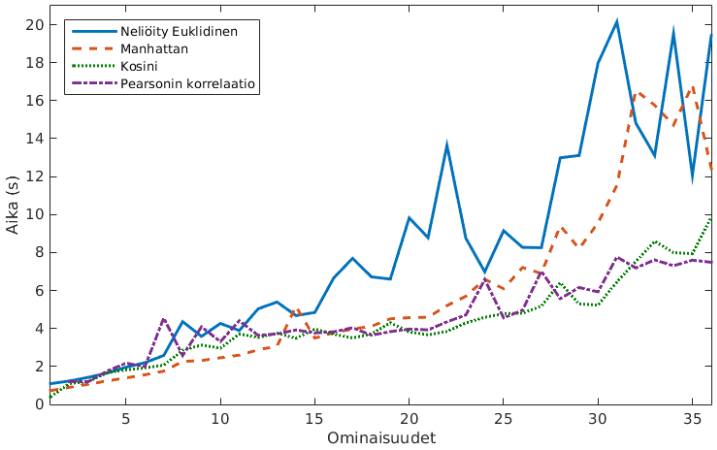
Testin tulokset on esitetty kosinietäisyyden osalta taulukossa 3. Paras AUC-keskiarvo saavutettiin z-piste-menetelmällä, ja muut normalisointimenetelmät osoittautuivat käytännössä käyttökelvottomiksi. Kolmen muun etäisyysmittarin osalta tulokset on esitetty liitteen B taulukoissa 22, 23 ja 24.

6.5 Menetelmän arviointi testausdatalla

Lopuksi menetelmän toimivuutta arvioitiin *UNSW_NB15_testing-set.csv*-tiedoston testausdatalla. Sopiva kynnysarvo etsittiin opetusdatan avulla ositetulla 10-kertaisella ristiinvalidoinnilla siten, että optimaalinen toimintapiste kustakin ROC-käyrästä etsittiin kaavassa 5.8



Kuvio 18: Valittavien ominaisuuksien määrän vaikutus ominaisuuksien valinnan kestoon.



Kuvio 19: *K*-means: Valittavien ominaisuuksien määrän vaikutus laskennan kestoon.

määritellyllä tavalla. Etsintään käytettiin valmista Matlabin *perfcurve*-funktiossa ollutta toteutusta. Näin saatiin kymmenen kappaletta ROC-käyriä ja sama määrä optimaalisia toimintapisteitä. Tässä testissä näistä kymmenestä pisteestä parhaaksi valittiin pienimmän arvon x-akselilla saanut piste, joka samalla tarkoitti myös alhaisinta kynnysarvoa. Klustereiden keskipisteiksi otettiin ne keskipisteet, joita käyttäen löytyi paras optimaalinen ROC-käyrän toimintapiste.

Ositettu 10-kertainen ristiinvalidointi ajettiin sata kertaa aiemmissa luvuissa parhaiksi havaituilla parametreilla. Kaikki taulukoissa 4, 5 ja 7 esiintyvät keskiarvot ovat siten sadan kappaleen otoksesta saatuja keskiarvoja. Klustereiden määrä oli 34 kappaletta, etäisyysmittarina käytettiin Manhattan-etäisyyttä, normalisointimenetelmänä minimi-maksimi-menetelmää ja

Taulukko 3: *K*-means: Normalisointimenetelmien vertailu PCA:lla ja kosinietäisyydellä.

	AUC (keskiarvo)	**Keskihajonta**	**Luottamusväli (99,9 %)**
Z-piste	0,9260	0,0042	0,9246 – 0,9274
Minimi-maksimi	0,4431	0,0261	0,4345 – 0,4518
Logaritmi (kantaluku 10)	0,7553	0,1136	0,7178 – 0,7928
Ei normalisointia	0,5840	0,0395	0,5710 – 0,5970

JMI-algoritmilla valittiin ominaisuusjoukosta kuusi ominaisuutta.

Taulukossa 4 tarkastellaan datajoukon eri ryhmien luokittelun osuvuutta. Taulukon ensimmäisessä tulossarakkeessa huomioidaan ainoastaan kaksi luokkaa, jotka ovat normaali- ja haittaliikenne. Tämä tarkoittaa, että jokaisen hyökkäysryhmän kohdalla huomioidaan ainoastaan kuinka suuri osa ryhmän instansseista on luokiteltu oikein haittaliikenteeksi. Taulukon toisessa tulossarakkeessa tarkastellaan tarkemmin eri ryhmien luokittelun osuvuutta. Sarakkeessa ilmaistu osuvuus siis ilmaisee, kuinka suuri osa ryhmän instansseista on ryhmitelty täsmällisesti oikeaan ryhmään.

Taulukko 4: *K*-means: Luokittelun osuvuus.

Luokka	**Ryhmä**	**Osuvuus kahdella pääluokalla luottamusvälillä 99 %**	**Osuvuus kymmenellä ryhmällä luottamusvälillä 99 %**
Normaali	**Normal**	$0,8591 \pm 0,0165$	$0,8591 \pm 0,0165$
Haitta-liikenne	**Analysis**	$0,8416 \pm 0,0253$	0
	Backdoor	$0,9326 \pm 0,0143$	0
	DoS	$0,9517 \pm 0,0076$	$0,2680 \pm 0,0559$
	Exploits	$0,9372 \pm 0,0117$	$0,5039 \pm 0,0153$
	Fuzzers	$0,7462 \pm 0,0678$	$0,3079 \pm 0,0549$
	Generic	$0,9973 \pm 0,0007$	$0,9736 \pm 0,0013$
	Reconnaissance	$0,8998 \pm 0,0368$	$0,5675 \pm 0,0537$
	Shellcode	$0,9026 \pm 0,0362$	0
	Worms	$0,9202 \pm 0,0286$	0

Taulukossa 5 esitetään datajoukon instanssien ennustetut ja todelliset ryhmät. Taulukon riveillä on esitetty todelliset ryhmät ja sarakkeilla ennustetut ryhmät. Esitetyt arvot ovat lähimpään kokonaislukuun pyöristettyjä sadan kappaleen otoksen keskimääräisiä arvoja. Taulukossa diagonaalisesti esitetyt tummennetut arvot edustavat siis oikeiden luokittelujen lukumääriä.

Taulukko 5: *K*-means: Instanssien ennustetut ja todelliset ryhmät.

		Ennustettu ryhmä									
		Normal	***Analysis***	***Backdoor***	***DoS***	***Exploits***	***Fuzzers***	***Generic***	***Reconnaissance***	***Shellcode***	***Worms***
Todellinen ryhmä	**Normal**	**48110**	0	0	159	1706	3758	1104	1162	0	0
	Analysis	317	**0**	0	516	317	16	493	341	0	0
	Backdoor	118	0	**0**	489	153	54	470	463	0	0
	DoS	593	0	0	**3286**	2114	411	3100	2760	0	0
	Exploits	2098	0	0	4369	**16826**	1472	3789	4839	0	0
	Fuzzers	4615	0	0	950	2315	**5599**	1893	2812	0	0
	Generic	107	0	0	206	360	131	**38944**	252	0	0
	Reconn-aissance	1051	0	0	871	1192	658	765	**5954**	0	0
	Shellcode	110	0	0	50	203	121	26	624	**0**	0
	Worms	10	0	0	1	65	22	1	32	0	**0**

Taulukossa 6 tarkastellaan opetukseen ja testaukseen käytettyä aikaa. Tämän testin tulokset on esitetty taulukon ensimmäisellä tulosrivillä (JMI). Opetusaika on keskimääräinen 10-kertaiseen ristiinvalidointiin kulunut aika, eli ilmoitetun ajan sisällä klusterointi suoritetaan kymmenen kertaa. Sekä opetus- että testausaikoihin on laskettu kuuluvaksi normalisointeihin tai pääkomponenttianalyysiin kulunut aika. Opetusaikaan kuuluu näiden lisäksi klustereiden muodostaminen ja testausaikaan lähimpien klustereiden valinta testausdatalle.

Taulukossa 7 ilmoitetaan mittauksen tulokset seitsemällä eri mittarilla. Taulukon ensimmäi-

sessä tulossarakkeessa (Ominaisuuksien valinta JMI:llä) on ilmoitettu tämän testin tulokset. Lisäksi taulukossa viimeisenä ilmoitetaan keskimääräinen käytetty kynnysarvo. Arvot, lukuun ottamatta osuvuutta kymmenellä ryhmällä, on saatu kahteen luokkaan – haittaliikenne ja normaali – luokittelun perusteella. Luottamusvälin laskemiseksi tarvittavaa tietoa osuvuudelle kymmenellä ryhmällä ei valitettavasti otettu laskennan jälkeen talteen, joten tämä tieto on saatavilla ryhmille vain erikseen taulukossa 4 ja liitteen C taulukossa 25.

Vastaava vertailu testausdatalla tehtiin myös pääkomponenttianalyysin avulla saaduilla ulottuvuusjoukoilla. Asetuksina käytettiin luvussa 6.4 löydettyjä optimaalisia asetuksia. Tässä testissä huomioitiin kuusi ensimmäistä pääkomponenttianalyysin tuottamaa ulottuvuutta. Normalisointimenetelmänä käytettiin z-piste-menetelmää ja etäisyysmittarina kosinietäisyyttä. Muut asetukset olivat samat kuin edellisessä JMI-algoritmia hyödyntäneessä testissä. Useimmilla käytetyillä mittareilla tämä pääkomponenttianalyysiä hyödyntänyt menetelmä jäi hieman jälkeen JMI:tä hyödyntäneestä menetelmästä. Tämän testin tarkemmat tulokset on esitetty taulukon 6 toisella tulosrivillä (PCA) ja taulukon 7 toisessa tulossarakkeessa (Ulottuvuuksien vähennys PCA:lla) sekä liitteen C taulukoissa 25 ja 26.

Taulukko 6: *K*-means: Ajankäyttö.

Ominaisuuksien valinta	**Opetusvaiheen ajankäyttö luottamusvälillä 99 %**	**Testausvaiheen ajankäyttö luottamusvälillä 99 %**
JMI	$17,72 \pm 1,83$ s	$2,67 \pm 0,02$ s
PCA	$59,53 \pm 7,98$ s	$2,79 \pm 0,01$ s

Taulukko 7: K-means: Menetelmän arviointi testausdatalla.

	Ominaisuuksien valinta JMI:llä	**Ulottuvuuksien vähennys PCA:lla**
	Keskiarvo luottamusvälillä 99 %	**Keskiarvo luottamusvälillä 99 %**
FPR	$0,1366 \pm 0,0269$	$0,1910 \pm 0,0204$
TNR	$0,8634 \pm 0,0269$	$0,8090 \pm 0,0204$
Osuvuus kahdella luokalla	$0,9036 \pm 0,0074$	$0,8826 \pm 0,0061$
Osuvuus kymmenellä ryhmällä	0,6771	0,6318
Recall	$0,9357 \pm 0,0065$	$0,9279 \pm 0,0044$
Precision	$0,9244 \pm 0,0181$	$0,8987 \pm 0,0142$
F-measure	$0,9276 \pm 0,0066$	$0,9116 \pm 0,0055$
Kynnysarvo	$0,3610 \pm 0,0278$	$0,3833 \pm 0,0191$

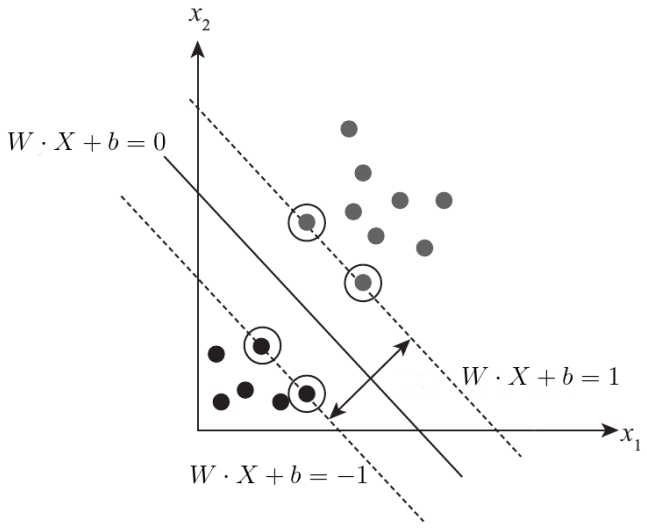
7 Tukivektorikone

Tukivektorikone eli SVM (*support vector machine*) on ohjattu menetelmä, jota voidaan hyödyntää ainakin luokittelussa sekä jatkuvien muuttujien arvoja ennustavassa regressiossa (Kantardzic 2011, luku 4.5). Tässä tutkielmassa tarkastellaan menetelmän käyttöä sekä kahden luokan luokitteluongelmassa että moniluokkaisessa luokittelussa. Menetelmä esitellään tarkemmin luvussa 7.1 ja toteutetaan kahdella eri tavalla luvuissa 7.2, 7.3 ja 7.4. SVM toteutettiin LIBSVM-kirjaston (Chang ja Lin 2011) version 3.21 Matlab-laajennoksella, joka tulee kirjaston mukana.

7.1 Menetelmän kuvaus

SVM:n perustapaus on tilanne, jossa kaksi luokkaa ovat lineaarisesti separoituvia. Esimerkki tällaisesta tapauksesta on kuviossa 20 esitetty kaksiulotteinen data. Aidoissa käyttöympäristöissä ei tällaista tilannetta kuitenkaan käytännössä tule vastaan, ja dataa kutsutaan lineaarisesti ei-separoituvaksi tai epälineaarisesti separoituvaksi. Tällöin opetusdatajoukko kuvataan epälineaarisesti alkuperäistä ulottuvuuksien määrää suurempaan avaruuteen. Tämän jälkeen SVM pyrkii löytämään datajoukolle sellaisen hypertason, jossa luokat toisistaan erottava marginaali on suurin mahdollinen. Tätä hypertasoa kutsutaan suurimman marginaalin hypertasoksi, jota kuviossa 20 merkitään $W \cdot X + b = 0$. Merkinnässä W on painotukset n:lle ominaisuudelle ilmaiseva vektori $W = \{w_1, w_2, \ldots, w_n\}$, X opetusdatan instanssi ja b skalaarinen korjauskerroin. Tukivektoreiksi kutsutaan niitä pisteitä, jotka kuviossa 20 sijaitsevat suurimman marginaalin hypertason molemmilla puolilla olevilla kahdella hypertasolla (Han ja Kamber 2006, luku 6.7.1). Esimerkkikuviossa nämä pisteet on ympyröity.

Ei-separoituvan datan tapauksessa joudutaan usein hyväksymään, että dataa ei saada täydellisesti luokiteltua oikein. Tällöin on tavoitteena löytää kompromissi, jolla saadaan minimoitua väärien luokittelujen määrä (Tan, Steinbach ja Kumar 2006, luku 5.5.3). C-SVC (*C-support vector classification*) eli joustavan marginaalin luokitin pyrkii saavuttamaan tämän lisäämällä optimointiongelmaan ylimääräisiä luokitteluvirheet huomioivia muuttujia. Merkitään opetusdatan instansseja $x_i \in R^n, i = 1, \ldots, l$ ja instanssien nimiöintitiedot sisältävää



Kuvio 20: SVM lineaarisesti separoituvalle datalle (Kantardzic 2011).

vektoria $y \in R^l$, joka jakaa instanssit kahteen luokkaan. Tässä on $y_i \in \{1,-1\}$. C-SVC pyrkii minimoimaan muuttujien W, b ja ξ arvot kaavassa

$$\frac{1}{2}W^T W + C\sum_{i=1}^{l}\xi_i \tag{7.1}$$

siten, että $y_i(W^T\phi(x_i)+b) \geq 1-\xi_i$ ja $\xi_i \geq 0, i = 1,\ldots,l$. Näissä $\phi(x_i)$ kuvaa x_i:n ulottuvuusmäärältään suurempaan avaruuteen ja $C > 0$ on säädettävä regularisointitermi (Chang ja Lin 2011). Koska vektori W voi sisältää paljon ulottuvuuksia, pyrkii LIBSVM-kirjasto ratkaisemaan ongelman minimoimalla α:n kaavassa

$$\frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{7.2}$$

siten, että $y^T\alpha = 0$ ja $0 \leq \alpha_i \leq C, i = 1,\ldots,l$. Kaavassa 7.2 on e pelkkiä ykkösiä sisältävä vektori $e = [1,\ldots,1]^T$ ja $Q_{l\times l}$ positiivisesti semidefiniitti matriisi, missä $Q_{ij} \equiv y_i y_j K(x_i,x_j)$. Edellisessä $K(x_i,x_j)$ on ydinfunktio $K(x_i,x_j) \equiv \phi(x_i)^T\phi(x_j)$ (Chang ja Lin 2011). Tästä saadaan C-SVC:n päätösfunktioksi

$$\mathrm{sgn}(W^T\phi(x)+b) = \mathrm{sgn}(\sum_{i=1}^{l} y_i\alpha_i K(x_i,x)+b). \tag{7.3}$$

Ydinfunktioita käytetään muuntamaan lineaarisesti ei-separoituva data alkuperäistä ulottuvuusjoukkoa suurempaan avaruuteen. Tavoitteena on saada data muunnettua siten, että se voidaan uudessa avaruudessa lineaarisesti separoida (Tan, Steinbach ja Kumar 2006, luku 5.5.4). SVM:ssä käyttökelpoisia ydinfunktioita on olemassa useita. Tässä tutkielmassa käytetään Gaussin ydinfunktiota (*Gaussian radial basis kernel*), joka määritellään LIBSVM:ssä (Chang ja Lin 2011)

$$K(x_i, x_j) = e^{-\gamma \left\| x_i - x_j \right\|^2}, \tag{7.4}$$

missä γ on käyttäjän määrittämä parametri.

Alun perin SVM on suunniteltu binääriseen luokitteluun, mutta sitä on laajennettu mahdollistamaan myös moniluokkaisten ongelmien ratkaisu (Hsu ja Lin 2002). LIBSVM hyödyntää monen luokan luokittelussa yksi yhtä vastaan -menetelmää (*one-against-one method*) (Chang ja Lin 2011). Kun k on luokkien määrä, käytetään menetelmässä $k(k-1)/2$ kappaletta kaksiluokkaisia luokittelijoita (Hsu ja Lin 2002).

7.2 Datan esikäsittely

Kuten tehtiin k-meansin tapauksessa luvussa 6.2, poistettiin UNSW-NB15-datan opetus- ja testausdatajoukoista ensimmäiseksi rivit yksilöivä *id*-arvo. Myös kaksi nimiöintitiedot sisältävää ominaisuutta otettiin datajoukoista erilleen. Opetusdatajoukon kategoriset ominaisuudet muunnettiin binäärisiksi pienin muutoksin valmiilla *categorical2bins*-funktiolla (Weidenbaum 2012).

Esimerkiksi opetusdatajoukon *state*-ominaisuuden muuttujat sisälsivät alunperin seitsemän mahdollista kategorista nominaalista arvoa: *ACC*, *CLO*, *CON*, *FIN*, *INT*, *REQ* ja *RST*. Binäärimuunnoksen jälkeen data esitettiin seitsemällä uudella ominaisuudella, nimiltään *state_ACC*, *state_CLO*, *state_CON*, *state_FIN*, *state_INT*, *state_REQ* ja *state_RST*. Mahdolliset muuttujien arvot näille ominaisuuksille olivat siten joko 0 tai 1. Yksi ominaisuus siis jaettiin tällä menetelmällä seitsemään uuteen ominaisuuteen. Vastaavalla tavalla suoritettiin muunnos datajoukon muillekin enemmän kuin kaksi arvoa sisältäneille nominaalisille omi-

naisuuksille, jotka olivat *proto*, *service*, *is_ftp_login* ja *ct_ftp_cmd*. Muunnosten jälkeen alkuperäiset ominaisuudet poistettiin datajoukosta. Ennen muunnoksia datajoukossa oli ollut eri ominaisuuksia 42 kappaletta, ja muunnosten jälkeen ominaisuuksien määrä kasvoi näin 194:ään.

