Koneoppiminen tietoverkoissa

Ilkka Lehikoinen

Helsinki 12.12.2018

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen osasto

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Studieprogram — Study Programme
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytiede

Tekijä — Författare — Author
Ilkka Lehikoinen

Työn nimi — Arbetets titel — Title

Koneoppiminen tietoverkoissa

Ohjaajat — Handledare — Supervisors
Jussi Kangasharju

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	12.12.2018	51 sivua

Tiivistelmä — Referat — Abstract

Internetin kasvun myötä lisääntynyt tietoliikenne tuottaa vaikeuksia tietoverkkojen tehokkaaseen ja luotettavaan hallinnointiin. Tietoverkoissa liikkuu paljon dataa, minkä lisäksi verkkolaitteissa myös syntyy paljon dataa. Tietoverkoissa liikkuvaa ja syntyvää dataa hyödynnetään tietoverkkojen tehokkaassa hallinnoinnissa, mutta liikennemäärien kasvaessa tarvitaan automaattisia menetelmiä hallinnointidatan käsittelyyn.

Koneoppiminen on useasta eri menetelmästä koostuva tekoälyn ala, jossa datasta oppiminen automatisoidaan. Koneoppimista hyödynnetään tietoverkkojen hallinnoinnissa usealla eri tavalla. Tietoverkkojen hallinnointiin liittyviä tehtäviä voidaan yksinkertaistaa luokitteluksi tai ryhmittelyksi, jolloin näihin voidaan soveltaa useita erilaisia koneoppimismenetelmiä. Tutkielmassa on kirjallisuusanalyysin avulla selvitetty eri koneoppimismenetelmien soveltuvuutta erilaisiin tietoverkon hallinnointiin liittyviin tehtäviin. Tutkimuksessa havaittiin, että eri koneoppimismenetelmiä oli kokeiltu laajalti eri tehtävien ratkaisemiseksi. Useimmat menetelmät ratkaisivat ongelman, mutta käytettävyydessä tai tarkkuudessa löytyi eroja, niin että käyttökelpoisimmiksi luokittelijoiksi osoittautuvat päätöspuut ja neuroverkot.

ACM Computing Classification System (CCS):
Networks∼Network algorithms
Networks∼Network performance evaluation
Computing methodologies∼Machine learning

Avainsanat — Nyckelord — Keywords
koneoppiminen, tietoverkko

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Yksi merkittävin talouskasvuun vaikuttanut tekijä on Internet. Internetin kaikkialle ulottuva tietoverkko on mahdollistanut nopean, tehokkaan ja edullisen tavan siirtää tietoa ympäri maailmaa. Tietoverkkojen on kyettävä siirtämään dataa tehokkaasti, jotta tämä olisi mahdollista on tärkeää tietää verkkoon kytkettyjen laitteiden ja reittien käytettävissä olevat resurssit kullakin ajanhetkellä. Tietoverkot eivät ainoastaan siirrä dataa vaan samalla myös tuottavat sitä verkkojen hallinnointia varten. Verkkolaitteiden lokitietoihin kertyy tietoa mm. siirretystä tiedosta, siirron onnistumisesta, laitteiden ja reittien käyttöasteista. Lisäksi reitittimet näkevät välittämiensä pakettien otsaketiedot, jotka voidaan tarvittaessa tallentaa mahdollista myöhempää käyttöä varten.

Maailmalla syntyy kasvavassa määrin yhä enemmän tietoa, jonka tehokas hyödyntäminen manuaalisesti ei ole enää mahdollista. Suurten datamäärien hyödyntämiseen kehitetään uusia menetelmiä, joista yksi käytetyimmistä on koneoppiminen. Koneoppimisen avulla suuresta tietomassasta erityyppistä dataa pystytään automaattisesti ja tehokkaasti jalostamaan arvokasta tietoa päätöksenteon tueksi tai toiminnan ohjaamiseksi. Eräs koneoppimiseen tietoverkoissa liittyvä ongelma on tietoverkoissa syntyvä datan paikallisuus. Tietoverkojen toiminta on perusolemukseltaan hajautettua, joka tekee paikallisesti syntyvän datan hyödyntämisestä tietoverkkojen hallintaan koneoppimisen menetelmin haastavampaa.

Koneoppinen jaetaan tyypillisesti ohjattuun (supervised) ja ohjaamattomaan (unsupervised) oppimiseen sekä vahvistusoppimiseen (reinforcement learning). Ohjattua oppimista varten tarvitaan dataa, jossa syöttötietojen lisäksi on käytettävissä syöttötiedoista seurannut tulos. Ohjatussa oppimisessa järjestelmä osaa kertoa, mikä on todennäköisin tulos annetulla datalla. Ohjaamattomassa oppimisessa ei ole syöttödatan lisäksi käytettävissä siihen liittyvää tulosta tai sitä ei hyödynnetä oppimiseen. Ohjaamaton oppiminen soveltuu esimerkiksi datan ryhmittelyyn. Eräänä sovelluksena ohjaamattomalle oppimiselle on selvittää aineistosta kuinka monesta eri kategoriasta se koostuu. Vahvistusoppimisessa seurataan sovelluksen tilaa ja sen perusteella positiivisilla vahvistavilla ja negatiivisilla rankaisevilla signaaleilla ohjataan mallin oppimista.

Tietoverkkojen laitteissa, kuten reitittimissä ja kytkimissä, on kasvavassa määrin enemmän laskentatehoa ja muistia datan käsittelemiseen. Tehokkaammat laitteet mahdollistavat laitteissa syntyvän datan käsittelemisen koneoppimisen menetelmin. Tietoverkoissa on useita tehtäviä, joihin voidaan soveltaa koneoppimista, kuten esim. verkkolaitteiden toiminnan analysointi, verkossa liikkuvan datan luokittelu, verkon toiminnan analysointi ja siihen liittyvä reititys, sekä verkon tilan tarkkailu mahdollisten vikojen tai kyberhyökkäysten havaitsemiseksi.

Koneoppimisessa tietoverkon tehtäviä voidaan monesti pelkistää mm. luokitteluksi ja regressioksi. Näiden tehtävien toteuttamiseksi on käytettävissä erilaisia koneoppimisen menetelmiä, kuten lineaarinen regressio, neuroverkot, tukivektorikoneet (Support Vector Machines, SVM) ja päättelypuut, sekä eri menetelmien muunnelmia ja

yhdistelmiä. Näillä menetelmillä on omat hyvät ja huonot puolensa sovellettaessa niitä tietoverkkojen erilaisiin tehtäviin.

Tutkielman tavoitteena on selvittää miten eri koneoppimismenetelmiä on käytetty tietoverkkojen tehtäviin ja miten hyvin ne soveltuvat tietoverkkojen toiminnan parantamiseen. Lisäksi tutkielmassa pyritään selvittämään onko jokin menetelmä yleisesti toisia menetelmiä parempi vai onko eri menetelmien paremmuudessa eroja eri tehtävien kesken. Tutkielmassa keskitytään kiinteän verkon tehtäviin ja siksi sen ulkopuolelle rajataan mobiili- ja sensoriverkot.

Aihetta käsitellään vertailevan kirjallisuusanalyysin avulla.

Tutkielmassa lähdetään liikkeelle koneoppisesta, esitellään sen periaatteet ja menetelmiä tarvittavalla laajuudella ja tuodaan esille niiden erityisominaisuuksia tietoverkon tehtävien suhteen. Seuraavaksi esitellään tietoverkkojen toimintaa lyhyesti. Sen jälkeen käsitellään yksi kerrallaan tietoliikenteeseen liittyviä tehtäviä, joita koneoppimisella on pyritty ratkaisemaan. Jokaiselle tehtävälle esitetään käytettyjä koneoppimismenetelmiä sekä pyritään löytämään kuhunkin tehtävään parhaiten sopiva menetelmä.

Yhteenvetona saadaan matriisi, jossa on pystyakselilla tietoverkojen tehtävät ja vaaka-akselilla eri koneoppimismenetelmät. Matriisista voidaan arvioida eri menetelmien yleisyyttä tietoverkkoihin soveltamisessa sekä eri verkonhallinnan menetelmiin liittyvä kiinnostus koneoppimisen tuomiin vaihtoehtoihin.

2 Koneoppiminen

Digitalisoituva maailma tuottaa suuren määrän dataa, jota ei ole mahdollista käsitellä tehokkaasti manuaalisesti. Suurten datamäärien tehokas käsittely edellyttää automaatiota, johon koneoppiminen tuo yhden vaihtoehdon. Koneoppiminen koostuu menetelmistä, joilla suuresta joukossa dataa voidaan muodostaa malleja. Suurella datamäärällä opetettujen mallien avulla voidaan ratkaista aihepiirin tehtäviä ennennäkemättömälle datalle [29].

Koneoppiminen on tapa ratkaista tiedonkäsittelyongelma, kun ongelma on joko liian monimutkainen tai vaikeasti ymmärrettävissä ohjelmoijan ratkaistavaksi. Tällainen tehtävä voi olla vaikka tunnistaa onko annetussa valokuvassa kissa vai koira. On erittäin vaikeaa ohjelmoida ohjelma, joka osaa eritellä kuvista tunnusmerkit, jotka erottavat kissat koirista. Koneoppimisessa ohjelmalle annetaan syötedatana kuvia kissoista ja koirista mukaan lukien tieto kumpaan kategoriaan kuvan eläin kuuluu, ja ohjelma opettelee mitkä piirteet kuvissa viittaavat siihen että valokuvassa on todennäköisemmin kissa kuin koira. Opittujen piirteiden perusteella malli sitten luokittelee kumpaan ryhmään sille syötetyt kuvat kuuluvat.

2.1 Taustaa

Tekoälyllä tarkoitetaan ohjelmistoja, jotka suorittavat jonkinlaisia älykkyyttä vaativia tehtäviä. Älykkyyttä itsessään on hankala määritellä ja siksi myös tekoälyn määritelmä jää heikoksi. Tekoälyyn kuuluvat mm. asiantuntijajärjestelmät, luonnollisen kielen käsittely, robotiikka ja koneoppiminen.

Koneoppiminen on tekoälyn osa-alue, joka koostuu oppimiseen perustuvista menetelmistä. Fadullah et al. on katsauksessaan koneoppimisesta tietoverkkojen ohjauksessa [15] luokitellut koneoppisen kolmeen luokkaan: Ohjattu oppiminen, vahvistusoppiminen ja ohjaamaton oppiminen. Oheisessa luettelossa on listattuna myös joitakin yleisempiä menetelmiä. Luettelosta nähdään kuinka monia menetelmiä voidaan soveltaa erityyppisiin oppimistehtäviin.

- Ohjattu oppiminen
  - Luokittelu
    - Binääriset päätöspuut
    - Naiivi Bayes-luokittelijat
    - Neuroverkot
    - Konvoluutioneuroverkot
    - Syvät uskomusverkot
    - Toistuvat neuroverkot
  - Regressio
    - Neuroverkot
    - Puut
- Vahvistusoppiminen
    - Syvät Q-verkot
    - Tupla Q-oppiminen
    - Priorisoitu kokemuksen toisto
- Ohjaamaton oppiminen
  - Ulottuvuuksien vähentäminen
    - Pinotut autoenkooderit
    - Paikallinen lineaarinen upotus
    - Auto-assosioivat neuroverkot
  - Klusterointi
    - Konvoluutioneuroverkot
    - Syvät uskomusverkot
    - K-Means

- Tiheysarvio
  - Gaussin yhdistelmä
  - Kernelin tiheys
  - Syvät Boltzmannin koneet
  - Yleistetyt kohinanpoisto autoenkooderit

Koneoppiminen käsittää myös menetelmiä, jotka eivät sovi tähän kolmijakoon, kuten ohjattua ja ohjaamatonta oppimista yhdistävät puoliohjatut (semi-supervised) oppimismenetelmät, jotka hyödyntävät merkitsemätöntä dataa ohjattuun oppimiseen. Muitakin oppimismenetelmiä on kuten esimerkiksi siirto-oppiminen (transfer learning) ja moninäkymäoppiminen (multi-view learning). Siirto-oppimisessa on useampia näyteryhmiä, jotka saattavat tulla eri jakaumista. Yhdestä näyteryhmästä saatua oppia hyödynnetään toisiin näyteryhmiin. Moninäkymäoppimisessa näytteestä on useampi näkymä, esimerkiksi kuva ja sitä vastaava kuvaus tai sitten on sama teksti useammalla kielellä.