Kaikki datajoukkojen ei-binäärisiä muuttujien arvoja sisältävät ominaisuudet normalisoitiin minimi-maksimi-menetelmällä (ks. kaava 4.8) välille $[0,1]$. LIBSVM:n tekijät suosittelevat arvojen normalisointia joko välille $[-1,1]$ tai $[0,1]$ (Hsu, Chang ja Lin 2016). Normalisointi suoritettiin samalla tavalla kuin k-meansin tapauksessa luvussa 6.4, eli ristiinvalidoinnin aikana validointidata normalisoitiin opetusdatasta saatujen minimi- ja maksimiarvojen pohjalta.

7.3 Parametrien etsintä

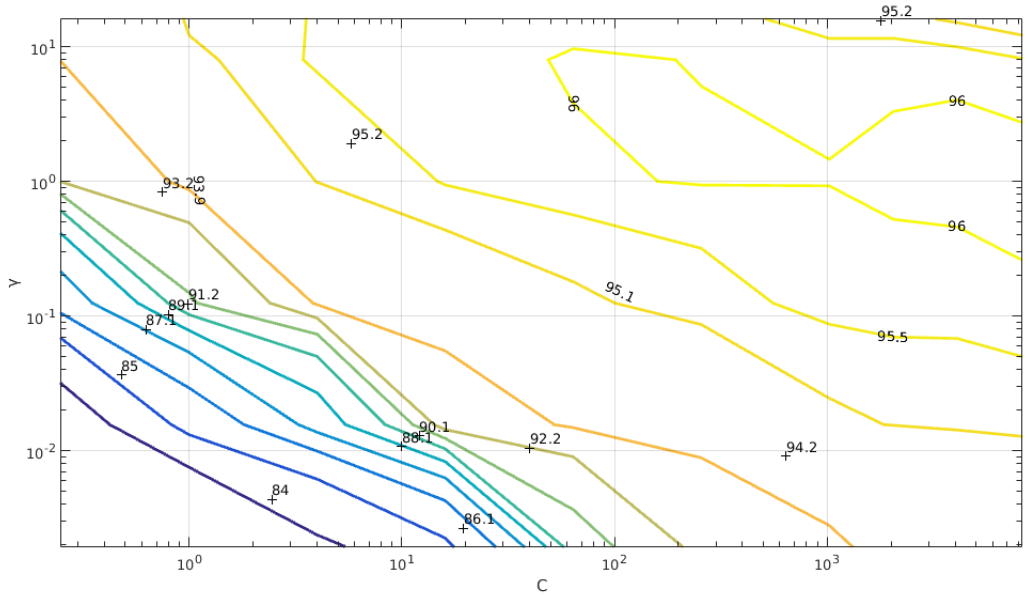
Kun käytetään joustavan marginaalin luokitinta eli C-SVC:tä ja Gaussin ydinfunktiota, vaaditaan alkuparametreiksi arvot C ja γ. Näiden parametrien etsintä suoritettiin ensin kaksiluokkaiselle SVM:lle, minkä jälkeen tarkasteltiin moniluokkaisen SVM:n tapausta.

7.3.1 Kaksiluokkainen tukivektorikone

Optimaalisia parametreja C ja γ etsittiin kaksiluokkaiselle SVM:lle opetusdatan pohjalta viisinkertaisella ositetulla ristiinvalidoinnilla. Etsintä aloitettiin kasvattamalla kumpaakin parametria kasvavin harppauksin logaritmisesti, jotta sopivien parametrien suuruusluokka selviäisi mahdollisimman nopeasti. Parametrien hyvyyttä arvioitiin yksinkertaisella kaavassa 5.6 esitetyllä luokittelun osuvuuden ilmoittavalla mittarilla. Tuloksissa esitetyt arvot ovat osuvuuden keskiarvoja prosenttiosuuksina, eli kaavan tulokset on vielä kerrottu sadalla (Chang ja Lin 2011). Tässä vaiheessa huomioitiin ainoastaan kaksi luokkaa, jotka olivat normaali liikenne ja haittaliikenne.

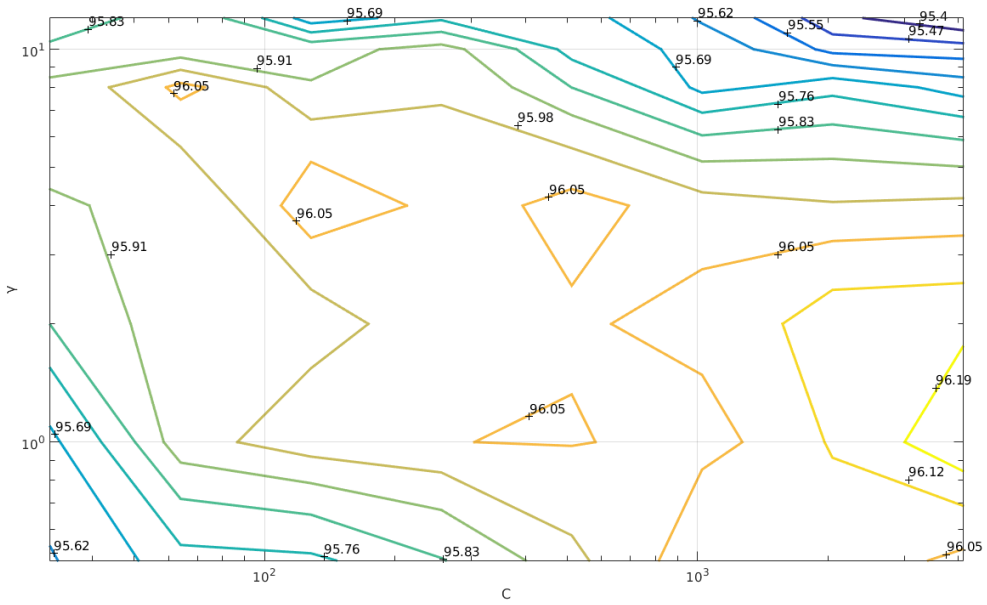
Ensimmäisen parametrien etsinnän tulokset on esitetty korkeuskäyrinä kuviossa 21. Kuvion pohjana olleet alkuperäiset mittaustulokset ovat liitteen D taulukossa 27. Tässä vaiheessa erot olivat vielä selkeitä, ja arvoja saatiin väliltä 83–96 %. Mittausten pohjalta päädyttiin tarkastelemaan toisella mittauksella tarkemmin noin 96 prosentin osuvuuden tuottavia parametrien

arvoja. Toisen parametrien etsinnän tulokset on esitetty kuviossa 22, ja alkuperäiset mittaustulokset esitetään liitteen D taulukossa 28. Tässä vaiheessa erot olivat jo huomattavasti pienempiä, ja mitatut arvot asettuivat välille 95,33–96,26 %. Alueen keskivaiheilta kuitenkin yhä erottui selkeästi alueita, joiden arvot ylittivät jatkuvasti 96 prosentin rajan. Suurilla C:n arvoilla saavutettiin tämän mittauksen parhaat arvot, ja myös nämä haluttiin ottaa mukaan viimeiseen etsintään.



Kuvio 21: Kaksiluokkainen SVM: γ- ja C-parametrien etsinnän ensimmäinen vaihe.

Kolmannessa vaiheessa tarkasteltiin myös SVM-mallin muodostamiseen ja sen pohjalta tehtyyn luokitteluun käytettyjä aikoja. Kuviossa 23 korkeuskäyrissä näkyvät ajat on ilmoitettu minuutteina. Ilmoitetut ajat on saatu jakamalla viisinkertaiseen ristiinvalidointiin käytetty aika viidellä, jolloin ajat ovat yksittäisten ajojen keskiarvoja. Ajat sisältävät *svmtrain*- ja *svmpredict*-funktioiden ajon sekä joitakin käytännössä vakioajan vieviä toimia – eniten aikaa vievimpänä normalisointi. Kuvion pohjana olleet alkuperäiset ajat esitetään liitteen D taulukoissa 31 ja 32. Liitteen taulukoissa käytetyt ajat on ilmoitettu sekunteina, ja ajat ovat koko viisinkertaiseen ristiinvalidointiin käytettyjä aikoja. Etsinnässä käytettiin LIBSVM:n Matlab-laajennoksen kahdelle suoritinytimelle käännettyä versiota, ja välimuistin kooksi *svmtrain*-funktiossa määriteltiin 512 megatavua. Oletuksena päällä oleva optimointiongelman pienentämiseen tähtäävä *shrinking*-asetus pidettiin käytössä.



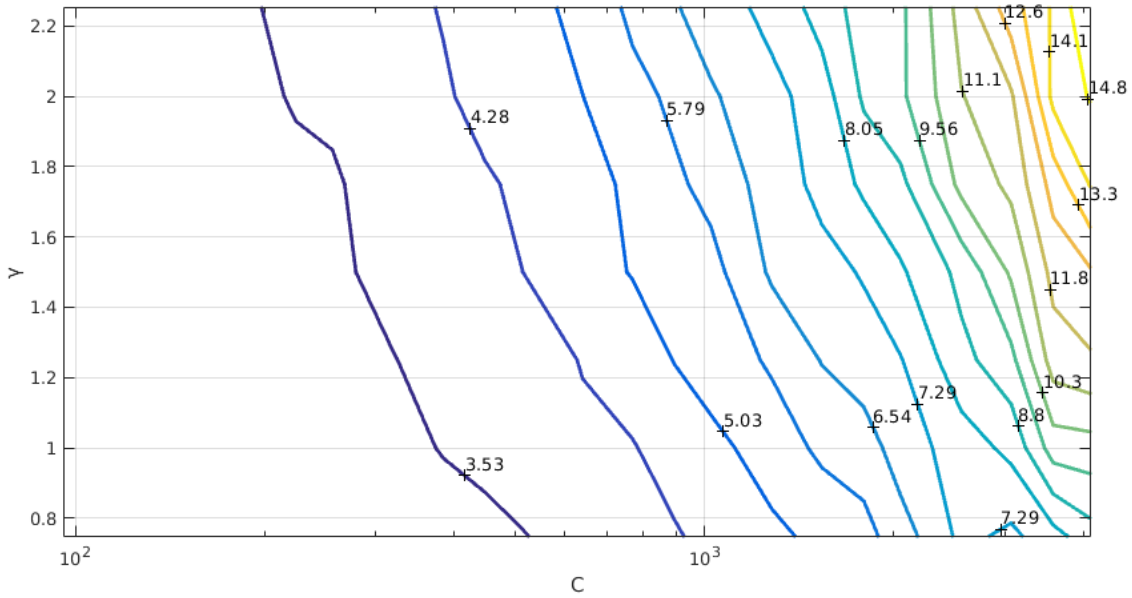
Kuvio 22: Kaksiluokkainen SVM: γ- ja C-parametrien etsinnän toinen vaihe.

Kuviosta 23 ilmenee, että sekä γ- että C-parametrin kasvattaminen pidentää SVM-mallin muodostamiseen käytettyä aikaa. Tietyssä vaiheessa lisäajankäytöstä ei kuitenkaan ole hyötyä mallin suorituskyvyn kannalta katsottuna. Kolmannen parametrien etsinnän osuvuusarvot on esitetty kuviossa 24. Tässä vaiheessa erot ovat jo hyvin pieniä, ja selkeitä vahvoja alueita ei kuviosta enää juurikaan erotu. Tiedettiin myös, että kuvion oikea yläkulma on laskennallisesti aikaa vievin, joten sitä haluttiin välttää. Lopulta päädyttiin kuvion 24 perusteella parametrien arvoihin $\gamma = 2$ ja $C = 768$. Tämä yhdistelmä tuotti opetusvaiheessa viisinkertaisella ositetulla ristiinvalidoinnilla osuvuuden keskiarvon 96,28 %.

7.3.2 Moniluokkainen tukivektorikone

Vastaavalla tavalla suoritettiin parametrien etsintä myös moniluokkaiselle SVM:lle. Ensimmäisessä etsinnässä haravoitiin samat arvot kuin kaksiluokkaisen SVM:n tapauksessa. Ensimmäisen vaiheen tulokset esitetään liitteen D kuviossa 33 ja kuvion pohjana olleet alkuperäiset arvot liitteen D taulukossa 33. Ensimmäisessä vaiheessa osuvuuden keskiarvot kymmenelle ryhmälle vaihtelivat välillä 71–86 %.

Toisessa etsintävaiheessa keskityttiin tutkimaan tarkemmin 85–86 prosentin osuvuuden tuottaneita parametrien arvoja. Vaiheen tulokset esitetään kuviossa 25 ja alkuperäiset arvot liit-



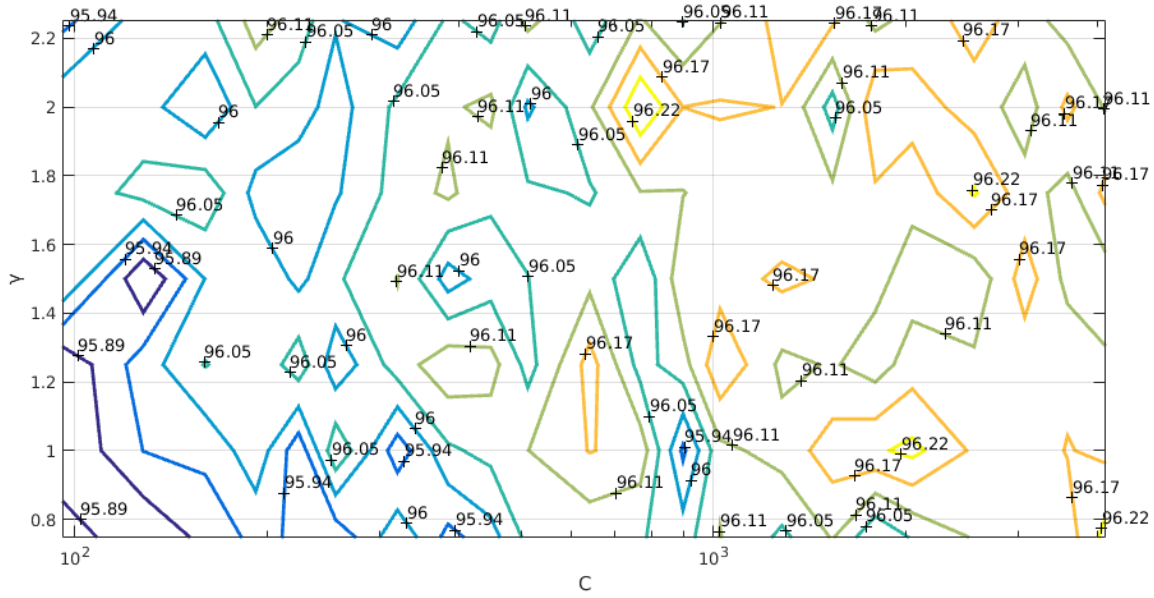
Kuvio 23: Kaksiluokkainen SVM: γ- ja C-parametrien etsinnän kolmannen vaiheen ajankäyttö.

teen D taulukossa 34.

Myös moniluokkainen SVM käytti sitä enemmän mallin muodostamiseen, mitä suuremmiksi C:n ja γ:n arvot kasvoivat. Lopullisiksi parametreiksi valittiin siis parhaan mahdollisen osuvuuden tuottaneet parametrit siten, että sekä C että γ olivat mahdollisimman alhaiset. Moniluokkaisen SVM:n tapauksessa päädyttiin kuvion 25 perusteella parametrien arvoihin $\gamma = 1,25$ ja $C = 1024$. Parametrien yhdistelmä tuotti opetusvaiheessa viisinkertaisella ositetulla ristiinvalidoinnilla kymmenen ryhmän osuvuudelle keskiarvon 85,56 %.

7.4 Menetelmän arviointi testausdatalla

Sopivien parametrien (kaksiluokkaiselle $\gamma = 2$, $C = 768$ ja moniluokkaiselle $\gamma = 1,25$, $C = 1024$) löydyttyä SVM-mallit muodostettiin uudelleen koko opetusdataa käyttäen. Muodostettuja malleja käytettiin tämän jälkeen ennustamaan luokat testausdatan instansseille. Ennen tätä muunnettiin testausdatan kategoriset ominaisuudet binäärisiksi luvussa 7.2 esitellyllä tavalla. Binäärimuunnoksen jälkeen erosivat opetus- ja testausdatajoukko hieman toisistaan. Tämä johtui siitä, että osaa muunnoksella muodostetuista ominaisuuksista ei välttä-



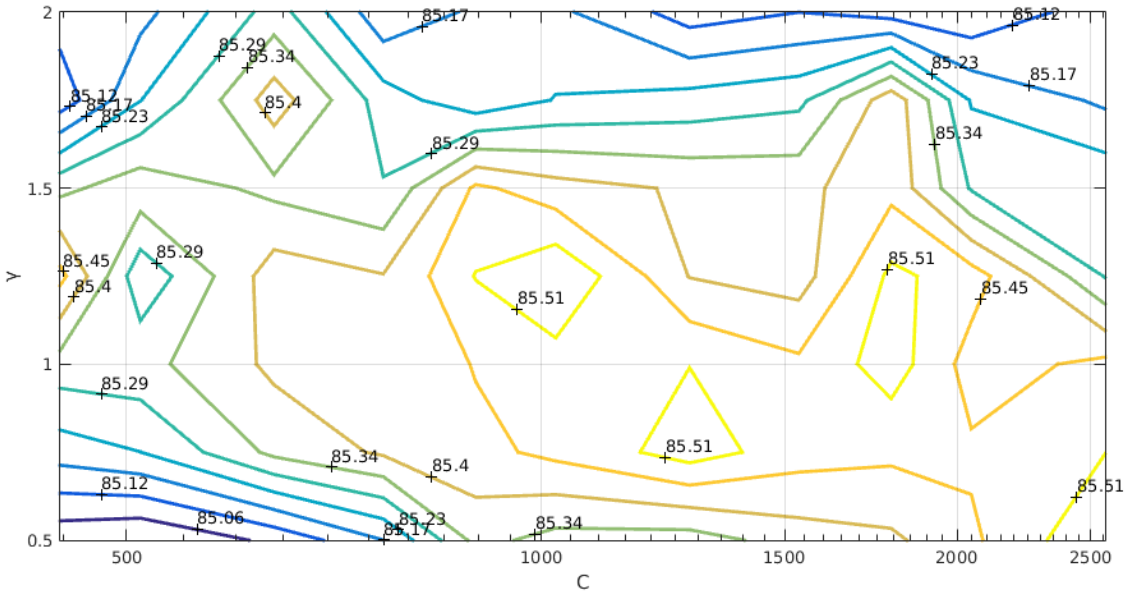
Kuvio 24: Kaksiluokkainen SVM: γ- ja C-parametrien etsinnän kolmas vaihe.

mättä ollut olemassa molemmissa datajoukossa. Datajoukoista tehtiin yhtenevät ominaisuusjoukon osalta poistamalla opetusdatasta ominaisuudet *state_ACC* ja *state_CLO* sekä testausdatasta ominaisuudet *proto_icmp*, *proto_rtp*, *state_ECO*, *state_PAR*, *state_URN*, *state_no*, *is_ftp_login_4* ja *ct_ftp_cmd_4*. Näiden poistojen jälkeen molemmissa datajoukoissa oli samat 192 ominaisuutta.