Koneoppiminen lähtee tehtävästä eli ongelmasta, johon haetaan ratkaisua. Tehtävän ratkaisemiseksi tarvitaan aihepiiriin liittyvää dataa. Data koostuu kohteena olevan aihepiirin objekteista. Objektit koostuvat niitä kuvaavista ominaispiirteistä. Datan avulla koneoppimisalgoritmit tuottavat mallin, joka on tehtävän ratkaisu.

2.1.1 Tilastotiede ja todennäköisyyslaskenta

Koneoppimisella on vahva perusta tilastotieteessä ja todennäköisyyslaskennassa. Monet koneoppimisessa käytetyt aineistot ovat muodostuneet tilastollisista jakaumista, joko diskreeteistä jakaumista, kuten Bernoulli-, Binomi- tai Poisson-jakauma, tai jatkuvista jakaumista kuten Normaalijakauma tai Beta-jakauma. Kohteena olevien ilmiöiden ollessa tilastollisesti jakautunut on luontevaa käsitellä ongelmia tilastotieteen käsitteiden avulla.

Koneoppimisalgoritmi koostuu aineistosta, virhefunktiosta, optimointiproseduurista ja mallista. Aineistoon sisältyvät havainnot sekä niihin mahdollisesti liittyvät tulokset. Virhefunktion tehtävänä on arvioida mallin hyvyyttä käytetyillä parametreilla. Optimointiproseduuri ratkaisee tehtävää kuvaavan optimointiongelman, kun käytetään opetusdataa eikä tulevaa dataa. Jos malli on lineaarinen optimointi voidaan ratkaista suljetusta muodosta, mutta epälineaarisessa tapauksessa joudutaan turvautumaan numeerisiin menetelmiin, kuten esimerkiksi gradienttimenetelmään.

2.1.2 Datan keräys ja esikäsittely

Dataa kerätään monin tavoin monista paikoista. Erilaisten sensorit ovat halpoja ja niiden tuottama data on usein digitaalista tai helposti digitalisoitavissa. Dataa saadaan paitsi mittauksien tuloksina, myös ihmisten syöttämänä ja laitteiden tuottamana. Dataa on myös halpaa siirtää pitkiäkin matkoja Internetin välityksellä ja nykyiset kiintolevyt ovat edullisia ja niihin voidaan tallentaa suuri määrä dataa.

Tämä on johtanut siihen, että dataa keräävät useat tahot, niin yksityiset yritykset, julkiset tutkimuslaitokset kuin myös verkko-operaattoritkin.

Ratkaistaessa tehtävää koneoppimismenetelmillä on hankittava aihepiiriin kuluvaa dataa. Datan on oltava sellaista, että sen avulla on mahdollista ratkaista ongelma. Data on harvoin sellaisenaan koneoppimisalgoritmien hyödynnettävissä vaan se vaatii esikäsittelyä. Esikäsittelyn tarkoituksena on saattaa ongelman ratkaisemiksi hankittu data sellaiseen muotoon, että ohjelmat kykenevät käsittelemään sen tehokkaasti. Kaikki aihepiiriin liittyvä data ei ole sopivaa tai välttämätöntä tehtävän ratkaisemiseksi. Jos datasta joudutaan tekemään jotain oletuksia, esimerkiksi siitä miten se hankittu, on tarpeen tallentaa nämä oletukset, koska niitä tarvitaan myös tulosten analysoinnissa. Datasta on myös hyvä tietää kuinka hyvin se kattaa tehtävän koko aihepiirin, esimerkiksi jääkö jokin osa-alue huomioimatta. On hyvä selvittää onko osa datasta sellaista jota ei tarvita tehtävän ratkaisuun, kuten vaikka tieto viikonpäivästä tutkittaessa tähtitaivaan ilmiöitä. Ylimääräisen datan poistaminen nopeuttaa ohjelman toimintaa.

Data saattaa tulla useista eri lähteistä, jolloin se voi tulla myös monessa eri muodossa, osa voi olla taulukkona, osa relaatiokannassa, tai sovelluskohtaisessa tiedostomuodossa. Datan muotoilu on muutettava sitä käytettävien ohjelmistojen hallitsemaan muotoon.

Data saattaa sisältää myös dataobjekteja, joissa on virheellisiä tai puuttuvia tietoja. Ennen kuin dataa voidaan käyttää on se käytävä läpi mahdollisten puutteiden havaitsemiseksi ja tehtävät tarvittavat toimet sen saattamiseksi käyttökelpoiseksi. Jos dataa on runsaasti niin virheelliset dataobjektit voidaan yksinkertaisesti vain poistaa ilman, että ratkaisun tarkkuus olennaisesti heikkenee. Jos taas dataa on niin vähän, että virheellisen datan poistaminen heikentää saavutettavissa olevaa mallin tarkkuutta olennaisesti, täytyy virheellinen tieto korvata nollatiedolla, keskiarvolla tai jollain muulla tehtävän kannalta sopivalla arvolla.

Jos dataa on paljon on hyödyllistä käyttää siitä vain osaa, jolloin tarvitaan vähemmän prosessointi- ja muistiresursseja, joka nopeuttaa käsittelyä ilman, että mallin tarkkuus olennaisesti heikkenee. Tämä voidaan tehdä esimerkiksi satunnaisesti näytteistämällä datasta.

Käytettävä algoritmi ja tehtävän luonne määrittävät missä muodossa käytettävän datan on oltava. Liukulukuja voidaan joutua muuttaa kokonaisluvuiksi tai päin vastoin, eri lähteistä kerätyt tiedot voivat olla eri mittayksiköissä, jolloin ne on muutettava samoiksi yksiköiksi. Jotkut arvot voidaan joutua normalisoimaan esimerkiksi välille 0...1. Joissain tapauksissa aineistossa olevia tietoja voi joutua hajottamaan, esimerkiksi tapahtuman ajankohta saatetaan antaa muodossa, jossa on päivämäärä ja kellonaika, kun tehtävän ratkaisun kannalta vain kellonajalla on merkitystä. Aineiston tietoja voidaan myös joutua yhdistämään, kuten esimerkiksi tilanteessa, jossa tehtävän ratkaisun kannalta on olennaista kuinka monta kertaa kukin käyttäjä on kirjautunut järjestelmään, mutta aineistossa kukin kirjautuminen on omana tapahtumanaan.

Datan käyttöä rajoittaa myös yksityisyyteen liittyvä lainsäädäntö. Dataa kertyy mo-

nesta paikasta, jolloin se saattaa sisältää myös yksityisyyden suojan piiriin kuuluvaa tietoa, kuten esimerkiksi tiettyyn IP-osoitteeseen liittyvät tietoliikennevirrat ja niihin liittyvien datapakettien sisältö. Tällaista arkaluontoista tietoa on käsiteltävä ja säilytettävä voimassa olevien lakien mukaan. Arkaluontoista dataa voidaan hyödyntää anonymisoimalla se poistamalla aineistosta tietoa tai sotkemalla sitä niin, että henkilöt eivät ole enää tunnistettavissa.

2.2 Mallin valinta

Tilastotieteessä George Boxin tunnetuksi tekemä sanonta "Kaikki mallit ovat vääriä, mutta jotkut ovat käyttökelpoisia." [3] kuvaa sitä, ettei malleilla koskaan pystytä täsmällisesti kuvaamaan sen kohdetta, mutta osa malleista kuvaavat kohdetta riittävän tarkasti, jotta niitä voidaan hyödyntää käytännössä.

Koneoppimismallit sisältävät kahdenlaisia parametreja, nimittäin parametreja, joita koneoppimisalgoritmit muokkaavat sekä hyperparametreja. Käyttäjä määrittelee hyperparametrit ennen algoritmin suoritusta. Näillä parametreilla mm. määritellään klustereiden lukumäärä k-lähimmän naapurin menetelmässä sekä kerrosten ja solmujen määrät neuroverkoissa.

Koneoppimisella ratkaistavat ongelmat voivat olla hyvin erilaisia eikä yksi malli voi kuvata jokaista ongelmaa kattavasti, siksi koneoppisessa käytettään paljon erilaisia malleja kuvaamaan ratkaistavaa ongelmaa. Muita käytettyjä malleja ovat esimerkiksi generatiiviset mallit, diskriminatiiviset mallit ja Gaussin mallit.

Generatiiviset mallit ovat tilastollisia malleja perustuvat yhteiseen todennäköisyysjakaumaan, kun taas diskriminatiiviset mallit taas perustuvat ehdolliseen todennäköisyyteen.

Todennäköisyyspohjaiset (probabilistic) mallit ryhmitellään parametrillisiin ja parametrittomiin [29]. Parametrillisissa malleissa on kiinteä määrä parametreja, kun taas parametrittomien mallien parametrien määrä kasvaa aineiston koon kasvaessa. Parametrilliset mallit ovat nopeampia käyttää, mutta toisaalta ne tekevät oletuksia aineistoa jakaumasta, minkä vuoksi ne eivät ole yhtä joustavia kuin parametrittomat mallit. Parametrittomat mallit aiheuttavat vaikeuksia aineistokokojen ollessa suuria.

Koneoppimisprosessin tavoitteena on opettaa malli M kuvaamaan ratkaistavaa tehtävää. Opettaminen perustuu aineistoon D, joka koostuu kokoelmasta havaintoja ilmiöstä, joka halutaan mallintaa. Oppiminen tai sovittaminen tarkoittaa mallin M parametrin θ valintaa. Parametri θ voi olla myös vektori sisältäen useampia muuttujia.

Malleja opetettaessa ja valittaessa on voitava arvioida mallin hyvyyttä. Jos tavoitteena olisi mahdollisimman hyvä tulos vain opetusaineistolla tehtävässä olisi kyse optimointiongelmasta. Kun tavoitteena on mallin yleistäminen, eli että ratkaisuna saatu malli toimii mahdollisimman hyvin tulevalla, ennen näkemättömällä aineistolla, on kyseessä koneoppimisongelma.

Koneoppimisessa ei mallin hyvyyttä arvioitaessa ei voida käyttää täysin samaa aineistoa kuin sen opettamiseen käytetään, koska tämä johtaisi mallin ylisovittumiseen. Ratkaisuna tähän ongelmaan on jakaa käytettävissä oleva opetusaineisto kahteen osaan niin, että noin 80 % siitä käytettään opettamiseen ja loput noin 20 % mallin hyvyyden arviointiin eli validointiin. Opetusaineiston ollessa pieni sen jakaminen opetus- ja validointiaineistoksi heikentää aineiston käytettävyyttä, eli opetusaineistoa ei ole tarpeeksi, jotta mallista tulisi riittävän tarkka tai validointiaineisto ei riitä mallin tarkkaan validoimiseen. Eräs ratkaisu on ristiinvalidointi, jossa aineisto jaetaan useampaan toisensa täydentävään osajoukkoon, joista yhtä osajoukkoa kerrallaan varataan validointiin, samalla kun muita osajoukkoja käytetään opettamiseen.

K-kertainen ristiinvalidointi on paljonkäytetty tapa suorittaa mallin opetus ja validointi. Menetelmässä aineisto jaetaan k:hon osaan, joista yhtä osaa vuorollaan käytettään validointiin ja muita osia aineistosta mallin opettamiseen. Opetus ja validointi suoritetaan siis k kertaa, jonka jälkeen tulokset voidaan keskiarvoistaa yksittäisen arvioin saamiseksi.