Kun algoritmi ajetaan samoilla parametreilla ja datajoukoilla, tuottaa se aina samat tulokset. Menetelmä ajettiin samoilla parametreilla ja datajoukoilla viisi kertaa, jotta LIBSVM:n SVM-mallin luontiin käytetyn *svmtrain*-funktion ja mallin pohjalta tehtävään ennustukseen käytetyn *svmpredict*-funktion suorituksiin kulunutta aikaa voitiin tarkastella paremmin. Ajankäyttöä yllä mainittujen kahden funktion osalta tarkastellaan taulukossa 8. Ilmoitetut ajat ovat viiden ajon pohjalta laskettuja keskiarvoja.

Taulukko 8: SVM: LIBSVM-funktioiden ajankäyttö.

SVM:n tyyppi	**Ajankäyttö (svmtrain)**	**Ajankäyttö (svmpredict)**
Kaksiluokkainen	9 min 18 s	9 min 0 s
Moniluokkainen	10 min 27 s	24 min 39 s



Kuvio 25: Moniluokkainen SVM: γ- ja C-parametrien etsinnän toinen vaihe.

Eri ryhmien luokittelun osuvuutta kaksiluokkaisessa SVM:ssä tarkastellaan taulukossa 9. Kuten k-means-osuuden taulukon 4 ensimmäisessä tulossarakkeessa, huomioidaan kunkin hyökkäysryhmän kohdalla ainoastaan kuinka suuri osa ryhmän instansseista on luokiteltu oikein haittaliikenteeksi.

Taulukko 9: Kaksiluokkainen SVM: Luokittelun osuvuus kahdella pääluokalla.

Luokka	**Ryhmä**	**Osuvuus**
Normaali	**Normal**	0,9686
Haitta-liikenne	**Analysis**	0,7735
	Backdoor	0,9748
	DoS	0,9701
	Exploits	0,9560
	Fuzzers	0,2691
	Generic	0,9984
	Reconnaissance	0,8745
	Shellcode	0,8305
	Worms	0,9231

Moniluokkaisen SVM:n luokittelun osuvuutta tarkastellaan taulukossa 10. Ensimmäinen osuvuussarake ilmoittaa kuinka suuri osa kyseisen ryhmän instansseista on luokiteltu oikein pääluokan (normaali tai haittaliikenne) osalta. Toinen osuvuussarake taas kertoo, kuinka suuri osa ryhmän instansseista on luokiteltu täsmälleen oikeaan ryhmään.

Taulukko 10: Moniluokkainen SVM: Luokittelun osuvuus.

Luokka	**Ryhmä**	**Osuvuus kahdella pääluokalla**	**Osuvuus kymmenellä ryhmällä**
Normaali	**Normal**	0,9769	0,9769
Haitta-liikenne	**Analysis**	0,7660	0
	Backdoor	0,9427	0,0092
	DoS	0,9600	0,6307
	Exploits	0,9400	0,5818
	Fuzzers	0,2102	0,0730
	Generic	0,9976	0,9806
	Reconnaissance	0,8871	0,6263
	Shellcode	0,7114	0,2868
	Worms	0,8846	0,0923

Taulukossa 11 esitetään testausdatajoukon instanssien ennustetut ja todelliset ryhmät. Taulukon riveillä on esitetty todelliset ryhmät ja sarakkeilla ennustetut ryhmät. Taulukossa diagonaalisesti esitetyt tummennetut arvot edustavat siten oikeiden luokittelujen lukumääriä. Lopuksi taulukossa 12 esitetään kaksiluokkaisen ja moniluokkaisen SVM:n mittausten tulokset 6–7 mittarilla.

Kokonaisuutena SVM:n opetusvaiheessa havaitut osuvuusprosentit laskivat selvästi varsinaisessa testausvaiheessa. Kaksiluokkaisella SVM:llä saavutettiin opetusvaiheessa 96,28 %:n osuvuus, mutta samoilla parametreilla laski testausvaiheessa osuvuus 89,18 prosenttiin. Moniluokkainen SVM heikensi tulostaan vielä selvemmin, sillä opetusvaiheessa saavutettu 85,56 % laski testausvaiheessa 73,77 prosenttiin.

Taulukko 11: Moniluokkainen SVM: Instanssien ennustetut ja todelliset ryhmät.

		Ennustettu ryhmä									
		Normal	***Analysis***	***Backdoor***	***DoS***	***Exploits***	***Fuzzers***	***Generic***	***Reconnaissance***	***Shellcode***	***Worms***
Todellinen ryhmä	**Normal**	**54708**	0	1	115	693	333	13	125	12	0
	Analysis	468	**0**	0	1153	348	0	29	2	0	0
	Backdoor	100	0	**16**	1148	396	16	23	45	2	0
	DoS	491	4	10	**7735**	3418	81	306	162	53	4
	Exploits	2005	4	16	10296	**19428**	293	518	757	64	12
	Fuzzers	14361	0	4	1151	935	**1328**	68	269	67	1
	Generic	95	1	2	279	353	17	**39225**	20	5	3
	Reconn-aissance	1184	0	0	1379	1196	92	45	**6571**	12	12
	Shellcode	327	0	0	2	94	25	10	349	**325**	1
	Worms	15	0	0	2	89	1	1	5	5	**12**

Taulukko 12: SVM: Menetelmän arviointi testausdatalla.

SVM:n tyyppi	**FPR**	**TNR**	**Osuvuus kahdella luokalla**	**Osuvuus kymmenellä ryhmällä**	**Recall**	**Precision**	**F-measure**
Kaksi-luokkainen	0,2409	0,7590	0,8918	-	0,9831	0,8558	0,9150
Moni-luokkainen	0,2582	0,7418	0,8840	0,7377	0,9873	0,8404	0,9079

8 Keinotekoiset neuroverkot

Keinotekoiset neuroverkot (*artificial neural networks* eli *ANN*) pyrkivät mallintamaan biologisten hermojärjestelmien toimintaa (Rojas 1996, luku 1.1). Neuroverkko koostuu yksittäisistä neuroneista, jotka ottavat vastaan annetun syötteen. Neuronit tuottavat syötteen perusteella tiedosta eräänlaisen koosteen, joka edelleen välitetään eteenpäin verkossa. Syötteen eri osille taas on neuronissa määritelty painoarvoja, joita säädetään sopiviksi neuroverkon opetusvaiheessa (Rojas 1996, luku 1.3; Han ja Kamber 2006, luku 6.6).

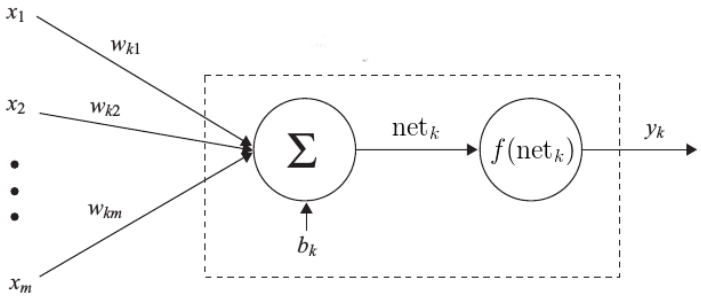
Tässä tutkielmassa keinotekoisen neuroverkon menetelmät toteutetaan Matlabin *Neural network toolbox* -lisäosan avulla. Luvussa 8.1 kuvataan yleisesti aihepiirin käsitteitä ja esitellään tarkemmin kaksi käytettävää oppimismenetelmää. Käytettävät oppimismenetelmät valitaan luvussa 8.2 ja ne arvioidaan testausdatalla luvussa 8.3.

8.1 Menetelmän kuvaus

Yksittäisen keinotekoisen neuronin k malli on esitetty kuviossa 26. Merkitään syötettä $X = \{x_0, x_1, x_2, \ldots, x_m\}$. Jokaisella joukon X alkiolla eli syötteen osalla on painoarvo w_{ki}, ja painoarvojoukkoa merkitään $W = \{w_{k0}, w_{k1}, w_{k3}, \ldots, w_{km}\}$ (Kantardzic 2011, luku 7.1). Kuviossa b_k on ulkopuolinen painotuskerroin, jonka voidaan katsoa olevan painoarvo instanssin nimiöintitiedolle. Tällöin voidaan merkitä $b_k = w_{k0}$, jolloin x_0 on varsinainen nimiöinnin arvo (Haykin 2009, luku 1.3; Kantardzic 2011). Tämän jälkeen neuronin ensimmäisen vaiheen ulostuloksi net_k saadaan (Kantardzic 2011)

$$\text{net}_k = \sum_{i=0}^{m} x_i w_{ki} = X \cdot W. \tag{8.1}$$

Neuronilla on siirtofunktio (*activation function* tai *transfer function*) f, joka kuvaa sille tulevan syötteen net_k tietylle välille (Han ja Kamber 2006, luku 6.6). Tässä tutkielmassa käytetään neuroverkon opetuksessa vastavirtapohjaisia (*backpropagation*) algoritmeja, jotka vaativat siirtofunktion olevan differentioituva ja jatkuva (Rojas 1996, luku 7.1). Siirtofunktiona voidaan vastavirta-algoritmin yhteydessä käyttää esimerkiksi hyperbolista tangenttia, jolloin



Kuvio 26: Keinotekoisen neuronin malli (Kantardzic 2011).

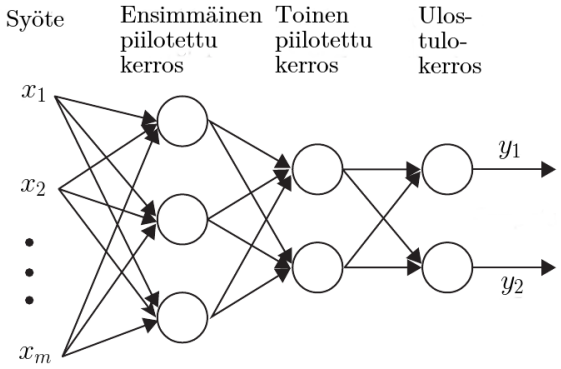
neuronille k määritellään ulostulo y_k

$$y_k = \tanh(\text{net}_k) = \frac{e^{\text{net}_k} - e^{-\text{net}_k}}{e^{\text{net}_k} - e^{-\text{net}_k}}, \tag{8.2}$$

jonka arvo on välillä $[-1,1]$ (Kantardzic 2011, luku 7.1). Vastavirta-algoritmissa verkon virhetietoa kuljetetaan verkossa taaksepäin ulostulokerrokselta syötekerrokseen päin (Rojas 1996, luku 7.2).

Neuroneista koostuvan neuroverkon malli on esitetty kuviossa 27. Neuroverkko voi olla joko yksitasoinen tai monitasoinen. Yksitasoisessa neuroverkossa syötekerros on kytketty suoraan ulostulokerroksen neuroneihin, kun taas monitasoisessa neuroverkossa on yksi tai useampi piilotettu kerros syötekerroksen ja ulostulokerroksen välissä. Kuviossa 27 esitetyssä verkossa on kaksi piilotettua kerrosta, joten se on monitasoinen neuroverkko. Tarkemmin ottaen kyseinen verkko on myös myötäkytketty (*feedforward*) neuroverkko, sillä se ei sisällä takaisinkytkentöjä aikaisempiin neuroneihin (Haykin 2009, luku 1.6). Takaisinkytketty (*feedback*) neuroverkko sisältää vähintään yhden takaisinkytkennän aikaisempaan neuroniin (MacKay 2005, luku 42). Verkko on myös täysin kytketty, sillä jokaisen kerroksen kaikki neuronit on yhdistetty seuraavan kerroksen kaikkiin neuroneihin (Haykin 2009).

Verkon virhettä mittaavana kustannusfunktiona $V(x)$ käytetään tässä tutkielmassa keskineliövirhettä (*mean squared error* eli *MSE*). Kun neuroverkossa on K kappaletta ulostuloja ja P kappaletta opetusdatan instansseja, määritellään keskineliövirhe (Ampazis ja Perantonis 2000)



Kuvio 27: Keinotekoisen neuroverkon malli (Kantardzic 2011).

$$V(x) = \frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{K}(d_i^p - y_i^p)^2, \tag{8.3}$$

missä y_i^p sisältää verkon ennustamat arvot ja d_i^p tavoitearvot.

Tässä tutkielmassa käytetyt oppimismenetelmät pohjautuvat vastavirta-algoritmiin. Kaikki tutkielman neuroverkot ovat myös monitasoisia, myötäkytkettyjä sekä täysin kytkettyjä. Siirtofunktiona käytetään hyperbolista tangenttia. Seuraavissa luvuissa esitellään tarkemmin kaksi käytettyä menetelmää, jotka ovat Levenbergin-Marquardtin algoritmi sekä bayesiläinen regularisointi (*Bayesian regularization*).

8.1.1 Levenbergin-Marquardtin algoritmi

Levenbergin-Marquardtin algoritmi on Gauss-Newton-nimisen algoritmin muunnos (Hagan ja Menhaj 1994). Laskennassa tarvitaan teoriassa Hessen matriisia $\nabla^2 V(x)$, jonka laskeminen on usein aikavaativuudeltaan raskasta (Haykin 2009, luku 4.16). Koska käytetty kustannusfunktio keskineliövirhe kaavassa 8.3 on muodoltaan neliöiden summa, voidaan Hessen matriisi kuitenkin kirjoittaa muotoon (Ampazis ja Perantonis 2000; Hagan ja Menhaj 1994)

$$\nabla^2 V(x) = J^T(x)J(x) + S(x), \tag{8.4}$$

missä $J(x)$ on aikavaativuudeltaan Hessen matriisin laskemiseen verrattuna kevyempi Jacobin matriisi

$$J(x) = \begin{bmatrix} \frac{\partial v_1(x)}{\partial x_1} & \frac{\partial v_1(x)}{\partial x_2} & \cdots & \frac{\partial v_1(x)}{\partial x_n} \\ \frac{\partial v_2(x)}{\partial x_1} & \frac{\partial v_2(x)}{\partial x_2} & \cdots & \frac{\partial v_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial v_N(x)}{\partial x_1} & \frac{\partial v_N(x)}{\partial x_2} & \cdots & \frac{\partial v_N(x)}{\partial x_n} \end{bmatrix} \tag{8.5}$$

ja $S(x) = \sum_{i=1}^{N} v_i(x)\nabla^2 v_i(x)$. Gaussin-Newtonin algoritmissa on $S(x) \approx 0$. Matriisissa 8.5 ja $S(x)$:n määritelmässä $v(x)$ on virheen ilmaiseva vektori. Gradientti $\nabla V(x)$ voidaan kirjoittaa muotoon

$$\nabla V(x) = J^T(x)v(x). \tag{8.6}$$

Verkossa käytössä oleville painotuskertoimille x määritellään Levenbergin-Marquardtin algoritmissa optimaaliset uudet arvot erotuksena Δx, joka määritellään (Hagan ja Menhaj 1994; Saini ja Soni 2002)

$$\Delta x = [J^T(x)J(x) + \mu I]^{-1} J^T(x)v(x), \tag{8.7}$$

missä I on $J(x)$:n kanssa yhtä monta ulottuvuutta sisältävä yksikkömatriisi. μ on säädettävä parametri, jota etsinnän aikana joko kasvatetaan tai pienennetään. Jokaisen kierroksen jälkeen lasketaan uudelleen kustannusfunktion $V(x)$ arvo. Mikäli uudet painokertoimet ovat heikentäneet verkon suorituskykyä eli kustannusfunktion arvo on kasvanut, kerrotaan parametrin μ arvo kertoimella $\beta > 1$. Jos kustannusfunktion arvo on laskenut, jaetaan parametrin μ arvo β:lla (Hagan ja Menhaj 1994; Saini ja Soni 2002). Algoritmi saavuttaa tavoitteensa, kun kaavan 8.6 gradientin normi tai kustannusfunktion $V(x)$ arvo kaavassa 8.3 alittaa aiemmin määritellyn tavoitearvon (Hagan ja Menhaj 1994). Levenbergin-Marquardtin algoritmi ei takaa globaalia optimiarvoa ongelmaan, vaan se voi aloitusarvoista riippuen päätyä myös paikalliseen minimiarvoon (Mukherjee ja Routroy 2012).

Gaussin-Newtonin algoritmi on muuten kaavan 8.7 mukainen, mutta siinä ei esiinny termiä μI. Pienillä μ:n arvoilla Levenbergin-Marquardtin algoritmi vastaa siis Gaussin-Newtonin algoritmia. Toisaalta mitä suuremmaksi μ:n arvoa kasvatetaan, sitä enemmän algoritmi alkaa

muistuttaa gradienttimenetelmää (*gradient descent* tai *steepest descent*) (Hagan ja Menhaj 1994; Saini ja Soni 2002). Levenbergin-Marquardtin algoritmissa termi μI huolehtii siitä, että matriisi $J^T(x)J(x)$ on aina kääntyvä (Saini ja Soni 2002).

8.1.2 Bayesiläinen regularisointi

Regularisoinnilla yleisesti pyritään estämään opetettavan mallin ylisovitus (*overfitting*) (Foresee ja Hagan 1997; Haykin 2009, luku 7.1). Ylisovitettu malli on opetusdatan avulla oppinut kiinnittämään huomiota liian pieniin opetusdatan yksityiskohtiin. Samalla malli on kehittynyt tarpeettoman monimutkaiseksi. Tällainen malli näyttää tuottavan hyviä tuloksia opetusdatalla, mutta toimii huomattavasti heikommin yleistettynä, erillisellä testausdatalla (MacKay 2005, luku 44.4; 1992b).

Regularisointi lisää käytettävään kustannusfunktioon E_D ylimääräisen kerroinparametrin β. Lisäksi otetaan käyttöön kokonaan uusi termi αE_W, jolloin uusi minimoitava kohdefunktio F määritellään (Foresee ja Hagan 1997)

$$F = \beta E_D + \alpha E_W, \tag{8.8}$$

missä E_W on jokin verkon monimutkaisuudesta rankaiseva funktio, esimerkiksi verkon painoarvojen neliöiden summa $E_W = \frac{1}{2}\sum_i w_i^2$. Sekä α että β ovat siten säädettäviä parametreja, joille bayesiläisen oppimisen aikana haetaan optimaalisia arvoja (MacKay 1992a; Foresee ja Hagan 1997).

Matlab käyttää bayesiläisen regularisoinnin toteutusta, jossa hyödynnetään Levenbergin-Marquardtin algoritmia ja Gaussin-Newtonin menetelmästä tuttua Hessen matriisin kaavassa 8.4 esitettyä approksimaatiota ("MathWorks documentation: Bayesian regularization backpropagation" 2016), joka tässä yhteydessä saa muodon (Foresee ja Hagan 1997)

$$H = 2\beta J^T J + 2\alpha I_N. \tag{8.9}$$

Bayesiläisellä käsitteistöllä ilmaistuna optimaaliset painoarvot w verkolle saadaan, kun maksimoidaan posterioritodennäköisyys Bayesin säännöllä

$$P(w|D,\alpha,\beta,M) = \frac{P(D|w,\beta,M)P(w|\alpha,M)}{P(D|\alpha,\beta,M)}, \tag{8.10}$$

mikä vastaa kaavan 8.8 kohdefunktion F minimoimista (Foresee ja Hagan 1997; MacKay 1992b). Kaavassa 8.10 on D käytetty datajoukko ja M käytetty neuroverkkomalli. $P(D|w,\beta,M)$ on uskottavuusfunktio, joka edustaa datan esiintymisen todennäköisyyttä painoarvoilla w. $P(w|\alpha,M)$ on tietämyksen painoarvoista ilmaiseva prioritiheys. $P(D|\alpha,\beta,M)$ takaa normalisointitekijänä, että kokonaistodennäköisyys on suuruudeltaan 1 (Foresee ja Hagan 1997).