Annetusta aineistosta opetukseen käytetyllä osalla saadun mallin häviötä kutsutaan opetusvirheeksi ja vastaavasti tulevan aineiston simulointiin käytetyllä osalla saatua häviötä kutsutaan testausvirheeksi. Varsinaisella tulevalla aineistolla saatua häviötä kutsutaan yleistysvirheeksi ja koneoppimisessa on nimenomaan tavoitteena löytää mallin parametri θ, jolla tämä yleistysvirhe on mahdollisimman pieni. Tavoitteena on siis aikaansaada malli, joka laskee tarkimpia ennusteita annettuun tehtävään tulevan aineiston perusteella.

Ratkaisuna saatu malli voi olla liian yksinkertainen, jolloin kaikkia mallinnettavan ilmiön ominaisuuksia ei saada kuvattua, tätä kutsutaan alisovittumiseksi. Esimerkiksi alisovittumista tapahtuu, kun mallina on suora ja mallinnettava ilmiö on neliöllinen. Toinen ongelma on ylisovittuminen, jolloin vastaavasti malli on monimutkaisempi kuin mallinnettava ilmiö. Esimerkiksi kun neliöllistä ilmiötä mallinnetaan polynomilla, jonka astetta ei ole rajoitettu saadaan malli, joka seuraa tarkasti opetusaineiston jokaista häiriötä ja virhettä. Eräs tapa puuttua tähän ongelmaan on lisätä virhefunktioon rangaistus mallin monimutkaisuuden mukaan tätä kutsutaan regularisoinniksi. Regularisoinnin tarkoituksen on pienentää yleistysvirhettä, mutta ei opetusvirhettä, ja siten pienentää ylisovittamisen riskiä.

2.3 Ohjattu oppiminen

Ohjatussa koneoppimisessa käytettävässä opetusdatassa on syötteet ja niitä vastaavat tulokset. Menetelmällä on tavoitteena löytää malli, joka parhaiten ennustaa tulevien syötteiden tulokset. Malli opetetaan löytämään kuvaukset syötteistä niitä vastaaviin tuloksiin. Ohjatun oppimisen mallit ovat luonteeltaan ennustavia.

Ohjatun oppimisen menetelmät jaetaan regressioon ja luokitteluun, jako tapahtuu tulosmuuttujan tyypin mukaan. Syötteet ja tulokset voivat olla kategorisia tai jatkuva arvoisia. Menetelmiä, joilla on kategorisarvoiset tulokset, sanotaan luokitteluiksi.

Luokittelu voi tapahtua kahteen tai useampaan luokkaan. Kahteen kategoriaa luokittelua kutsutaan binääriluokitteluksi. Monen luokan ongelmat voidaan ratkaista paitsi suorilla ratkaisuilla myös jakamalla tehtävä useampaan binääriluokittimella ratkaistavaan ongelmaan. Tällöin verrataan joko yhtä luokkaa kaikkiin luokkiin, jolloin suurimman tuloksen saanut luokka on ratkaisu, tai vertaamalla kaikkia luokkia vuorotellen keskenään toisiinsa ja se luokka joka voittaa useimman vertailun on ratkaisu.

On myös tehtäviä, joissa tulokset voivat kuulua samanaikaisesti useampaan luokkaan, esimerkiksi jokin uutiskirjoitus voi samanaikaisesti olla sekä ulkomaan uutinen että talousuutinen. Tällöinkin luokittelu voidaan tehdä kategoria kerrallaan.

Jatkuva-arvoisia menetelmiä kutsutaan termillä regressio. Yleinen tapaus on yhden tulosmuuttujan lineaarinen regressio. Logistinen regressio on luokittelu, jossa tulosmuuttuja normalisoidaan nollan ja yhden välille [0, 1], jolloin tulos kuvaa luokan todennäköisyyttä. Jos tulosarvojen suuruudella ei ole väliä, vaan ainoastaan niiden järjestyksellä, niin menetelmää kutsutaan järjestysregressioksi.

Yleensä tulosmuuttujat ovat yksiarvoisia, mutta myös useamman tulosmuuttujan tapauksia on. Useamman tulosmuuttujan tapaukset voidaan ratkaista jakamalla ne useaksi yhden tulosmuuttujan tapaukseksi.

2.3.1 Regressio

Lineaarisessa regressiossa tulosmuuttuja on jatkuva-arvoinen. Usein myös syötemuuttujat ovat jatkuvia, mutta osa voi olla myös kategorisia. Eräs esimerkki regressiosta on asunnon hinnan ennustaminen siihen vaikuttavien muuttujien, kuten esimerkiksi koko ja sijainti, avulla. Tulosmuuttuja asunnon hinta on jatkuva-arvoinen, kuten myös syötemuuttuja asunnon koko, mutta asunnon sijainti voidaan määritellä kaupunginosan mukaan, jolloin se on kategorinen.

Yksinkertaisin menetelmä regressio toteuttamiseksi on lineaarinen regressio, jossa suora sovitetaan opetusaineistoon. Ennustus lineaarisesta regressiosta saadaan syöttämällä syötearvo opetuksesta saatuun suoran yhtälöön, josta saatu tulos on ennustettu ratkaisu. Regressio voidaan toteuttaa myös muilla menetelmillä, esim. neuroverkoilla ja päätöspuilla.

2.3.2 Luokittelu

Luokittelussa tulosmuuttuja on kategorinen, esim. ihminen, koira, auto jne. ja siinä on tavoitteena ennustaa tulevista dataobjekteista mihin luokkaan ne kuuluvat. Yksinkertaisin luokittelija on binääriluokitin, jonka perusteella dataobjektit, joko kuuluvat kohdeluokkaan tai eivät.

Yksi yleisimmistä esimerkeistä binääriluokittelusta on Naiivi Bayes-luokittimella toteutettu roskapostisuodatin, joka luokittelee saapuvat sähköpostit tarkoituksenmukaisiin sähköposteihin ja roskaposteihin, niissä esiintyvien sanojen perusteella.

Tietyt sanat esiintyvät useammin roskaposteissa kuin normaaleissa sähköposteissa. Määrittelemällä kullekin sähköposteissa esiintyvälle sanalle ehdolliset todennäköisyyden sille että se esiintyy roskapostissa ja sille, että se esiintyy normaalissa sähköpostista, voidaan Bayesin kaavan ja ketjusäännön avulla laskea todennäköisyydet sille, että sähköposti on roskaposti ja sekä sille, että se on normaali sähköposti. Näiden todennäköisyyksien avulla sähköposti voidaan luokitella joko roskapostiksi tai normaaliksi sähköpostiksi.

Dataobjekteja voidaan luokitella muutenkin kuin vain kuuluvaksi yhteen luokaan. Ajoneuvoja on erilaisia, kuten esimerkiksi polkupyöriä, moottoripyöriä, henkilöautoja ja kuorma-autoja. Ajoneuvojen luokittelu ominaisuuksien mm. painon perusteella ajoneuvoluokkiin on esimerkki moniluokka-luokittelusta (multiclass). Lehtiartikkelit voivat käsitellä useaa aihetta samanaikaisesti. Tällöin artikkeli tulee myös luokitelluksi useampaan luokkaan samanaikaisesti (multi-label).

Naiivin Bayes-luokittelijan lisäksi luokittimia voidaan toteuttaa monelle eri menetelmällä, kuten esim. logistisella regressiolla, neuroverkolla ja binäärisellä päätöspuulla. Päätöspuu on kehittynyt malli päätössäännöistä. Päätöspuissa kukin päätös kulkee puun juurisolmusta lähtien kysymys kerrallaan niin, että jokainen polku on toisensa poissulkeva.

Lineaarinen erotteluanalyysi (LDA, Linear Discriminant Analysis) on myös ohjatun oppimisen menetelmä. Sen avulla voidaan luokitella jatkuvia muuttujia. Tekniikkaa käytetään haettaessa datasta se ominaispiirteiden yhdistelmä, jonka avulla aineistossa olevat ryhmät voidaan parhaiten erottaa tosistaan.

2.4 Ohjaamaton oppiminen

Toisin kuin ohjatussa oppimisessa ohjaamattomassa oppimisessa käytettävässä datassa ei ole syötteitä vastaavia tuloksia. Ohjaamattomassa oppimisessa pyritään havaitsemaan datassa piilossa oleva tieto ja tuomaan se esiin niin, että se on helpommin ymmärrettävissä ja hyödynnettävissä. Tämä tapahtuu mm. klusteroimalla, jossa jonkin ominaisuuden perusteella samankaltaiset alkiot ryhmitellään yhteen.

2.4.1 Klusterointi

Suuresta aineistosta on joskus tarpeen löytää toisiaan muistuttavat objektit klustereiksi. Objektit klusteroidaan niiden samankaltaisuuden perusteella. Se mitä samankaltaisuudella tarkoitetaan riippuu kulloinkin annetusta tehtävästä ja siinä olevista objekteista. Yksikertaisemmillaan samankaltaisuutta voidaan mitata esimerkiksi kuinka lähellä toisiaan datapisteet sijaitsevat kaksiulotteisessa koordinaatistossa. Menetelmiä klusterointiin ovat mm. K-means, konvoluutioneuroverkko ja syvä uskomusverkko.

K-means menetelmä on tunnettu versio Gaussin sekoitusmallin (Gaussian Mixture Model, GMM) käytöstä, jossa optimointi toteutetaan odotusarvon minimoinnil-

la [29]. Menetelmässä valitaan ensin klustereiden lukumäärää kuvaavan parametrin K mukainen määrä edustajia edustamaan klustereiden keskipisteitä. Edustajat voidaan valita esimerkiksi satunnaisesti näytteistä. Tämän jälkeen muut näytteet sijoitetaan kuhunkin klusteriin sen mukaan mitä lähimpänä ne ovat valitun etäisyysmitan mukaan. Seuraavaksi lasketaan klustereille uudet keskipisteet. Tätä jatketaan niin kauan kunnes klustereiden keskipisteet eivät enää muutu, jolloin näytteet on saatu jaettua klustereihin.

2.4.2 Dimensioiden vähentäminen

Data saattaa olla korkea dimensioista, eli sen dataobjektit muodostuvat hyvin monesta ominaispiirteestä, jolloin sen käsittely on hidasta. Käsittelyn nopeuttamiseksi datan dimensioita pitäisi voida vähentää ilman, että datasta katoaa olennaista tietoa. Data saattaakin sisältää piileviä tekijöitä (Latent factors), jotka selittävät suurimman osan datassa olevasta vaihtelusta.

Yleisin menetelmä dimensioiden vähentämiseksi on Pääkomponenttianalyysi (Principal Component Analysis, PCA). Menetelmää voidaan pitää ohjaamattoman oppimisen versiona monen lähdön lineaarista regressiosta, jossa tarkastellaan monidimensioista lähtöä $\mathbf{y}$ eikä alempi dimensiosta syytä $\mathbf{z}$. Haluttuun malliin pääsemiseksi meidän on käännettävä seuraamussuhde niin, että korkea dimensioisesta datasta $\mathbf{y}$ seuraa alempi dimensioinen data $\mathbf{z}$.

Tehtävänä voi sinällään olla piilevien tekijöiden löytäminen tai sitten sitä voidaan käyttää esikäsittelynä nopeuttamaan varsinaista koneoppimistehtävää. Koska mallista on dimensioiden vähentämisen myötä poistunut epäoleellista tieto tuottavat tällaiset mataladimensioiset esitykset usein paremman ennustustuloksen [29, s. 12] kuin lähtökohtana ollut korkeampi dimensioinen esitys. Muita esimerkkejä dimensioiden vähentämisestä ovat mm. pinottu auto-enkooderi, paikallinen lineaarinen upotus ja auto-assosioiva neuroverkko.

2.4.3 Tiheysarvio

Tiheysarvio (Density Estimation) menetelmällä arvioidaan satunnaismuuttujan todennäköisyyden tiheysfunktiota. Kernelin tiheysarvio (Kernel Density Estimation, KDE) on datan silotteluongelma, jossa päätelmät populaatiosta tehdään äärellisestä datasta. KDE on ohjaamattoman oppimisen proseduuri, joka edelsi kernel regressiota. Se johtaa myös luonnollisesti yksinkertaiseen proseduurijoukkoon parametrittomia luokittelijoita.