Kun Bayesin sääntöä sovelletaan parametrien α ja β optimointiin, saadaan laskettua verkon merkityksellisten parametrien kokonaislukumäärä γ kaavalla (Foresee ja Hagan 1997)

$$\gamma = N - 2\alpha \mathrm{tr}(H)^{-1}, \tag{8.11}$$

missä N on verkon parametrien kokonaislukumäärä ja H kaavassa 8.9 esitetty Hessen matriisin approksimaatio. Arvot parametreille α ja β saadaan tämän jälkeen kaavoista (Foresee ja Hagan 1997)

$$\alpha = \frac{\gamma}{2E_W(w)} \qquad \beta = \frac{n-\gamma}{2E_D(w)}. \tag{8.12}$$

Algoritmia kutsutaan nimellä GNBR (*Gauss-Newton approximation to Bayesian regularization*), ja se esitetään algoritmilistauksessa 2 (Foresee ja Hagan 1997).

Algoritmi 2 GNBR-algoritmi

1: valitse aloitusarvot parametreille α ja β sekä verkon painoarvot w
2: **toista**
3: minimoi kaavan 8.8 kohdefunktio Levenbergin-Marquardtin algoritmilla
4: laske lukumäärä verkon merkityksellisille parametreille kaavalla 8.11
5: laske uudet arvot parametreille α ja β kaavoilla 8.12
6: **kunnes** algoritmi konvergoituu

8.2 Oppimismenetelmien valinta

Neuroverkon oppimismenetelmien vertailun alkuvaiheessa havaittiin, että kaikkea datajoukon opetusdataa ei voida neuroverkon opetuksessa käytännössä kerralla hyödyntää. Esimerkiksi Levenbergin-Marquardtin algoritmin opetus kymmenillätuhansilla instansseilla vie muistia käytetyn koneen resursseihin nähden liikaa, ja datamäärän kasvaessa myös opetukseen kuluu enemmän aikaa. Näin ollen menetelmien vertailuvaiheessa algoritmien opetuksessa käytettiin 6175 instanssia, jotka poimittiin opetusdatasta ositetulla otannalla.

Otannan jälkeen neuroverkko rakennettiin Matlabin *Neural network toolbox* -lisäosan kuvioiden tunnistamiseen tarkoitetulla *patternnet*-funktiolla. Poimittu data jaettiin satunnaisesti kolmeen osaan, joista 70 prosenttia käytettiin varsinaiseen neuroverkon opetukseen, 15 prosenttia opetuksen aikana mahdollisesti suoritettavaan verkon validointiin ja 15 prosenttia testaukseen. Bayesiläinen regularisointi ei käytä opetuksen aikana tässä valittua validointidataa. Ilmoitetut tulokset perustuvat kymmenen ajon keskiarvoihin, ja ositettu otanta suoritettiin koko opetusdatasta kullekin ajolle aina uudestaan.

Kaikissa algoritmeissa käytettiin opetusvaiheessa maksimi-iterointikierrosten määränä tuhatta kappaletta, ja muinakin oppimismenetelmien parametreina pidettiin Matlabin oletusasetuksia. Säädettävänä parametrina tarkasteltiin oppimismenetelmissä neuroverkon piilotettujen kerrosten määrää. Kerrosten määrä oli Levenbergin-Marquardtin algoritmissa, Bayesin regularisoinnissa ja BFGS-algoritmissa 1–10 kappaletta. Muut menetelmät olivat huomattavasti näitä kolmea menetelmää nopeampia, joten tarkasteluun voitiin ottaa menetelmien suorituskyky 1–30 piilotetulla kerroksella. Tarkasteltavina mittareina olivat keskineliövirhe sekä opetukseen käytetty aika. Osion mittaustuloksille laskettiin 95 %:n luottamusväli, ja kuvioissa tämä luottamusväli on ilmaistu kunkin mittaustuloksen kohdalla pystypalkkina.

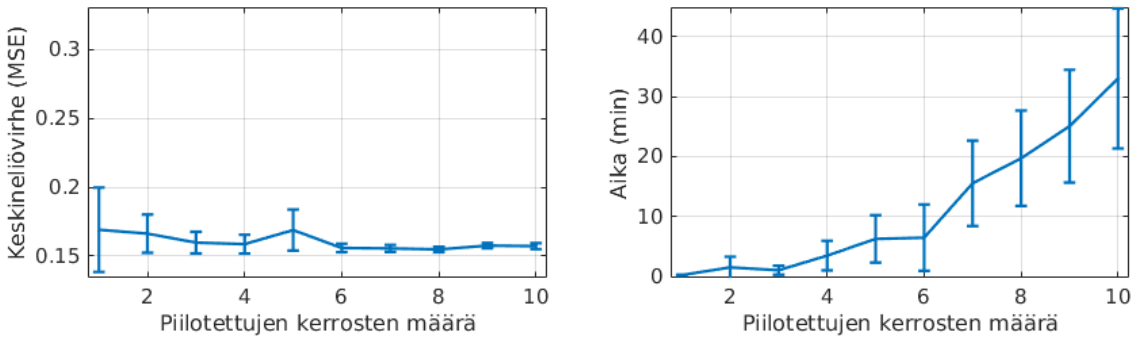
Data esikäsiteltiin normalisointia lukuun ottamatta samoin kuin tukivektorikoneen tapauksessa luvussa 7.2. Tässä vaiheessa tehtiin myös luvussa 7.4 esitelty toimenpide, jossa opetus- ja testausdatajoukoista tehtiin valmiiksi yhtenevät ominaisuuksien osalta. Näiden toimenpiteiden jälkeen käytetyssä opetusdatassa oli 192 ominaisuutta.

Kaikissa algoritmeissa käytettiin normalisointimenetelmänä Matlabin *Neural network toolbox* -lisäosan sisäistä minimi-maksimi-normalisointia, joka normalisoi muuttujien arvot vä-

lille $[-1, 1]$ (“MathWorks documentation: Choose neural network input-output processing functions” 2016). Myös neuroverkon tuottamat arvot normalisoitiin samalla menetelmällä verkon ulostulokerroksen jälkeen. Tämän lisäksi testattiin vastaavilla otannoilla myös yhdistelmää, jossa data normalisoitiin ensin kaavassa 4.11 esitetyllä z-piste-normalisoinnilla, minkä jälkeen se toimitettiin Matlabin sisäiselle minimi-maksimi-toteutukselle. Tällä yhdistelmällä ja otannalla ei kuitenkaan havaittu merkittäviä vaikutuksia minkään oppimismenetelmän suorituskykyyn.

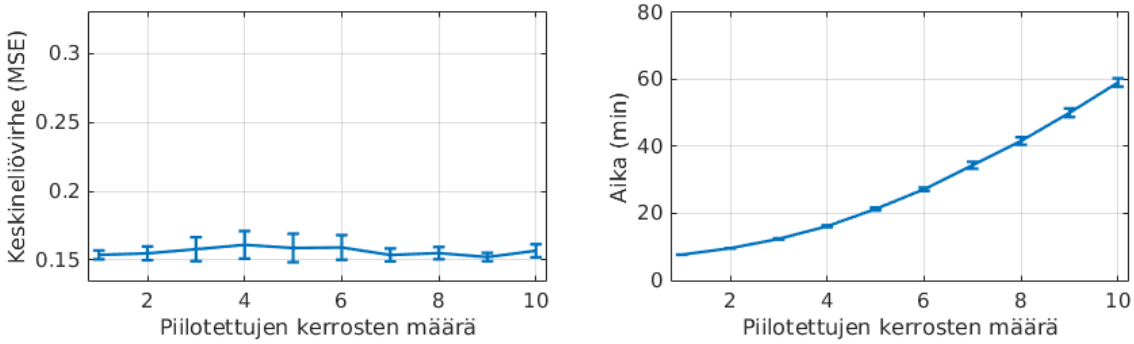
Levenbergin-Marquardtin algoritmin tulokset esitetään kahden luokan luokittelussa kuviossa 28 ja moniluokkaisessa luokittelussa kuviossa 30. Vastaavat bayesiläisen regularisoinnin tulokset taas on esitelty kuviossa 29 ja liitteen E kuviossa 44. Muiden opetusdatalla testattujen oppimismenetelmien tulokset kahdelle luokalle esitetään liitteessä E:

- BFGS- eli Broydenin-Fletcherin-Goldfarbin-Shannon menetelmä kuviossa 34
- Skaalattu liittogradienttimenetelmä eli SCG-algoritmi (*scaled conjugate gradient*) kuviossa 35
- Powellin ja Bealen liittogradienttimenetelmä kuviossa 36
- Fletcherin ja Reevesin liittogradienttimenetelmä kuviossa 37
- Polakin ja Ribièren liittogradienttimenetelmä kuviossa 38
- Vastavirta-algoritmi (*backpropagation*) eli gradienttimenetelmä (*gradient descent* tai *steepest descent*) kuviossa 39
- VLBP-algoritmi (*variable learning rate backpropagation* tai *gradient descent with adaptive learning rate*) kuviossa 40
- MOBP-algoritmi (*momentum backpropagation* tai *gradient descent with momentum backpropagation*) kuviossa 41
- VLBP- ja MOBP-algoritmien yhdistelmä kuviossa 42
- OSS-menetelmä (*one-step secant backpropagation*) kuviossa 43



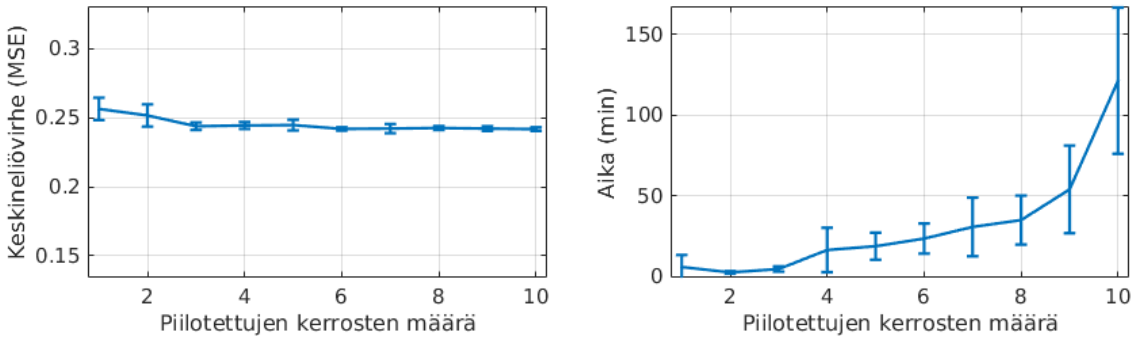
(a) Keskineliövirhe (MSE) (b) Opetusvaiheen ajankäyttö

Kuvio 28: Kaksiluokkainen Levenbergin-Marquardtin algoritmi.



(a) Keskineliövirhe (MSE) (b) Opetusvaiheen ajankäyttö

Kuvio 29: Kaksiluokkainen bayesiläinen regularisointi.



(a) Keskineliövirhe (MSE) (b) Opetusvaiheen ajankäyttö

Kuvio 30: Moniluokkainen Levenbergin-Marquardtin algoritmi.

Kuvioiden perusteella päädyttiin oppimismenetelmien valinnassa kokeilemaan kahta eri op-

pimismenetelmää, jotka ovat luvussa 8.1.1 esitelty Levenbergin-Marquardtin algoritmi sekä luvussa 8.1.2 esitelty bayesiläinen regularisointi. Nämä menetelmät tuottivat vertailussa tasaisesti alhaisimmat keskineliövirheet. Huonona puolena molemmissa menetelmissä oli opetusvaiheeseen käytetty aika. Varsinaiseen luvun 8.3 testausvaiheeseen päädyttiin

- kuvion 28 perusteella käyttämään kuutta piilotettua kerrosta Levenbergin-Marquardtin algoritmin kaksiluokkaisessa luokittelussa
- kuvion 29 perusteella yhtä piilotettua kerrosta bayesiläisellä regularisoinnilla kaksiluokkaisessa luokittelussa
- kuvion 30 perusteella kolmea piilotettua kerrosta Levenbergin-Marquardtin algoritmin moniluokkaisessa luokittelussa

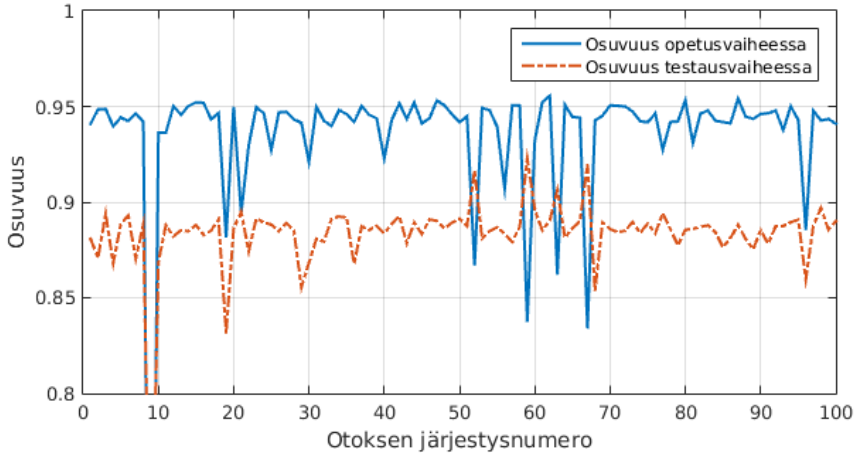
Liitteen E kuviossa 44 esitetty bayesiläinen regularisointi moniluokkaisena muistuttaa neliökeskivirheellä mitattuna moniluokkaista Levenbergin-Marquardtin algoritmia, mutta on aikavaatimuksiltaan huomattavasti vaativampi. Tämän vuoksi moniluokkainen bayesiläinen regularisointi päätettiin jättää testausvaiheen ulkopuolelle.

8.3 Menetelmien arviointi testausdatalla

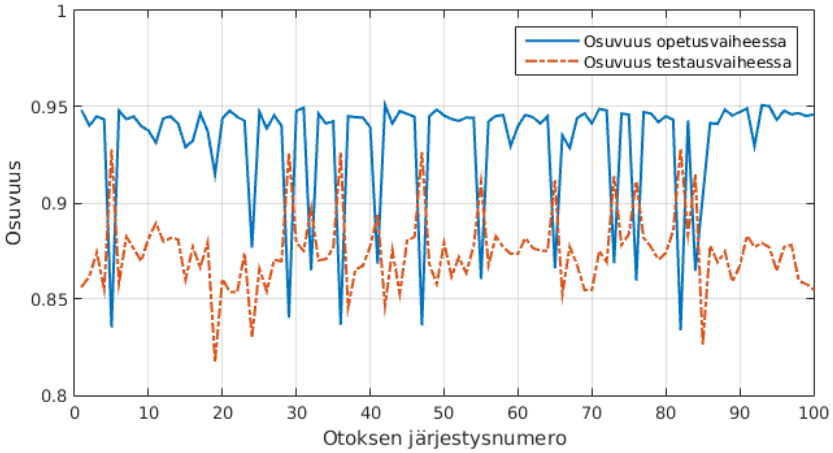
Kuten luvun 5.1 taulukosta 1 ilmeni, käytetyn datajoukon harjoitusdatassa on joidenkin hyökkäysryhmien instansseja huomattavan vähäinen määrä. Kun datajoukolle suoritetaan ositettu otanta, oli edellisessä luvussa käytetyllä otosmäärällä (6175 instanssia) esimerkiksi *worms*-ryhmän instansseja otannassa vain noin kolme kappaletta. Tämän vuoksi pohdittiin suuremman otoksen ottamista opetusdatasta. Datalla tehdyt karkeat testit viittasivat kuitenkin siihen, että datamäärän kaksinkertaistaminen vähintään kaksinkertaistaa oppimisvaiheeseen käytetyn ajan.

Opetukseen käytettävää datamäärää haluttiin joka tapauksessa kasvattaa hieman, joten ositettu otanta määriteltiin tässä vaiheessa poimimaan jokaisella oppimiskerralla opetusdatasta kymmenesosa, eli 8233 instanssia. Näytteiden määrän ja niiden esikäsittelyn vaikutus neuroverkon opetukseen on kuitenkin monimutkaisempi ongelma, jota ei tässä tutkielmassa juurikaan tarkastella. Neuroverkon testaus sen sijaan on hyvin nopeaa, ja siinä käytettiin datajoukon testausdataa kokonaisuudessaan.

Ajojen määrä kaikilla menetelmillä oli tässä vaiheessa sata kappaletta, ja mittaustuloksille laskettiin 99 %:n luottamusväli. Kuviossa 31 on vertailtu kahteen luokkaan luokittelevan Levenbergin-Marquardtin algoritmin suorituskykyä opetus- ja testausvaiheessa. Vastaava vertailu bayesiläisen regularisoinnin osalta esitetään kuviossa 32. Molemmista kuvioista voidaan erottaa piikkeinä kohdat, joissa testausvaiheen osuvuus oli parhaimmillaan. Samalla kuitenkin voidaan havaita, että nämä piikit ovat useimmiten kohdissa, joissa opetusvaiheen osuvuus on ollut heikoimmillaan. Tämä ilmiö tuli esille etenkin bayesiläisen regularisoinnin aikana ja osoittaa osaltaan, että opetusvaiheen hyvä suorituskyky ei takaa mallin toimivuutta tositilanteessa.



Kuvio 31: Opetus- ja testausvaiheen osuvuuksien vertailu Levenbergin-Marquardtin algoritmilla.



Kuvio 32: Opetus- ja testausvaiheen osuvuuksien vertailu bayesiläisellä regularisoinnilla.

Taulukossa 13 esitetään molempien käytetyn kaksiluokkaisen oppimismenetelmän luokittelujen osuvuus ja taulukossa 14 luokittelujen osuvuus moniluokkaisella Levenbergin-Marquardtin algoritmilla. Taulukosta 14 ilmenee, että kun Levenbergin-Marquardtin algoritmia opetettiin kymmeneen ryhmään jaetulla datalla, suoriutui se kahden pääluokan luokittelussakin huomattavasti kahdella luokalla opetettua samaa algoritmia (ks. taulukko 13) heikommin. Tällä tavalla opetettu neuroverkko ei luokitellut ryhmiin *analysis*, *backdoor*, *shellcode* ja *worms* yhtäkään instanssia. Nämä ovat samalla juuri ne ryhmät, joiden instansseja on opetusdatassa kutakin alle prosentin verran.

Taulukko 13: Luokittelun osuvuus kahdella pääluokalla.

Luokka	**Ryhmä**	**Levenberg-Marquardt, osuvuuden keskiarvo luottamusvälillä 99 %**	**Bayesiläinen regularisointi, osuvuuden keskiarvo luottamusvälillä 99 %**
Normaali	**Normal**	$0,9563 \pm 0,0261$	$0,9550 \pm 0,0127$
Haitta-liikenne	**Analysis**	$0,7494 \pm 0,0190$	$0,7208 \pm 0,0131$
	Backdoor	$0,9381 \pm 0,0075$	$0,8993 \pm 0,0129$
	DoS	$0,9398 \pm 0,0074$	$0,9035 \pm 0,0128$
	Exploits	$0,9297 \pm 0,0082$	$0,9065 \pm 0,0081$
	Fuzzers	$0,3206 \pm 0,0329$	$0,3186 \pm 0,0534$
	Generic	$0,9975 \pm 0,0004$	$0,9970 \pm 0,0003$
	Reconnaissance	$0,8432 \pm 0,0100$	$0,8403 \pm 0,0149$
	Shellcode	$0,7901 \pm 0,0135$	$0,8178 \pm 0,0180$
	Worms	$0,9125 \pm 0,0230$	$0,8753 \pm 0,0141$

Testausdatajoukon instanssien todelliset ja ennustetut ryhmät esitetään taulukossa 15. Taulukko vastaa *k*-means-luvun taulukkoa 5 ja tukivektorikone-luvun taulukkoa 11, eli taulukossa on diagonaalisesti tummennettuna esitetty oikein luokiteltujen instanssien määrät, ja kaikki kappalemäärät ovat sadan ajon keskiarvoja.