Parametrittomia tiheysarvioita voidaan käyttää luokitteluun Bayesin lauseen avulla. Muita esimerkkejä tiheysarviosta ovat mm. Gaussian yhdistelmä, syvä Boltzmannin kone ja yleistetty kohinanpoisto auto-enkooderi.

2.4.4 Puuttuvien arvojen imputointi

Data-aineistosta saattaa joidenkin objektien kohdalla puuttua dataelementtejä tai osa dataelementeistä on virheellisiä, jolloin kyseistä objektia ei voi sellaisenaan käyttää. Jos aineisto on iso voidaan puutteelliset objektit poistaa ja saada silti riittävän tarkkoja ennusteita. Jos aineistoa on niin vähän, että objektien poistaminen huonontaisi ennusteen tarkkuutta merkittävästi, voidaan eri menetelmien avulla estimoida puuttuvia dataelementtejä. Puuttuvien arvojen imputointia hyödynnetään esikäsiteltäessä dataa jotakin toista koneoppimistehtävää varten.

Jos sarakkeet korreloivat voimme käyttää havaittuja arvoja puuttuvien arvojen ennustamiseen. Jos aineiston jakauma on tunnettu voidaan havaittujen arvojen perusteella laskea jakauman parametrit ja näytteistää jakaumasta puuttuvat arvot. Jos jakauma on tuntematon voidaan se usein approksimoida normaalijakaumalla keskeinen raja-arvolauseen (central limit theorem) perusteella.

2.5 Vahvistusoppiminen

Vahvistusoppimisessa malli muodostuu tavoitteesta, tilasta, toiminnoista ja palkkioista. Tehtävällä on tavoite eli tila, johon se pyrkii. Tehtävällä on myös nykyinen tila, joka yleensä poikkeaa tavoitetilasta. Tehtävässä suoritetaan toimintoja, joiden tarkoituksena on muuttaa nykyinen tila tavoitetilaksi. Toiminnon suorituksen jälkeen malli saa palkkion siitä kuinka hyvin toiminto auttoi kohti tavoitetta. Esimerkkinä vahvistusoppimisesta voidaan pitää järjestelmää, joka pyrkii tasapainotilaan. Järjestelmän ollessa esimerkiksi tasapainotilan alapuolella sille annetaan komento liikkua ylöspäin tietyn määrän. Järjestelmän tila siirtyy ylöspäin annetun määrän ja sitten havainnoidaan kuinka hyvin uusi tila vastaa tavoitetilaa ja sen perusteella annetaan palkkio, joka kuvaa toiminnan onnistumista.

Esimerkkinä vahvistusoppimisesta on Q-oppiminen, jossa tavoitteena on oppia menettelytapa, joka kertoo mikä toiminto on sopivin eri tilanteissa. Menettelytapaa kuvataan tila-toiminnan arvofunktiolla Q, ja tavoitteena on estimoida se maksimoimaan tulevat palkkiot. Se ei tarvitse erillistä mallia toimintaympäristöstä vaan ratkaisee ongelman palkkioiden avulla käyttäen stokastisia siirtymiä.

2.6 Syväoppiminen

Neuroverkoihin perustuva syväoppiminen on viime aikoina ollut kasvavan kiinnostuksen kohteena ja sitä sovelletaankin useilla aloilla. Neuroverkot ovatkin tällä hetkellä voimakkaimmin kehittyvä koneoppimisen ala. Neuroverkkojen rakennuspalikkoina toimivat solmut, joiden toiminta jäljittelee hermosolua, neuronia. Yksinkertaisimpana neuroverkkona voidaan pitää perseptronia, joka matkii yhden neuronin toimintaa niin, että aktivointifunktiona on askelfunktio, jolloin sen toiminta vastaa luokittelijaa. Solmut saavat viestejä eri lähteistä. Saadut viestit $\mathbf{x_i}$ solmu laskee yhteen sopivilla kertoimilla $\mathbf{v_{ij}}$ kerrottuina ja tuottaa aktivointifunktion $\mathbf{g()}$ avulla

tuloksen eteenpäin.

$$z_j = g\Big(\sum_{k=1}^{D} v_{ij} x_i\Big)$$

Monimutkaisemmat neuroverkot muodostuvan rinnakkain sijaitsevista neuroneista, jotka muodostavat kerroksen, joita on neuroverkossa voi olla useita peräkkäin. Neuroverkot muodostuvat syöte-, lähtö- sekä yhdestä tai useammasta piilokerroksesta. Alkujaan yhden piilokerroksen neuroverkkoja kutsuttiin mataliksi ja useamman piilokerroksen neuroverkkoja syviksi, mutta on näkyvissä, että neuroverkkojen koon kasvaessa myös syvän neuroverkon määritelmä muuttuu kohti useampia piilokerroksia.

Syvien neuroverkkojen tarkkuus paranee piilokerrosten lukumäärään ja leveyden leveyden kasvaessa, mutta samalla niiden opettaminen vaatii enemmän dataa ja laskentatehoa. Syynä syväoppimisen suosion kasvuun onkin paitsi kasvaneet datamäärät niin myös nopeutuneet prosessorit ja muistit. Syvemmät ja leveämmät neuroverkot mahdollistavat myös monimutkaisempien ilmiöiden oppimisen. Syväoppimisen suorituskyky paranee paremmin isommilla aineistoilla kuin perinteisillä koneoppimismenetelmillä.

Aktivointifunktiot ovat yleensä epälineaarisia, kuten Sigmoidi, tanh tai oikaistu lineaarinen yksikkö (ReLU). Jos aktivointifunktiot olisivat lineaarisia, ei kerrosten lisääminen auttaisi, vaan useampi kerros voitaisiin korvata yhdellä kerroksella, jonka kerroin olisi korvattavien kerrosten kertoimien tulo.