Taulukossa 16 esitetään kaksiluokkaisten oppimismenetelmien suorituskyky yhdellätoista mittarilla. Bayesiläisen regularisoinnin opetus keskeytyi kaikilla kerroilla iteraatioiden maksimimääräksi asetettuun tuhanteen iteraatioon. On siis mahdollista, että sallimalla suurem-

Taulukko 14: Moniluokkainen Levenbergin-Marquardtin algoritmi.

Luokka	**Ryhmä**	**Osuvuus kahdella pääluokalla luottamusvälillä 99 %**	**Osuvuus kymmenellä ryhmällä luottamusvälillä 99 %**
Normaali	**Normal**	$0,9820 \pm 0,0064$	$0,9820 \pm 0,0064$
Haitta-liikenne	**Analysis**	$0,6654 \pm 0,0292$	0
	Backdoor	$0,8231 \pm 0,0359$	0
	DoS	$0,8327 \pm 0,0355$	$0,2591 \pm 0,0555$
	Exploits	$0,8096 \pm 0,0420$	$0,6115 \pm 0,0488$
	Fuzzers	$0,1709 \pm 0,0118$	$0,0379 \pm 0,0080$
	Generic	$0,9943 \pm 0,0009$	$0,9787 \pm 0,0003$
	Reconnaissance	$0,5721 \pm 0,0520$	$0,2036 \pm 0,0463$
	Shellcode	$0,4341 \pm 0,0658$	0
	Worms	$0,7299 \pm 0,0600$	0

man määrän iteraatioita myös menetelmän suorituskyky olisi ollut hieman parempi. Nyt menetelmä pärjäsi lähes kaikilla mittareilla hieman Levenbergin-Marquardtin algoritmia heikommin, vaikka olikin ajallisesti nopeampi.

Taulukossa 17 esitetään moniluokkaisen Levenbergin-Marquardtin suorituskyky vastaavilla mittareilla. *FPR*, *TNR*, osuvuus kahdella luokalla, *recall*, *precision* ja *f-measure* on laskettu kahteen luokkaan luokittelun perusteella. Opetusvaiheen osuvuus kymmenellä ryhmällä oli 0,8007, ja varsinaisessa testausvaiheessa tämä osuvuus laski 0,6876:een.

Taulukko 15: Moniluokkainen Levenbergin-Marquardtin algoritmi: Instanssien ennustetut ja todelliset ryhmät.

		Ennustettu									
		Normal	***Analysis***	***Backdoor***	***DoS***	***Exploits***	***Fuzzers***	***Generic***	***Reconnaissance***	***Shellcode***	***Worms***
Todellinen ryhmä	**Normal**	**54991**	0	0	17	480	187	243	82	0	0
	Analysis	669	**0**	0	485	719	58	54	14	0	0
	Backdoor	309	0	**0**	469	797	79	59	33	0	0
	DoS	2052	0	0	**3178**	5861	469	505	199	0	0
	Exploits	6359	0	0	4357	**20421**	1012	721	523	0	0
	Fuzzers	15076	0	0	499	1206	**688**	448	265	0	0
	Generic	229	0	0	111	445	42	**39150**	24	0	0
	Reconn-aissance	4489	0	0	589	2488	605	183	**2136**	0	0
	Shellcode	641	0	0	0	93	89	7	303	**0**	0
	Worms	35	0	0	2	73	5	4	10	0	**0**

Taulukko 16: Kaksiluokkaisten oppimismenetelmien arviointi testausdatalla.

	Levenberg-Marquardt, keskiarvo luottamusvälillä 99 %	**Bayesiläinen regularisointi, keskiarvo luottamusvälillä 99 %**
FPR	$0,2576 \pm 0,0211$	$0,2599 \pm 0,0165$
TNR	$0,7424 \pm 0,0211$	$0,7401 \pm 0,0165$
Osuvuus	$0,8832 \pm 0,0061$	$0,8747 \pm 0,0051$
Recall	$0,9790 \pm 0,0085$	$0,9770 \pm 0,0057$
Precision	$0,8488 \pm 0,0076$	$0,8370 \pm 0,0121$
F-measure	$0,9083 \pm 0,0037$	$0,9004 \pm 0,0046$
Opetusaika	13 min 53 s $\pm$ 3 min 10 s	9 min 58 s $\pm$ 1 s
Testausaika (s)	$1,15 \pm 0,08$	$1,13 \pm 0,02$
Iteraatioiden lkm	427 ± 97	1000 ± 0
Keskineliövirhe opetuksen aikana	$0,1587 \pm 0,0048$	$0,1591 \pm 0,0040$
Osuvuus opetuksen aikana	$0,9346 \pm 0,0115$	$0,9314 \pm 0,0080$

Taulukko 17: Moniluokkaisen Levenbergin-Marquardtin algoritmin arviointi testausdatalla.

	Keskiarvo luottamusvälillä 99 %
FPR	$0,3452 \pm 0,0162$
TNR	$0,6548 \pm 0,0162$
Osuvuus kahdella luokalla	$0,8240 \pm 0,0145$
Osuvuus kymmenellä ryhmällä	$0,6876 \pm 0,0105$
Recall	$0,9888 \pm 0,0042$
Precision	$0,7498 \pm 0,0208$
F-measure	$0,8502 \pm 0,0156$
Opetusaika	13 min 0 s $\pm$ 5 s
Testausaika (s)	$1,04 \pm 0,14$
Iteraatioiden lukumäärä	155 ± 61
Keskineliövirhe opetuksen aikana	$0,2441 \pm 0,0012$
Osuvuus kymmenellä ryhmällä opetuksen aikana	$0,8007 \pm 0,0117$

9 Yhteenveto

Tutkielmassa vertailluista menetelmistä parhaiten UNSW-NB15-datajoukon verkkoliikenteen luokittelussa onnistui *k*-means-klusterointi JMI-algoritmilla suoritetulla ominaisuuksien valinnalla. Sillä saavutettiin alhaisin väärien positiivisten osuus (*FPR*) 0,1366, korkein oikeiden negatiivisten osuus (*TNR*) 0,8634 ja paras osuvuus kahdella luokalla 0,9036. Menetelmä oli myös selkeästi muita nopeampi. Paras osuvuus kaikilla kymmenellä ryhmällä mitattuna, 0,7377, sen sijaan saavutettiin moniluokkaisella tukivektorikoneella. Tukivektorikonemenetelmien eräänä huonona puolena oli niiden testaukseen käyttämä aika.

Datajoukon ryhmistä helpoimmin haittaliikenteeksi tunnistettiin *generic*-ryhmään kuuluva liikenne, jonka kaikki menetelmät luokittelivat oikein yli 99 %:n todennäköisyydellä. Haastavimmaksi osoittautui *fuzzers*-ryhmän liikenne, jonka tunnistuksen osuvuus vaihteli moniluokkaisen Levenbergin-Marquardtin algoritmin tuottaman 17 %:n tunnistusprosentin ja JMI-*k*-means-yhdistelmän tuottaman 75 prosentin välillä.

Vertailu menetelmien välillä oli tässä tutkielmassa osin epäreilu, sillä esimerkiksi datan vähentämisen menetelmiä hyödynnettiin ainoastaan *k*-means-klusteroinnin yhteydessä. Kaikista menetelmistä kuitenkin löytyy lähes loputtomasti säätövaraa, joiden kautta niiden suorituskykyä voidaan parantaa. UNSW-NB15-datajoukkoa käyttäen on jo julkaistu joitakin tutkimuksia, ja esimerkiksi Chowdhury, Ferens ja Ferens (2016) saavuttivat tukivektorikonemenetelmällä huomattavasti tässä tutkielmassa saavutettua tulosta paremman tuloksen kiinnittämällä huomiota datajoukosta valittuihin ominaisuuksiin.

Viime vuosina luvuissa 6, 7 ja 8 esiteltyjä menetelmiä on sovellettu alan tutkimuksessa usein yhdistelemällä niitä keskenään. Esimerkiksi *k*-means-klusterointia on käytetty naiivin Bayes-luokittelijan (Sharma ym. 2012) tai *k*:n lähimmän naapurin menetelmän (Lin, Ke ja Tsai 2015) kanssa. Tukivektorikonetta on käytetty muun muassa *k*-meansin (Hatim ym. 2015) sekä ydinpääkomponenttianalyysin (*kernel PCA*) ja geneettisten algoritmien (Kuang, Xu ja Zhang 2014) yhteydessä. Chandrashekhar ja Raghuveer (2014) taas ovat yhdistelleet neuroverkkomenetelmiä *k*-meansiin ja tukivektorikoneeseen.

Kuten Axelsson (2000) osoitti, rajoittaa väärien hälytysten määrä vakavasti käytännön so-

vellutuksia tällä tutkimusalueella. Todennäköistä onkin, että kaikissa tilanteissa luotettavasti toimivaa järjestelmää ei näillä menetelmillä ole mahdollista saada aikaan. Ongelma olisikin kenties jollain tavalla pilkottava paloiksi, ja määritellä tarkasti mitä verkkodatasta oikeasti halutaan löytää, kuten Sommer ja Paxson (2010) ovat ehdottaneet. Tällainen erikoistunut järjestelmä keskittyisi yhteen ongelmaan kerrallaan, eikä pyrkisi yhdellä iskulla ratkaisemaan kaikkia ongelmia. Myös perinteisillä väärinkäytöspohjaisilla menetelmillä olisi paikkansa tällaisessa mallissa. Toinen mahdollisuus voisi olla se, että hälytysten seulominen automaattisesti kehittyy huomattavasti, jolloin ihmisanalyytikon tulisi kiinnittää huomiota vain valikoituihin aidosti kiinnostaviin tapauksiin. Sekä reaaliaikaiselle että jälkikäteen tapahtuvalle verkkoliikenteen seurannalle näyttää myös olevan omat paikkansa tietoturvan ylläpitämisessä, kuten Brugger (2004) on todennut.

Lähteet

Abt, Sebastian, ja Harald Baier. 2014a. “A plea for utilising synthetic data when performing machine learning based cyber-security experiments”. Teoksessa *Proceedings of the 2014 workshop on artificial intelligent and security workshop,* 37–45. AISec ’14. Scottsdale, Arizona, USA: ACM. ISBN: 978-1-4503-3153-1. doi:10.1145/2666652.2666663.

———. 2014b. “Are we missing labels? A study of the availability of ground-truth in network security research”. Teoksessa *2014 third international workshop on building analysis datasets and gathering experience returns for security (BADGERS),* 40–55. Syyskuu. doi:10.1109/BADGERS.2014.11.

Ahmed, Mohiuddin, Abdun Naser Mahmood ja Jiankun Hu. 2016. “A survey of network anomaly detection techniques”. *Journal of network and computer applications* 60:19–31. ISSN: 1084-8045. doi:10.1016/j.jnca.2015.11.016.

Alpaydın, Ethem. 2004. *Introduction to machine learning.* MIT Press. ISBN: 0-262-01211-1.

Ampazis, N., ja S. J. Perantonis. 2000. “Levenberg-Marquardt algorithm with adaptive momentum for the efficient training of feedforward networks”. Teoksessa *Neural networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS international joint conference on,* 1:126–131. doi:10.1109/IJCNN.2000.857825.

Anderson, James P. 1980. *Computer security threat monitoring and surveillance.* Tekninen raportti. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania.

Androulidakis, G., ja S. Papavassiliou. 2008. “Improving network anomaly detection via selective flow-based sampling”. *Communications, IET* 2, numero 3 (maaliskuu): 399–409. ISSN: 1751-8628. doi:10.1049/iet-com:20070231.

Arlot, Sylvain, ja Alain Celisse. 2010. “A survey of cross-validation procedures for model selection”. *Statistics surveys* 4:40–79. doi:10.1214/09-SS054.

Arthur, David, ja Sergei Vassilvitskii. 2007. “K-means++: The advantages of careful seeding”. Teoksessa *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms,* 1027–1035. SODA ’07. New Orleans, Louisiana: Society for Industrial / Applied Mathematics. ISBN: 978-0-898716-24-5, viitattu 23. kesäkuuta 2016. http://dl.acm.org/citation.cfm?id=1283383.1283494.

Axelsson, Stefan. 1999. *Research in intrusion detection systems: A survey.* Tekninen raportti. Department of computer engineering, Chalmers university of technology, Göteborg, Sweden.

———. 2000. “The base-rate fallacy and the difficulty of intrusion detection”. *ACM Trans. Inf. Syst. Secur.* (New York, NY, USA) 3, numero 3 (elokuu): 186–205. ISSN: 1094-9224. doi:10.1145/357830.357849.

Bartos, Karel, ja Martin Rehak. 2012. “Traffic monitoring and analysis: 4th international workshop, TMA 2012, Vienna, Austria, March 12, 2012. Proceedings”. Luku Towards efficient flow sampling technique for anomaly detection, toimittanut Antonio Pescapè, Luca Salgarelli ja Xenofontas Dimitropoulos, 93–106. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-28534-9. doi:10.1007/978-3-642-28534-9_11.

Bhuyan, Monowar H, Dhruba K Bhattacharyya ja Jugal K Kalita. 2011. “Surveying port scans and their detection methodologies”. *The Computer Journal* 54 (10): 1565–1581. doi:10.1093/comjnl/bxr035.

———. 2014. “Network anomaly detection: Methods, systems and tools”. *Communications surveys tutorials, IEEE* 16 (1): 303–336. ISSN: 1553-877X. doi:10.1109/SURV.2013.052213.00046.

Bilge, Leyla, ja Tudor Dumitras. 2012. “Before we knew it: An empirical study of zero-day attacks in the real world”. Teoksessa *Proceedings of the 2012 ACM conference on computer and communications security,* 833–844. CCS ’12. Raleigh, North Carolina, USA: ACM. ISBN: 978-1-4503-1651-4. doi:10.1145/2382196.2382284.

Blum, Avrim L., ja Pat Langley. 1997. “Selection of relevant features and examples in machine learning”. *Artificial intelligence* 97 (1–2): 245–271. ISSN: 0004-3702. doi:10.1016/S0004-3702(97)00063-5.

Bolón-Canedo, V., N. Sánchez-Maroño ja A. Alonso-Betanzos. 2009. “A combination of discretization and filter methods for improving classification performance in KDD Cup 99 dataset”. Teoksessa *Neural networks, 2009. IJCNN 2009. International joint conference on,* 359–366. Kesäkuu. doi:10.1109/IJCNN.2009.5178622.

Boriah, Shyam, Varun Chandola ja Vipin Kumar. 2008. “Similarity measures for categorical data: A comparative evaluation”. Luku 21 teoksessa *Proceedings of the 2008 SIAM international conference on data mining,* 243–254. doi:10.1137/1.9781611972788.22.

Bou-Harb, E., M. Debbabi ja C. Assi. 2014. “Cyber scanning: A comprehensive survey”. *IEEE communications surveys tutorials* 16 (3): 1496–1519. ISSN: 1553-877X. doi:10.1109/SURV.2013.102913.00020.

Bradley, Andrew P. 1997. “The use of the area under the ROC curve in the evaluation of machine learning algorithms”. *Pattern recognition* 30 (7): 1145–1159. ISSN: 0031-3203. doi:10.1016/S0031-3203(96)00142-2.

Brown, Gavin, Adam Pocock, Ming-Jie Zhao ja Mikel Luján. 2012. “Conditional likelihood maximisation: A unifying framework for information theoretic feature selection”. *The journal of machine learning research* 13 (tammikuu): 27–66. ISSN: 1532-4435, viitattu 1. elokuuta 2016. http://dl.acm.org/citation.cfm?id=2188385.2188387.

Brugger, Terry S. 2004. “Data mining methods for network intrusion detection”. Tohtorinväitöskirja, University of California, Davis. Viitattu 29. maaliskuuta 2016. http://ftp.bstu.by/ai/To-dom/My_research/Papers-0/For-research/D-mining/Anomaly-D/Intrusion-detection/brugger-dmnid.pdf.

Catania, Carlos A., ja Carlos García Garino. 2012. “Automatic network intrusion detection: Current techniques and open issues”. Special issue on recent advances in security and privacy in distributed communications and image processing, *Computers & Electrical Engineering* 38 (5): 1062–1072. ISSN: 0045-7906. doi:10.1016/j.compeleceng.2012.05.013.

Cha, Sung-Hyuk. 2007. “Comprehensive survey on distance/similarity measures between probability density functions”. *International journal of mathematical models and methods in applied sciences* 1 (4): 300–307. ISSN: 1998-0140, viitattu 10. maaliskuuta 2016. http://www.naun.org/main/NAUN/ijmmas/mmmas-49.pdf.

Chandola, Varun, Arindam Banerjee ja Vipin Kumar. 2009. “Anomaly detection: A survey”. *ACM comput. surv.* (New York, NY, USA) 41, numero 3 (heinäkuu): 15:1–15:58. ISSN: 0360-0300. doi:10.1145/1541880.1541882.

Chandrashekhar, A. M., ja K. Raghuveer. 2014. “Amalgamation of k-means clustering algorithm with standard MLP and SVM based neural networks to implement network intrusion detection system”. Teoksessa *Advanced computing, networking and informatics - Volume 2: Wireless networks and security proceedings of the second international conference on advanced computing, networking and informatics (ICACNI-2014),* toimittanut Malay Kumar Kundu, Durga Prasad Mohapatra, Amit Konar ja Aruna Chakraborty, 273–283. Springer International Publishing. ISBN: 978-3-319-07350-7. doi:10.1007/978-3-319-07350-7_31.

Chang, Chih-Chung, ja Chih-Jen Lin. 2011. “LIBSVM: A library for support vector machines”. Ohjelma ladattavissa osoitteessa http://www.csie.ntu.edu.tw/~cjlin/libsvm, *ACM transactions on intelligent systems and technology* 2 (3): 27:1–27:27. Viitattu 23. syyskuuta 2016. doi:10.1145/1961189.1961199.

Chen, Thomas M., ja Patrick J. Walsh. 2013. “Guarding against network intrusions”. Teoksessa Vacca 2013, 81–95.

Chen, You, Yang Li, Xue-Qi Cheng ja Li Guo. 2006. “Survey and taxonomy of feature selection algorithms in intrusion detection system”. Teoksessa *Information security and cryptology,* toimittanut Helger Lipmaa, Moti Yung ja Dongdai Lin, 4318:153–167. Lecture notes in computer science. Springer Berlin Heidelberg. ISBN: 978-3-540-49608-3. doi:10.1007/11937807_13.

Chmielewski, Michal R., ja Jerzy W. Grzymala-Busse. 1996. “Global discretization of continuous attributes as preprocessing for machine learning”. Rough sets, *International journal of approximate reasoning* 15 (4): 319–331. ISSN: 0888-613X. doi:10.1016/S0888-613X(96)00074-6.

Chowdhury, Md Nasimuzzaman, Ken Ferens ja Mike Ferens. 2016. “Network intrusion detection using machine learning”. Teoksessa *Proceedings of the international conference on security and management (SAM),* 30–35. Athens: The steering committee of the world congress in computer science, computer engineering / applied computing (WorldComp). Viitattu 20. marraskuuta 2016. http://search.proquest.com/docview/1807002945?accountid=11774.

Danchev, Dancho. 2008. “Black market for zero day vulnerabilities still thriving”. Viitattu 21. tammikuuta 2016. http://www.zdnet.com/article/black-market-for-zero-day-vulnerabilities-still-thriving.

“DARPA intrusion detection data sets”. 2014. Viitattu 20. lokakuuta 2014. http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/.

Davis, Jonathan J., ja Andrew J. Clark. 2011. “Data preprocessing for anomaly based network intrusion detection: A review”. *Computers & security* 30 (6–7): 353–375. ISSN: 0167-4048. doi:10.1016/j.cose.2011.05.008.

Day, Christopher. 2013. “Intrusion prevention and detection systems”. Teoksessa Vacca 2013, 485–498.

Debar, Hervé, Marc Dacier ja Andreas Wespi. 1999. “Towards a taxonomy of intrusion-detection systems”. *Computer networks* 31 (8): 805–822. ISSN: 1389-1286. doi:10.1016/S1389-1286(98)00017-6.

“Definition of anomaly by Merriam-Webster”. 2015. Viitattu 15. joulukuuta 2015. http://www.merriam-webster.com/dictionary/anomaly.

Denning, D.E. 1987. “An intrusion-detection model”. *Software engineering, IEEE transactions on* SE-13, numero 2 (helmikuu): 222–232. ISSN: 0098-5589. doi:10.1109/TSE.1987.232894.

Deza, Michel Marie, ja Elena Deza. 2009. *Encyclopedia of distances.* Springer Berlin Heidelberg. ISBN: 978-3-642-00234-2. doi:10.1007/978-3-642-00234-2_1.

Fawcett, Tom. 2006. “An introduction to ROC analysis”. ROC analysis in pattern recognition, *Pattern recognition letters* 27 (8): 861–874. ISSN: 0167-8655. doi:10.1016/j.patrec.2005.10.010.

“FEAST: A feature selection toolbox for C and Matlab”. 2011. Viitattu 1. elokuuta 2016. http://www.cs.man.ac.uk/~gbrown/fstoolbox/.

Fleuret, François. 2004. “Fast binary feature selection with conditional mutual information”. *Journal of machine learning research* 5 (Nov): 1531–1555. Viitattu 9. elokuuta 2016. http://www.jmlr.org/papers/v5/fleuret04a.html.

Foresee, F. Dan, ja M. T. Hagan. 1997. “Gauss-Newton approximation to Bayesian learning”. Teoksessa *Neural networks,1997., International conference on,* nide 3, 1930–1935 vol.3. Kesäkuu. doi:10.1109/ICNN.1997.614194.

Foster, James C., ja Mike Price. 2005. *Sockets, shellcode, porting & coding: Reverse engineering exploits and tool coding for security professionals.* Syngress. ISBN: 9781597490054.

Friedman, Jerome, Trevor Hastie ja Robert Tibshirani. 2009. *The elements of statistical learning: Data mining, inference, and prediction - Second edition.* Springer series in statistics. Springer, Berlin. ISBN: 978-0-387-84858-7. doi:10.1007/978-0-387-84858-7.

Gan, Guojun, Chaoqun Ma ja Jianhong Wu. 2007. *Data clustering: Theory, algorithms, and applications.* Nide 20. Siam. ISBN: 978-0-898716-23-8.

García, S., J. Luengo, J. A. Sáez, V. López ja F. Herrera. 2013. “A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning”. *IEEE transactions on knowledge and data engineering* 25, numero 4 (huhtikuu): 734–750. ISSN: 1041-4347. doi:10.1109/TKDE.2012.35.

Gates, Carrie, ja Carol Taylor. 2007. “Challenging the anomaly detection paradigm: A provocative discussion”. Teoksessa *Proceedings of the 2006 workshop on new security paradigms,* 21–29. NSPW ’06. Germany: ACM. ISBN: 978-1-59593-923-4, viitattu 18. marraskuuta 2015. doi:10.1145/1278940.1278945. http://dl.acm.org/citation.cfm?id=1278945.

Gonzalez, Hugo, Marc Antoine Gosselin-Lavigne, Natalia Stakhanova ja Ali A. Ghorbani. 2015. “The impact of application-layer denial-of-service attacks”. Teoksessa Israr ja Issac 2015, 261–272.

Hagan, M. T., ja M. B. Menhaj. 1994. “Training feedforward networks with the Marquardt algorithm”. *IEEE transactions on neural networks* 5, numero 6 (marraskuu): 989–993. ISSN: 1045-9227. doi:10.1109/72.329697.

Halminen, Laura, Marko Junkkari, Esa Juntunen, Toni Lehtinen, Leo Pugin ja Anna Vanninen. 2013. “Ulkoministeriön verkko oli täysin ulkopuolisten hallussa”. Viitattu 12. huhtikuuta 2016. http://www.hs.fi/kotimaa/a1383199949222.

Han, Jiawei, ja Micheline Kamber. 2006. *Data mining: Concepts and techniques.* Morgan Kaufmann. ISBN: 978-1-55860-901-3.

Hatim, Mohamad Tahir, Wail Hasan, Abas Md Said, Nur Haryani Zakaria, Norliza Katuk, Nur Farzana Kabir, Mohd Hasbullah Omar, Osman Ghazali ja Noor Izzah Yahya. 2015. “Hybrid machine learning technique for intrusion detection system”. 5th international conference on computing ja informatics (ICOCI). ISBN: 978-967-0910-02-4, viitattu 16. marraskuuta 2016. http://www.lib.uum.edu.my:8080/jspui/handle/123456789/161.

Haykin, Simon. 2009. *Neural networks and learning machines: Third edition.* Pearson Education. ISBN: 978-0-13-129376-2.

Hoque, N., Monowar H. Bhuyan, R.C. Baishya, D.K. Bhattacharyya ja J.K. Kalita. 2014. “Network attacks: Taxonomy, tools and systems”. *Journal of network and computer applications* 40:307–324. ISSN: 1084-8045. doi:10.1016/j.jnca.2013.08.001.

Hsu, Chih-Wei, Chih-Chung Chang ja Chih-Jen Lin. 2016. “A practical guide to SVM classification”. Viitattu 29. syyskuuta 2016. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Hsu, Chih-Wei, ja Chih-Jen Lin. 2002. “A comparison of methods for multiclass support vector machines”. *IEEE transactions on neural networks* 13, numero 2 (maaliskuu): 415–425. ISSN: 1045-9227. doi:10.1109/72.991427.

Huang, Jin, ja C. X. Ling. 2005. “Using AUC and accuracy in evaluating learning algorithms”. *IEEE transactions on knowledge and data engineering* 17, numero 3 (maaliskuu): 299–310. ISSN: 1041-4347. doi:10.1109/TKDE.2005.50.

Hussain, Alefiya, John Heidemann ja Christos Papadopoulos. 2003. “A framework for classifying denial of service attacks”. Teoksessa *Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications,* 99–110. SIGCOMM ’03. Karlsruhe, Germany: ACM. ISBN: 1-58113-735-4. doi:10.1145/863955.863968.

Israr, Nauman, ja Biju Issac. 2015. *Case studies in secure computing: Achievements and trends.* Auerbach Publications. ISBN: 978-1-4822-0707-1.

Al-Jarrah, O., ja A. Arafat. 2014. “Network intrusion detection system using attack behavior classification”. Teoksessa *Information and communication systems (ICICS), 2014 5th international conference on,* 1–6. Huhtikuu. doi:10.1109/IACS.2014.6841978.

John, George H, Ron Kohavi ja Karl Pfleger. 1994. “Irrelevant features and the subset selection problem”. Teoksessa *Machine learning: Proceedings of the eleventh international conference,* 121–129. Viitattu 15. helmikuuta 2016. http://robotics.stanford.edu/~ronnyk/ml94.pdf.

Kantardzic, Mehmed. 2011. *Data mining: Concepts, models, methods, and algorithms.* John Wiley & Sons. ISBN: 978-1-118-02912-1.

Kayacık, H Güneş, A Nur Zincir-Heywood ja Malcolm I Heywood. 2004. “On dataset biases in a learning system with minimum a priori information for intrusion detection”. Teoksessa *Communication networks and services research, 2004. Proceedings. Second annual conference on,* 181–189. Toukokuu. doi:10.1109/DNSR.2004.1344727.

“KDD cup 1999 data”. 1999. Viitattu 20. lokakuuta 2014. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Kendall, Kristopher. 1999. “A database of computer attacks for the evaluation of intrusion detection systems”. Tutkielma, Massachusetts Institute of Technology.

Kohavi, Ron. 1995. “A Study of cross-validation and bootstrap for accuracy estimation and model selection”. Teoksessa *Proceedings of the 14th international joint conference on artificial intelligence - Volume 2,* 1137–1143. IJCAI’95. Montreal, Quebec, Kanada: Morgan Kaufmann Publishers Inc. ISBN: 1-55860-363-8, viitattu 28. kesäkuuta 2016. http://robotics.stanford.edu/~ronnyk/accEst.pdf.

Kotzanikolaou, Panayiotis, ja Christos Douligeris. 2007. “Computer network security: Basic background and current issues”. Teoksessa *Network Security: Current Status and Future Directions,* toimittanut Christos Douligeris ja Dimitrios N. Serpanos, 1–12. John Wiley & Sons, Inc. ISBN: 978-0-471-70355-6. doi:10.1002/9780470099742.ch1.

Kuang, Fangjun, Weihong Xu ja Siyang Zhang. 2014. “A novel hybrid KPCA and SVM with GA model for intrusion detection”. *Applied soft computing* 18:178–184. ISSN: 1568-4946. doi:10.1016/j.asoc.2014.01.028.

Lazarevic, Aleksandar, Vipin Kumar ja Jaideep Srivastava. 2005. “Intrusion detection: A survey”. Teoksessa *Managing cyber threats,* toimittanut Vipin Kumar, Jaideep Srivastava ja Aleksandar Lazarevic, 5:19–78. Massive computing. Springer US. ISBN: 978-0-387-24226-2. doi:10.1007/0-387-24230-9_2.

Leppänen, Mikko. 2013. “MTV3: Suomen ulkoministeriö laajan verkkovakoilun kohteena vuosia”. Viitattu 12. huhtikuuta 2016. http://yle.fi/uutiset/mtv3_suomen_ulkoministerio_laajan_verkkovakoilun_kohteena_vuosia/6911225.

Liao, Hung-Jen, Chun-Hung Richard Lin, Ying-Chih Lin ja Kuang-Yuan Tung. 2013. “Intrusion detection system: A comprehensive review”. *Journal of network and computer applications* 36 (1): 16–24. ISSN: 1084-8045. doi:10.1016/j.jnca.2012.09.004.

Lin, Dahua, ja Xiaoou Tang. 2006. “Conditional infomax learning: An integrated framework for feature extraction and fusion”. Teoksessa *Computer vision – ECCV 2006: 9th European conference on computer vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I,* toimittanut Aleš Leonardis, Horst Bischof ja Axel Pinz, 68–82. Springer Berlin Heidelberg. ISBN: 978-3-540-33833-8. doi:10.1007/11744023_6.

Lin, J., E. Keogh, A. Fu ja H. Van Herle. 2005. “Approximations to magic: finding unusual medical time series”. Teoksessa *Computer-based medical systems, 2005. Proceedings. 18th IEEE symposium on,* 329–334. Kesäkuu. doi:10.1109/CBMS.2005.34.

Lin, Wei-Chao, Shih-Wen Ke ja Chih-Fong Tsai. 2015. “CANN: An intrusion detection system based on combining cluster centers and nearest neighbors”. *Knowledge-based systems* 78:13–21. ISSN: 0950-7051. doi:10.1016/j.knosys.2015.01.009.

Liu, Huan, Farhad Hussain, Chew Lim Tan ja Manoranjan Dash. 2002. “Discretization: An enabling technique”. *Data mining and knowledge discovery* 6 (4): 393–423. ISSN: 1573-756X. doi:10.1023/A:1016304305535.

MacKay, David J. C. 1992a. “A practical Bayesian framework for backpropagation networks”. *Neural computation* 4 (3): 448–472. ISSN: 0899-7667. doi:10.1162/neco.1992.4.3.448.

———. 1992b. “Bayesian interpolation”. *Neural computation* 4 (3): 415–447. ISSN: 0899-7667. doi:10.1162/neco.1992.4.3.415.

———. 2005. *Information theory, inference and learning algorithms.* Cambridge university press. ISBN: 0-521-64298-1.

Mahoney, Matthew V., ja Philip K. Chan. 2003. “An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection”. Teoksessa *Recent advances in intrusion detection,* toimittanut Giovanni Vigna, Christopher Kruegel ja Erland Jonsson, 2820:220–237. Lecture notes in computer science. Springer Berlin Heidelberg. ISBN: 978-3-540-40878-9. doi:10.1007/978-3-540-45248-5_13.

Mai, Jianning, Chen-Nee Chuah, Ashwin Sridharan, Tao Ye ja Hui Zang. 2006. “Is sampled data sufficient for anomaly detection?” Teoksessa *Proceedings of the 6th ACM SIGCOMM conference on internet measurement,* 165–176. IMC ’06. Rio de Janeiro, Brazil: ACM. ISBN: 1-59593-561-4. doi:10.1145/1177080.1177102.

“MathWorks documentation: Bayesian regularization backpropagation”. 2016. Viitattu 7. marraskuuta 2016. https://se.mathworks.com/help/nnet/ref/trainbr.html.

"MathWorks documentation: Choose neural network input-output processing functions". 2016. Viitattu 8. marraskuuta 2016. https://se.mathworks.com/help/nnet/ug/choose-neural-network-input-output-processing-functions.html.

"MathWorks documentation: Perfcurve". 2016. Viitattu 13. syyskuuta. http://se.mathworks.com/help/stats/perfcurve.html#outputarg_OPTROCPT.

Mazel, J., R. Fontugne ja K. Fukuda. 2014. "A taxonomy of anomalies in backbone network traffic". Teoksessa *Wireless communications and mobile computing conference (IWCMC), 2014 international,* 30–36. Elokuu. doi:10.1109/IWCMC.2014.6906328.

Metz, Charles E. 1978. "Basic principles of ROC analysis". *Seminars in nuclear medicine* 8 (4): 283–298. ISSN: 0001-2998. doi:10.1016/S0001-2998(78)80014-2.

Mirkovic, Jelena, ja Peter Reiher. 2004. "A taxonomy of DDoS attack and DDoS defense mechanisms". *SIGCOMM comput. commun. rev.* (New York, NY, USA) 34, numero 2 (huhtikuu): 39–53. ISSN: 0146-4833. doi:10.1145/997150.997156.

Moustafa, Nour, ja Jill Slay. 2015. "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". Teoksessa *Military communications and information systems conference (MilCIS), 2015,* 1–6. Marraskuu. doi:10.1109/MilCIS.2015.7348942.

———. 2016a. "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set". *Information security journal: A global perspective:*1–14. doi:10.1080/19393555.2015.1125974.

———. 2016b. "The UNSW-NB15 data set description". Viitattu 18. toukokuuta 2016. https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/.

Mukherjee, Indrajit, ja Srikanta Routroy. 2012. “Comparing the performance of neural networks developed by using Levenberg–Marquardt and Quasi-Newton with the gradient descent algorithm for modelling a multiple response grinding process”. *Expert systems with applications* 39 (3): 2397–2407. ISSN: 0957-4174. doi:10.1016/j.eswa.2011.08.087.

“Nmap: Remote OS detection”. 2016. Viitattu 19. tammikuuta. https://nmap.org/book/osdetect.html.

“Nmap: the network mapper - Free security scanner”. 2016. Viitattu 19. tammikuuta. https://nmap.org.

Nummenmaa, Lauri, Martti Holopainen ja Pekka Pulkkinen. 2016. *Tilastollisten menetelmien perusteet.* Sanoma Pro Oy. ISBN: 978-952-63-2979-6.

Om, Hari, ja Alok Kumar Gupta. 2015. “Feature selection and decision tree: A combinational approach for intrusion detection”. Teoksessa Israr ja Issac 2015, 27–47.

Pandya, Pramod. 2013. “Local area network security”. Teoksessa Vacca 2013, 263–283.

Patcha, Animesh, ja Jung-Min Park. 2007. “An overview of anomaly detection techniques: Existing solutions and latest technological trends”. *Computer networks* 51 (12): 3448–3470. ISSN: 1389-1286. doi:10.1016/j.comnet.2007.02.001.

Pharate, Abhishek, Harsha Bhat, Vaibhav Shilimkar ja Nalini Mhetre. 2015. “Classification of intrusion detection system”. *International journal of computer applications* 118 (7). doi:10.5120/20758-3163.

Ringberg, Haakon, Matthew Roughan ja Jennifer Rexford. 2008. “The need for simulation in evaluating anomaly detectors”. *SIGCOMM comput. commun. rev.* (New York, NY, USA) 38, numero 1 (tammikuu): 55–59. ISSN: 0146-4833. doi:10.1145/1341431.1341443.

Rojas, Raúl. 1996. *Neural networks: A systematic introduction.* Springer. ISBN: 3-540-60505-3.

Said, D., L. Stirling, P. Federolf ja K. Barker. 2011. “Data preprocessing for distance-based unsupervised intrusion detection”. Teoksessa *Privacy, security and trust (PST), 2011 ninth annual international conference on,* 181–188. Heinäkuu. doi:10.1109/PST.2011.5971981.

Saini, L. M., ja M. K. Soni. 2002. “Artificial neural network based peak load forecasting using Levenberg-Marquardt and quasi-Newton methods”. *IEEE proceedings - Generation, transmission and distribution* 149, numero 5 (syyskuu): 578–584. ISSN: 1350-2360. doi:10.1049/ip-gtd:20020462.

Salvador, S., ja P. Chan. 2004. “Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms”. Teoksessa *Tools with artificial intelligence, 2004. ICTAI 2004. 16th IEEE international conference on,* 576–584. Marraskuu. doi:10.1109/ICTAI.2004.50.

Sangkatsanee, Phurivit, Naruemon Wattanapongsakorn ja Chalermpol Charnsripinyo. 2011. “Practical real-time intrusion detection using machine learning approaches”. *Computer communications* 34 (18): 2227–2235. ISSN: 0140-3664. doi:10.1016/j.comcom.2011.07.001.

Schroedl, Stefan. 2010. “File exchange: Feature selection based on interaction information”. Viitattu 29. heinäkuuta 2016. https://se.mathworks.com/matlabcentral/fileexchange/26981-feature-selection-based-on-interaction-information.

Sharma, S. K., P. Pandey, S. K. Tiwari ja M. S. Sisodia. 2012. “An improved network intrusion detection technique based on k-means clustering via naïve Bayes classification”. Teoksessa *Advances in engineering, science and management (ICAESM), 2012 international conference on,* 417–422. Maaliskuu.

Sommer, R., ja V. Paxson. 2010. “Outside the closed world: On using machine learning for network intrusion detection”. Teoksessa *Security and privacy (SP), 2010 IEEE symposium on,* 305–316. Toukokuu. doi:10.1109/SP.2010.25.

Sperotto, A., G. Schaffrath, R. Sadre, C. Morariu, A. Pras ja B. Stiller. 2010. “An overview of IP flow-based intrusion detection”. *Communications surveys tutorials, IEEE* 12 (3): 343–356. ISSN: 1553-877X. doi:10.1109/SURV.2010.032210.00054.

Stakhanova, Natalia, Samik Basu ja Johnny Wong. 2007. “A taxonomy of intrusion response systems”. *International journal of information and computer security* 1 (1-2): 169–184. doi:10.1504/IJICS.2007.012248.

Stolfo, Salvatore J., Wei Fan, Wenke Lee, Andreas Prodromidis ja Philip K. Chan. 1999. "KDD-Cup-99 task description". Viitattu 7. huhtikuuta 2016. http://kdd.ics.uci.edu/databases/kddcup99/task.html.

Tan, Pang-Nin, Michael Steinbach ja Vipin Kumar. 2006. *Introduction to data mining.* Addison-Wesley. ISBN: 0-321-42052-7.

Tavallaee, M., E. Bagheri, W. Lu ja A. A. Ghorbani. 2009. "A detailed analysis of the KDD CUP 99 data set". Teoksessa *2009 IEEE symposium on computational intelligence for security and defense applications,* 1–6. Heinäkuu. doi:10.1109/CISDA.2009.5356528.

Tavallaee, M., N. Stakhanova ja A.A. Ghorbani. 2010. "Toward credible evaluation of anomaly-based intrusion-detection methods". *Systems, man, and cybernetics, Part C: Applications and reviews, IEEE transactions on* 40, numero 5 (syyskuu): 516–524. ISSN: 1094-6977. doi:10.1109/TSMCC.2010.2048428.

Thomas, Ciza, Vishwas Sharma ja N. Balakrishnan. 2008. "Usefulness of DARPA dataset for intrusion detection system evaluation". Teoksessa *Data mining, intrusion detection, information assurance, and data networks security 2008,* nide 6973. Maaliskuu. doi:10.1117/12.777341.

Vacca, John R., toimittanut. 2013. *Computer and information security handbook.* Morgan Kaufmann. ISBN: 9780123943972.

"Välimatka- eli intervalliasteikko". 2016. Viitattu 28. tammikuuta. http://www.stat.fi/meta/kas/intervalliastei.html.

Virvilis, Nikos, Oscar Serrano ja Luc Dandurand. 2014. "Big data analytics for sophisticated attack detection". *ISACA journal* 3:22–25. ISSN: 1944-1967, viitattu 12. huhtikuuta 2016. http://www.isaca.org/Journal/archives/2014/Volume-3/Pages/Big-Data-Analytics-for-Sophisticated-Attack-Detection.aspx.

Wang, W., ja R. Battiti. 2006. "Identifying intrusions in computer networks with principal component analysis". Teoksessa *Availability, reliability and security, 2006. ARES 2006. The first international conference on.* Huhtikuu. doi:10.1109/ARES.2006.73.

Wang, Wei, Xiangliang Zhang, S. Gombault ja S.J. Knapskog. 2009. “Attribute normalization in network intrusion detection”. Teoksessa *Pervasive systems, algorithms, and networks (ISPAN), 2009 10th international symposium on,* 448–453. Joulukuu. doi:10.1109/I-SPAN.2009.49.

Weidenbaum, Brian. 2012. “File exchange: Recode categorical variable into new binary variables”. Viitattu 22. syyskuuta 2016. https://se.mathworks.com/matlabcentral/fileexchange/36042-recode-categorical-variable-into-new-binary-variables.

Weller-Fahy, D. J., B. J. Borghetti ja A. A. Sodemann. 2015. “A survey of distance and similarity measures used within network intrusion anomaly detection”. *IEEE communications surveys tutorials* 17 (1): 70–91. ISSN: 1553-877X. doi:10.1109/COMST.2014.2336610.

West, Michael. 2013. “Preventing system intrusions”. Teoksessa Vacca 2013, 63–79.

Witten, I. H., ja Eibe Frank. 2005. *Data mining: Practical machine learning tools and techniques.* Morgan Kaufmann Series in data management systems. Morgan Kaufmann. ISBN: 9780120884070.

Wu, Shelly Xiaonan, ja Wolfgang Banzhaf. 2010. “The use of computational intelligence in intrusion detection systems: A review”. *Applied soft computing* 10 (1): 1–35. ISSN: 1568-4946. doi:10.1016/j.asoc.2009.06.019.

Zagouras, A., A. Kazantzidis, E. Nikitidou ja A.A. Argiriou. 2013. “Determination of measuring sites for solar irradiance, based on cluster analysis of satellite-derived cloud estimations”. *Solar energy* 97:1–11. ISSN: 0038-092X. doi:10.1016/j.solener.2013.08.005.

Zagouras, Athanassios. 2014. “File exchange: L-method”. Viitattu 14. heinäkuuta 2016. https://www.mathworks.com/matlabcentral/fileexchange/38771-l-method.

Zagouras, Athanassios, Rich H. Inman ja Carlos F.M. Coimbra. 2014. “On the determination of coherent solar microclimates for utility planning and operations”. *Solar energy* 102:173–188. ISSN: 0038-092X. doi:10.1016/j.solener.2014.01.021.

Zargar, S.T., J. Joshi ja D. Tipper. 2013. “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks”. *Communications surveys tutorials, IEEE* 15 (4): 2046–2069. ISSN: 1553-877X. doi:10.1109/SURV.2013.031413.00127.

Liitteet

A K-means: Lista valituista ominaisuuksista

Taulukko 18 sisältää JMI-, CMIM- ja CIFE-algoritmien valitsemat *UNSW_NB15_training-set.csv*-opetusdatajoukon ominaisuudet. Tiedot esitetään taulukossa kumulatiivisesti, eli esimerkiksi ominaisuuksien lukumäärän ollessa kolme, valitsi JMI-algoritmi datajoukon ominaisuudet *sbytes*, *smean* ja *ct_state_ttl*.

Ennen ominaisuuksien valintaa datajoukosta on karsittu *k*-meansille soveltumattomat ominaisuudet *proto*, *service*, *state*, *is_ftp_login*, *ct_ftp_cmd* ja *is_sm_ips_ports*.

Taulukko 18: *K*-means: Kolmella menetelmällä valitut ominaisuudet.

Ominaisuuksien lukumäärä	**Valitut ominaisuudet**		
	JMI	**CMIM**	**CIFE**
1	sbytes	sbytes	sbytes
2	smean	smean	smean
3	ct_state_ttl	ct_srv_dst	dloss
4	ct_srv_dst	sttl	sttl
5	dbytes	ct_state_ttl	ct_dst_src_ltm
6	sttl	ct_src_dport_ltm	ct_flw_http_mthd
7	ct_dst_sport_ltm	ct_dst_sport_ltm	trans_depth
8	sload	sload	response_body_len
9	ct_srv_src	ct_dst_src_ltm	sloss
10	rate	dbytes	ct_dst_ltm
11	ct_src_dport_ltm	ct_srv_src	ct_src_ltm
12	ct_dst_src_ltm	ct_dst_ltm	sinpkt
13	dttl	dload	ct_srv_src
14	dload	rate	dur
15	ct_dst_ltm	ct_src_ltm	ct_srv_dst

16	dmean	spkts	spkts
17	dur	sloss	dmean
18	dpkts	synack	djit
19	dinpkt	sjit	sjit
20	sinpkt	dinpkt	tcprtt
21	sloss	dur	swin
22	synack	sinpkt	sload
23	ct_src_ltm	dloss	ct_src_dport_ltm
24	spkts	dmean	synack
25	sjit	djit	dbytes
26	tcprtt	dpkts	dtcpb
27	dloss	ackdat	stcpb
28	djit	dttl	dwin
29	ackdat	ct_flw_http_mthd	dpkts
30	swin	tcprtt	ackdat
31	dtcpb	response_body_len	dload
32	stcpb	stcpb	rate
33	dwin	dtcpb	ct_dst_sport_ltm
34	ct_flw_http_mthd	swin	dinpkt
35	trans_depth	trans_depth	dttl
36	response_body_len	dwin	ct_state_ttl

B K-means: Normalisointimenetelmien vertailu

Taulukko 19: *K*-means: Normalisointimenetelmien vertailu kosinietäisyydellä (JMI).

	AUC (keskiarvo)	**Keskihajonta**	**Luottamusväli (99,9%)**
Z-piste	0,9279	0,0057	0,9260 – 0,9299
Minimi-maksimi	0,9279	0,0054	0,9260 – 0,9297
Logaritmi (kantaluku 10)	0,9287	0,0088	0,9257 – 0,9317
Ei normalisointia	0,8942	0,0082	0,8914 – 0,8969

Taulukko 20: *K*-means: Normalisointimenetelmien vertailu neliöidyllä Euklidisella etäisyydellä (JMI).

	AUC (keskiarvo)	**Keskihajonta**	**Luottamusväli (99,9%)**
Z-piste	0,9141	0,0046	0,9125 – 0,9157
Minimi-maksimi	0,9283	0,0044	0,9269 – 0,9298
Logaritmi (kantaluku 10)	0,9131	0,0108	0,9095 – 0,9168
Ei normalisointia	0,5643	0,0149	0,5592 – 0,5693

Taulukko 21: *K*-means: Normalisointimenetelmien vertailu Pearsonin korrelaatioetäisyydellä (JMI).

	AUC (keskiarvo)	**Keskihajonta**	**Luottamusväli (99,9%)**
Z-piste	0,9288	0,0058	0,9268 – 0,9308
Minimi-maksimi	0,9271	0,0052	0,9253 – 0,9289
Logaritmi (kantaluku 10)	0,9288	0,0079	0,9261 – 0,9315
Ei normalisointia	0,8936	0,0089	0,8906 – 0,8965

Taulukko 22: *K*-means: Normalisointimenetelmien vertailu Manhattan-etäisyydellä (PCA).

	AUC (keskiarvo)	**Keskihajonta**	**Luottamusväli (99,9 %)**
Z-piste	0,9044	0,0067	0,9022 – 0,9067
Minimi-maksimi	0,4298	0,0928	0,3992 – 0,4600
Logaritmi (kantaluku 10)	0,7340	0,1122	0,6970 – 0,7710
Ei normalisointia	0,4247	0,0611	0,4045 – 0,4449

Taulukko 23: *K*-means: Normalisointimenetelmien vertailu neliöidyllä Euklidisella etäisyydellä (PCA).

	AUC (keskiarvo)	**Keskihajonta**	**Luottamusväli (99,9 %)**
Z-piste	0,8875	0,0084	0,8847 – 0,8902
Minimi-maksimi	0,3989	0,0543	0,3810 – 0,4168
Logaritmi (kantaluku 10)	0,7476	0,1260	0,7061 – 0,7892
Ei normalisointia	0,4075	0,0572	0,3887 – 0,4264

Taulukko 24: *K*-means: Normalisointimenetelmien vertailu Pearsonin korrelaatioetäisyydellä (PCA).

	AUC (keskiarvo)	**Keskihajonta**	**Luottamusväli (99,9 %)**
Z-piste	0,9258	0,0049	0,9242 – 0,9274
Minimi-maksimi	0,5250	0,0297	0,5152 – 0,5348
Logaritmi (kantaluku 10)	0,7168	0,1568	0,6650 – 0,7684
Ei normalisointia	0,5767	0,0299	0,5669 – 0,5866

C K-means: Menetelmän arviointi testausdatalla

Taulukko 25: *K*-means: Luokittelun osuvuus (PCA).

Luokka	**Ryhmä**	**Osuvuus kahdella pääluokalla luottamusvälillä 99 %**	**Osuvuus kymmenellä ryhmällä luottamusvälillä 99 %**
Normaali	**Normal**	$0,8482 \pm 0,0121$	$0,8482 \pm 0,0121$
Haitta-liikenne	**Analysis**	$0,9030 \pm 0,0253$	0
	Backdoor	$0,9271 \pm 0,0079$	0
	DoS	$0,9450 \pm 0,0072$	$0,0082 \pm 0,0099$
	Exploits	$0,8930 \pm 0,0177$	$0,4232 \pm 0,0091$
	Fuzzers	$0,6835 \pm 0,0399$	$0,3466 \pm 0,0414$
	Generic	$0,9966 \pm 0,0006$	$0,9796 \pm 0,0006$
	Reconnaissance	$0,8640 \pm 0,0219$	$0,3391 \pm 0,0057$
	Shellcode	$0,8505 \pm 0,0239$	0
	Worms	$0,6890 \pm 0,0476$	0

Taulukko 26: *K*-means: Instanssien ennustetut ja todelliset ryhmät (PCA).

		Ennustettu ryhmä									
		Normal	***Analysis***	***Backdoor***	***DoS***	***Exploits***	***Fuzzers***	***Generic***	***Reconnaissance***	***Shellcode***	***Worms***
Todellinen ryhmä	**Normal**	**47502**	0	0	45	1754	4014	2183	502	0	0
	Analysis	194	**0**	0	11	344	88	959	404	0	0
	Backdoor	127	0	**0**	18	196	111	914	381	0	0
	DoS	674	0	0	**100**	1916	590	6028	2955	0	0
	Exploits	3573	0	0	166	**14133**	3619	7593	4309	0	0
	Fuzzers	5755	0	0	32	1517	**6302**	2828	1749	0	0
	Generic	135	0	0	7	311	164	**39186**	197	0	0
	Reconn-aissance	1426	0	0	28	2049	1830	1600	**3557**	0	0
	Shellcode	169	0	0	1	198	247	55	462	**0**	0
	Worms	40	0	0	0	38	44	0	8	0	**0**

D Tukivektorikone: Parametrien etsintä

Taulukko 27: Kaksiluokkainen SVM: Parametrien etsinnän ensimmäinen vaihe.

		C									
		2^-2	**2^0**	**2^2**	**2^4**	**2^6**	**2^8**	**2^10**	**2^11**	**2^12**	**2^13**
γ	**2^-9**	83,13	83,01	84,07	85,23	91,02	93,13	93,83	94,09	94,32	94,40
	2^-6	83,68	85,93	88,41	93,36	94,08	94,65	95,04	95,11	95,19	95,28
	2^-3	87,26	91,32	94,08	94,85	95,04	95,34	95,74	95,89	95,84	95,97
	2^0	92,65	94,36	95,12	95,55	95,95	96,05	96,02	96,13	96,26	96,16
	2^3	93,95	95,03	95,61	95,85	96,07	95,97	95,67	95,74	95,65	95,52
	2^4	94,34	95,17	95,56	95,91	95,73	95,61	95,29	95,20	95,03	94,73

Taulukko 28: Kaksiluokkainen SVM: Parametrien etsinnän toinen vaihe.

		C							
		2^5	**2^6**	**2^7**	**2^8**	**2^9**	**2^10**	**2^11**	**2^12**
γ	**2^-1**	95,61	95,74	95,75	95,83	95,96	96,11	96,08	96,03
	2^0	95,67	95,95	96,02	96,05	96,05	96,02	96,13	96,26
	2^1	95,84	95,97	95,94	96,04	96,04	96,08	96,16	96,17
	2^2	95,90	95,91	96,11	96,02	96,07	96,00	95,98	95,99
	2^3	95,92	96,07	95,91	95,97	95,83	95,67	95,74	95,65
	10	95,85	95,85	95,88	95,93	95,73	95,66	95,52	95,51
	12	95,78	95,90	95,65	95,75	95,71	95,61	95,42	95,33

Taulukkojen 31 ja 30 ajat on ilmoitettu sekunteina.

Taulukko 29: Kaksiluokkainen SVM: Parametrien etsinnän kolmas vaihe, osa 1.

		C										
		96	**128**	**160**	**192**	**224**	**256**	**320**	**384**	**448**	**512**	**640**
γ	**0,75**	95,92	95,84	95,91	95,96	95,93	95,90	96,04	95,92	95,98	96,07	96,07
	1	95,84	95,94	95,96	96,03	95,90	96,10	95,91	96,05	96,07	96,11	96,17
	1,25	95,84	95,97	96,06	96,01	96,09	95,96	96,08	96,15	96,13	96,03	96,19
	1,5	96,05	95,83	96,00	96,05	95,98	96,05	96,11	95,97	96,03	96,05	96,09
	1,75	96,02	96,07	96,09	95,98	95,96	95,99	96,08	96,12	96,06	96,06	96,05
	2	96,04	96,02	95,95	96,05	96,02	95,97	96,07	96,10	96,12	95,99	96,10
	2,25	95,93	96,03	96,03	96,14	96,08	96,00	95,97	96,07	96,03	96,13	96,03

Taulukko 30: Kaksiluokkainen SVM: Parametrien etsinnän kolmas vaihe, osa 2.

		C										
		768	**896**	**1024**	**1280**	**1536**	**1792**	**2048**	**2560**	**3072**	**3584**	**4096**
γ	**0,75**	96,06	96,03	96,11	96,04	96,12	96,01	96,06	96,12	96,12	96,14	96,24
	1	96,14	95,93	96,10	96,13	96,20	96,21	96,24	96,16	96,14	96,17	96,15
	1,25	96,03	96,09	96,21	96,10	96,12	96,08	96,14	96,12	96,13	96,15	96,12
	1,5	96,01	96,15	96,14	96,18	96,16	96,13	96,08	96,07	96,19	96,09	96,08
	1,75	96,11	96,11	96,14	96,12	96,12	96,16	96,13	96,23	96,13	96,09	96,18
	2	96,28	96,17	96,17	96,17	96,03	96,22	96,19	96,14	96,08	96,18	96,11
	2,25	96,12	96,05	96,11	96,19	96,17	96,10	96,13	96,19	96,14	96,09	96,09

Taulukko 31: Kaksiluokkainen SVM: Parametrien etsinnän kolmannen vaiheen ajankäyttö, osa 1.

		C										
		96	**128**	**160**	**192**	**224**	**256**	**320**	**384**	**448**	**512**	**640**
γ	**0,75**	835	831	851	875	891	909	959	980	1033	1050	1131
	1	839	857	883	899	924	967	1005	1068	1083	1145	1240
	1,25	853	869	898	930	975	993	1053	1110	1177	1178	1296
	1,5	881	900	923	970	1002	1033	1102	1149	1223	1281	1444
	1,75	904	919	947	992	1026	1037	1148	1217	1254	1331	1430
	2	900	947	978	1029	1070	1089	1162	1253	1369	1399	1508
	2,25	930	962	1019	1044	1124	1152	1200	1301	1375	1421	1583

Taulukko 32: Kaksiluokkainen SVM: Parametrien etsinnän kolmannen vaiheen ajankäyttö, osa 2.

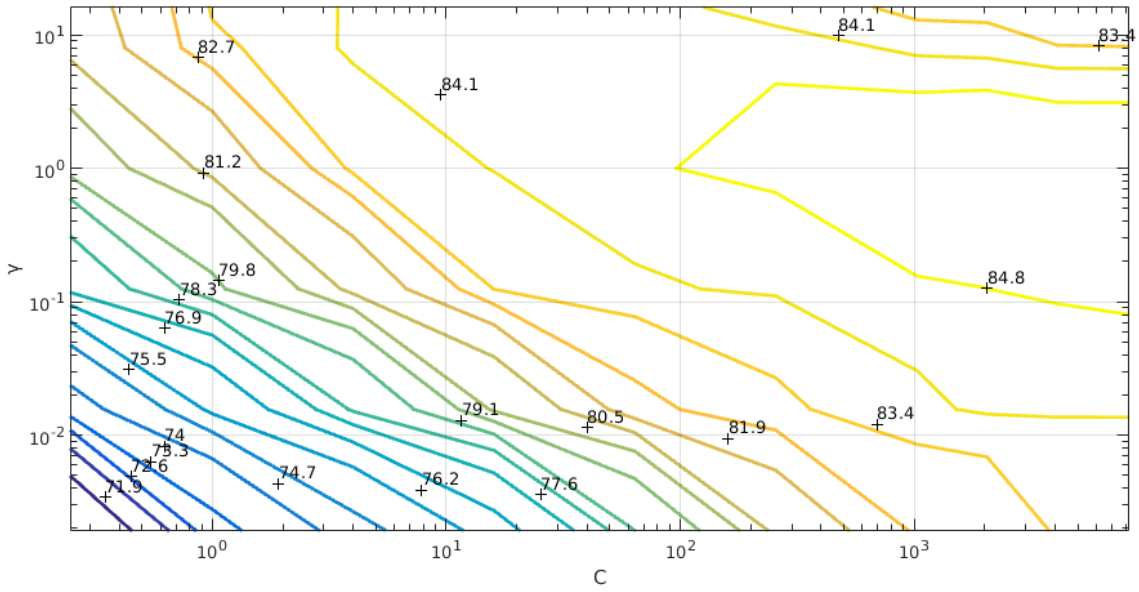
		C										
		768	**896**	**1024**	**1280**	**1536**	**1792**	**2048**	**2560**	**3072**	**3584**	**4096**
γ	**0,75**	1206	1261	1358	1460	1576	1661	1863	2248	2140	2337	2551
	1	1269	1389	1444	1626	1782	1852	2077	2301	2478	2976	2997
	1,25	1409	1516	1621	1767	1974	2090	2176	2574	2805	3430	3516
	1,5	1519	1568	1667	2000	2115	2209	2385	2709	3130	3620	3743
	1,75	1557	1699	1803	2080	2250	2462	2583	3162	3373	3862	4234
	2	1669	1781	1939	2115	2316	2678	2824	3308	3515	4291	4464
	2,25	1786	1945	2060	2271	2504	2761	2819	3409	3901	4256	4676

Taulukko 33: Moniluokkainen SVM: Parametrien etsinnän ensimmäinen vaihe.

		C									
		2^-2	**2^0**	**2^2**	**2^4**	**2^6**	**2^8**	**2^10**	**2^11**	**2^12**	**2^13**
γ	**2^-9**	71,14	73,87	75,30	76,68	79,09	81,48	82,82	82,92	83,48	83,63
	2^-6	74,51	76,40	78,47	80,64	82,53	83,29	84,00	84,23	84,21	84,19
	2^-3	77,88	79,70	81,51	83,41	84,05	84,23	84,81	84,83	85,04	85,26
	2^0	80,16	81,53	83,58	84,15	84,75	85,21	85,42	85,55	85,44	85,44
	2^3	81,55	83,27	84,31	84,81	84,64	84,40	83,88	83,77	83,42	83,40
	2^4	81,65	83,46	84,26	84,53	84,28	83,76	83,08	83,06	82,77	82,72

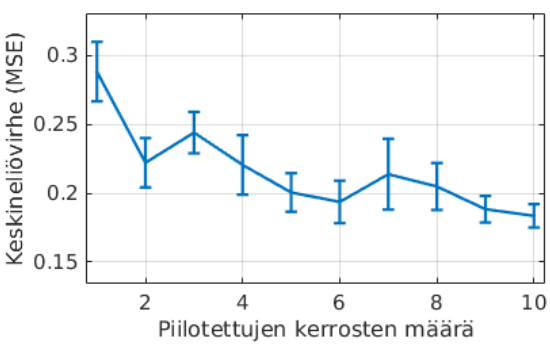
Taulukko 34: Moniluokkainen SVM: Parametrien etsinnän toinen vaihe.

		C										
		448	**512**	**640**	**768**	**896**	**1024**	**1280**	**1536**	**1792**	**2048**	**2560**
γ	**0,5**	85,02	85,01	85,08	85,17	85,36	85,32	85,32	85,37	85,39	85,44	85,57
	0,75	85,20	85,23	85,36	85,41	85,44	85,47	85,54	85,48	85,47	85,46	85,51
	1	85,32	85,32	85,41	85,40	85,46	85,49	85,51	85,46	85,54	85,43	85,47
	1,25	85,47	85,25	85,43	85,40	85,51	85,56	85,40	85,37	85,52	85,48	85,28
	1,5	85,33	85,38	85,33	85,29	85,46	85,42	85,39	85,38	85,44	85,28	85,28
	1,75	85,08	85,23	85,43	85,26	85,19	85,23	85,25	85,27	85,43	85,22	85,16

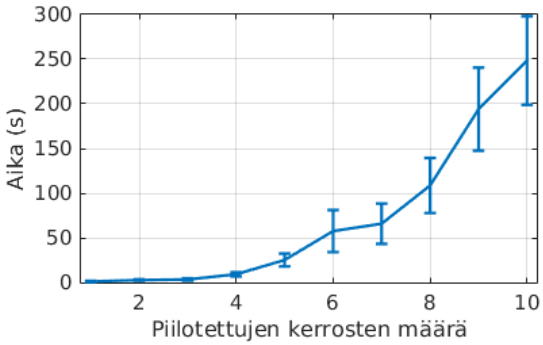


Kuvio 33: Moniluokkainen SVM: γ- ja C-parametrien etsinnän ensimmäinen vaihe.

E Keinotekoiset neuroverkot: Oppimismenetelmien vertailu

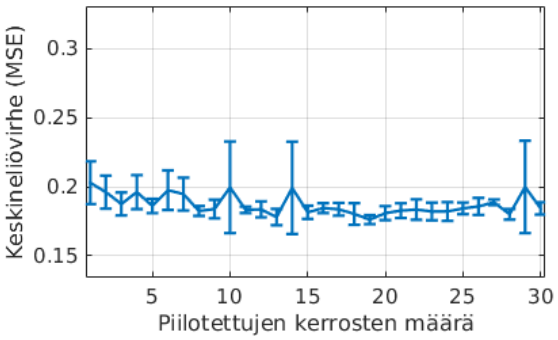


(a) Keskineliövirhe (MSE)

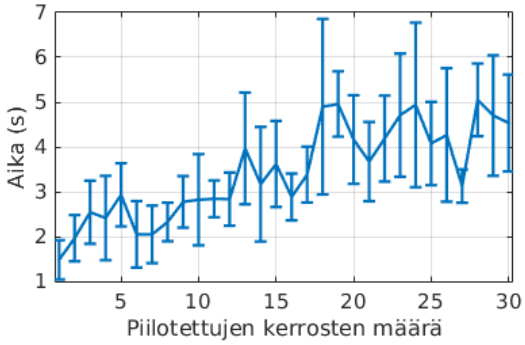


(b) Opetusvaiheen ajankäyttö

Kuvio 34: BFGS-algoritmi.

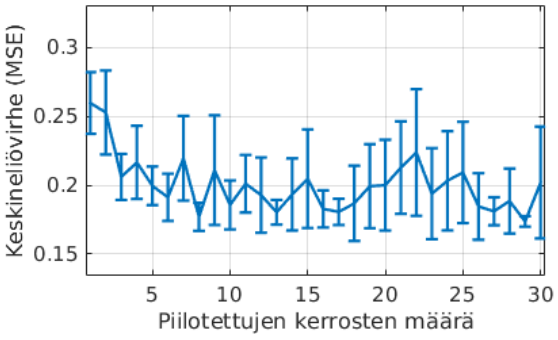


(a) Keskineliövirhe (MSE)

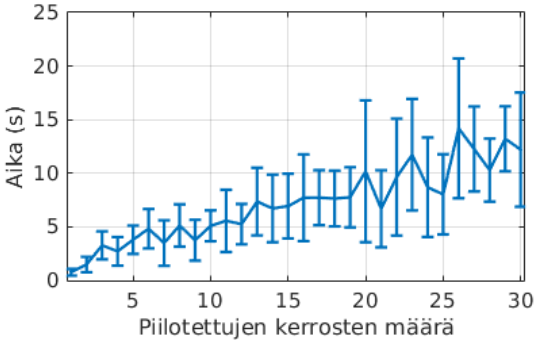


(b) Opetusvaiheen ajankäyttö

Kuvio 35: Skaalattu liittogradienttimenetelmä.

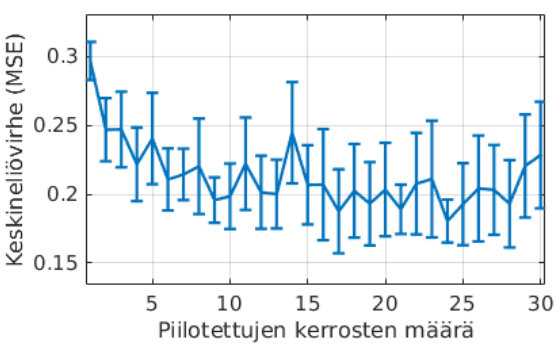


(a) Keskineliövirhe (MSE)

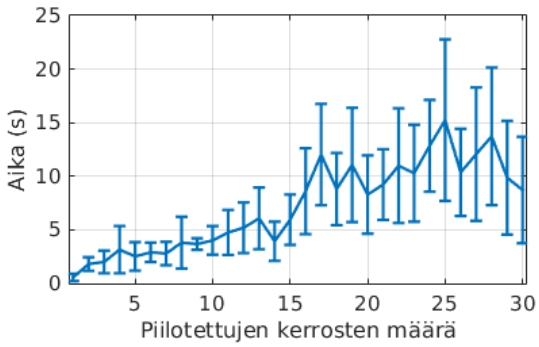


(b) Opetusvaiheen ajankäyttö

Kuvio 36: Powellin ja Bealen liittogradienttimenetelmä.

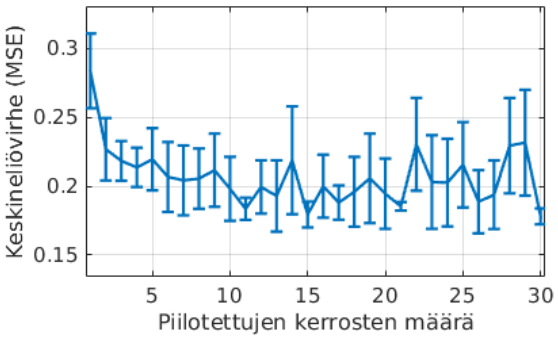


(a) Keskineliövirhe (MSE)

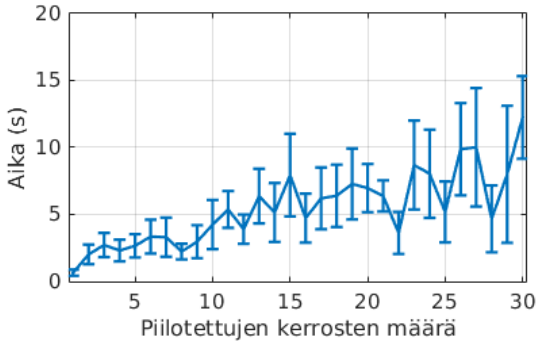


(b) Opetusvaiheen ajankäyttö

Kuvio 37: Fletcherin ja Reevesin liittogradienttimenetelmä.

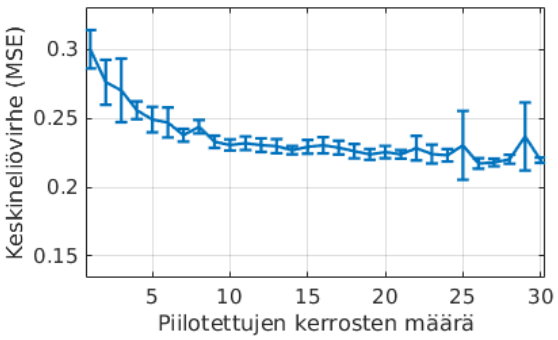


(a) Keskineliövirhe (MSE)

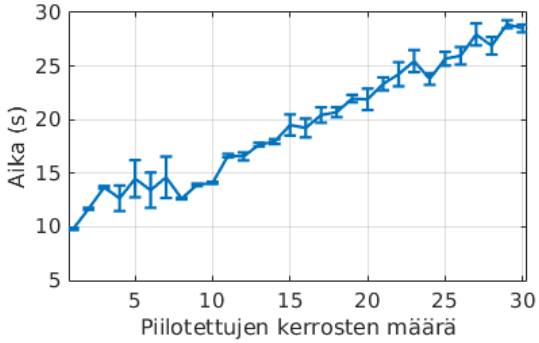


(b) Opetusvaiheen ajankäyttö

Kuvio 38: Polakin ja Ribièren liittogradienttimenetelmä.

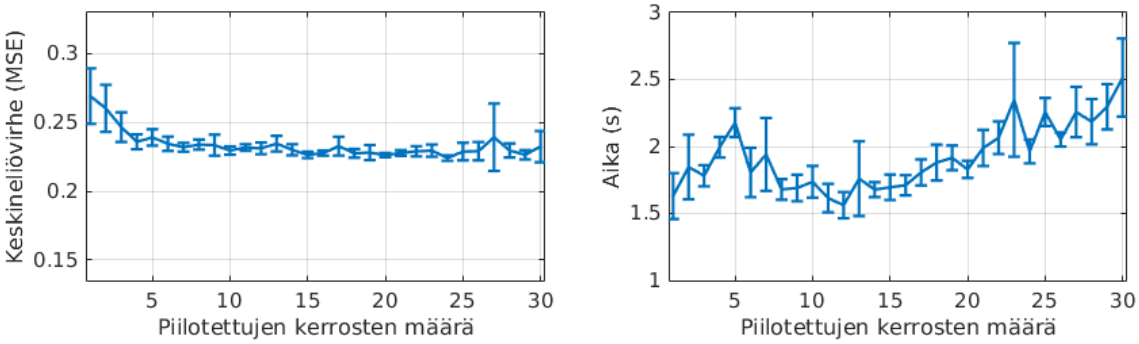


(a) Keskineliövirhe (MSE)



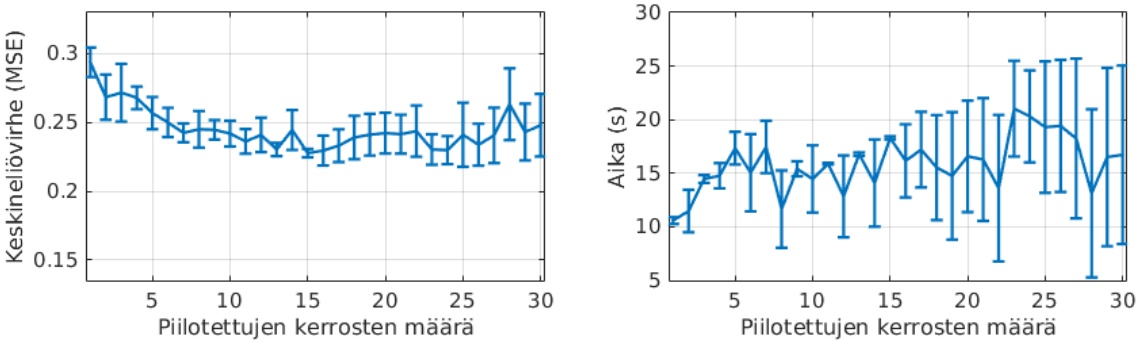
(b) Opetusvaiheen ajankäyttö

Kuvio 39: Vastavirta-algoritmi eli gradienttimenetelmä.



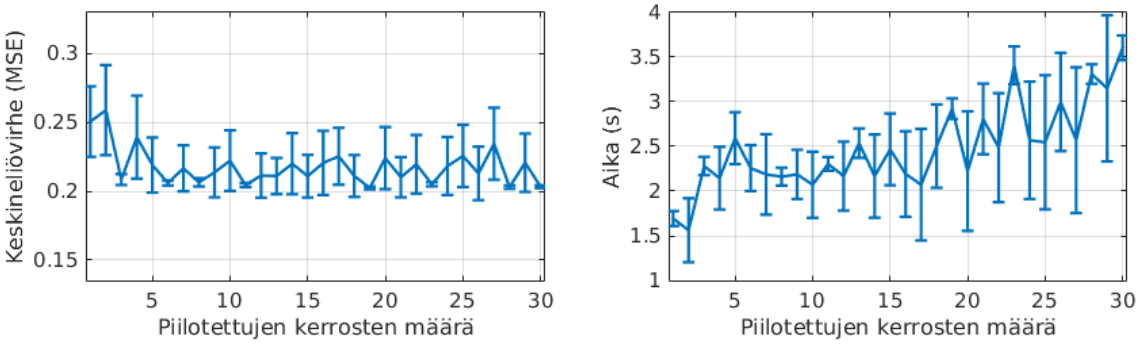
(a) Keskineliövirhe (MSE) (b) Opetusvaiheen ajankäyttö

Kuvio 40: VLBP-algoritmi.



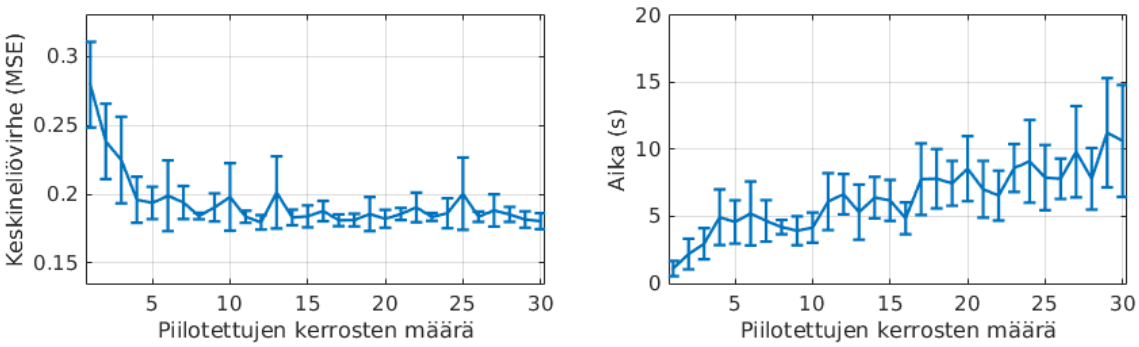
(a) Keskineliövirhe (MSE) (b) Opetusvaiheen ajankäyttö

Kuvio 41: MOBP-algoritmi.



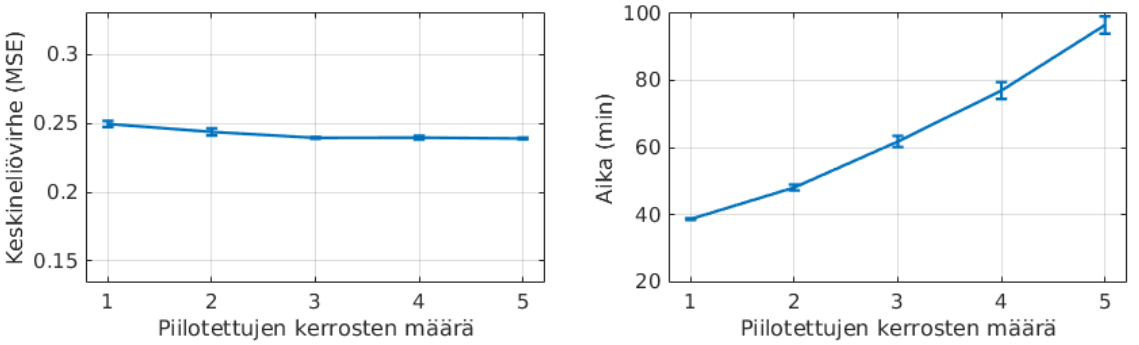
(a) Keskineliövirhe (MSE) (b) Opetusvaiheen ajankäyttö

Kuvio 42: VLBP- ja MOBP-algoritmien yhdistelmä.



(a) Keskineliövirhe (MSE) (b) Opetusvaiheen ajankäyttö

Kuvio 43: OSS-menetelmä.



(a) Keskineliövirhe (MSE) (b) Opetusvaiheen ajankäyttö

Kuvio 44: Moniluokkainen bayesiläinen regularisointi.