Lähtökerroksen aktivointifunktio valitaan tehtävän mukaan siten, että esimerkiksi regressiolle käytetään lineaarista yksikköä, binääriluokitteluun sigmoidia ja luokiteltaessa moneen luokkaan softmaxia. Sigmoidifunktio on tavallaan jatkuva versio askelfunktiosta.

$$softmax(u)_i = \frac{\exp(u_i)}{\sum_j \exp(u_j)}$$

Eräs yksinkertaisimmista neuroverkoista on monikerros perseptroni (MLP, Multilayer Perceptron), jossa neuronit muodostavat eteenpäin kytketyn verkon (feed forward network), joka on yleensä myös täysin kytketty (FCN. Fully Connected Network) [22]. Täysiin kytketty neuroverkko tarkoittaa, että jokaisen kerroksen jokaisesta neuronista on kytkentä seuraavan kerroksen jokaiseen neuroniin. Ennen kuin monikerros perseptronia voidaan opettaa pitää sen rakenne määrittää eli kerrosten lukumäärä, neuronien lukumäärät kussakin kerroksessa ja käytetyt aktivointifunktiot. Monikerros perseptronin opettaminen tarkoittaa painokertoimien määrittelyä. Jos kyseessä on ohjattu oppiminen ja meillä on käytössä näytteitä $\mathbf{x}$ vastaavat maalit $\mathbf{y}$ voimme käyttää takaisinvirtausalgoritmia (Backpropagation Algorithm).

Monikerros perseptronit ovat joustavina malleina herkkiä ylisovittumiselle, joten ne pitää regularisoida. Monikerros perseptroneja voidaan regularisoida monin tavoin, esimerkiksi varhaisella pysähtymisellä, painokertoimen heikkeneminen (weight

decay), painokertoimien jaolla, lisäämällä aineistoon kohinaa tai nollaamalla piilokerroksista neuroneja jollain määritellyllä todennäköisyydellä. Varhaisessa pysähdyksessä opetus keskeytetään, kun testausvirhe ei enää pienene.

Yhdistämällä syötteet pienillä kaksiulotteisilla alueilla saadaan konvoluutioneuroverkot (CNN, Convolutional Neural Networks), jotka ovat omiaan kuvien käsittelyssä, sillä kuvissa vierekkäiset pikselit liittyvät usein toisiinsa [15]. Konvoluutiot tapahtuu omissa piilokerroksissa yleensä verkon alkupäässä. Näitä kerroksia kutsutaan konvoluutiokerroksiksi. Konvoluutiokerroksia käytettään myös datamäärän pienentämiseksi. Yksittäisten piirteiden opettaminen on laskennallisesti kallista, tämä voidaan kiertää satunnaisilla ominaiskertoimilla tai käyttämällä esiopetettuja kerroksia, joita löytyy Internetistä.

Lisäämällä takaisinkytkentäsilmukoita saamme toistavat neuroverkot (RNN, Recurrent Neural Networks), joiden vahvuus on ajallisten muutosten oppimisessa, kuten puheentunnistamisessa, konekääntämisessä sekä yhdessä konvoluutioneuroverkkojen kanssa kuvien kaappauksessa [15]. Toistavat neuroverkot voivat myös tuottaa peräkäisdataa, kuten esimerkiksi tekstiä. Toistavia neuroverkkoja sovelletaan myös vahvistusoppimiseen.

Autoenkooderit ovat neuroverkkoja, joissa haluttu lähtö on sama kuin syöte. Menetelmän ideana on ahtaa informaation syötteisiin nähden kapeiden piilokerrosten läpi, jolloin se tiivistyy. Eräs tapa käyttää autoenkoodereita on kohinanpoisto.

Generatiiviset kilpailevat verkot (GAN, Generative Adversarial Network) on mielenkiintoinen tapa hyödyntää syviä neuroverkkoja esimerkiksi aitojen kuvien tuottamiseen. GAN perustuu kahteen neuroverkkoon, joista toinen kehittää esimerkiksi kuvan ja toinen kilpaileva neuroverkko luokittelee kuvat aitoihin ja generoituihin, näin saadaan aidommilta näyttäviä kuvia.

Ohjatut syvät verkot oppivat usein kiinnostavia sisäisiä esityksiä piilokerroksille. Näitä kutsutaan hajautetuiksi esityksiksi (Distributed Representations).

2.7 Yhteenveto koneoppimisesta

Datamäärien kasvaessa on vaikeaa löytää kaikkea siinä piilevää tietoa ilman automaattisia ja laskennallisesti tehokkaista menetelmiä. Koneoppiminen tekoälyn osa, joka tavoitteena on datasta oppiminen. Koneoppimisen perusta on tilastotieteessä, joten sen mallit ja terminologia ovat tilastotieteestä lähtöisin.

Koneoppiminen jaetaan ohjattuun oppimiseen, ohjaamattomaan oppimiseen ja vahvistusoppimiseen. Ohjatussa oppimisessa käytetään aineistoa, jossa yksittäiset dataobjektit sisältävät sekä syötetiedot, että niitä vastaavat tulostiedot. Ohjatun oppimisen menetelmiä ovat mm. luokittelu ja regressio. Luokittelussa uusi syöte sijoitetaan aineiston perusteella opetetun mallin avulla oikeaan luokkaan. Regressiossa aineiston perusteella tehdyn mallin avulla uuden syötteen perusteella ennustetaan tulosteen arvo.

Ohjaamattomassa oppimisessa dataobjekteissa on syöttötiedot, mutta ei ole tulos-

tietoa tai sitä ei hyödynnetä. Ohjaamattoman oppimisen menetelmiä ovat mm. klusterointi, dimensioiden vähentäminen, tiheysarvio ja puuttuvien arvojen imputointi. Klusterointi on näistä menetelmistä merkittävin, siinä aineistossa olevat dataobjektit ryhmitellään sen mukaan kuinka paljon ne jollakin mittarilla muistuttavat toisiaan. Osassa klusterointimenetelmiä klustereiden lukumäärä on eräs mallin ennalta määritelty hyperparametri, kun taas osa menetelmistä tekee aineistosta päätelmiä siinä olevien klustereiden lukumäärästä.

Vahvistusoppiminen on menetelmä, jossa järjestelmän tilaa pyritään muuttamaan eri toiminnoilla kohti tavoitetilaa. Järjestelmä saa palautetta siitä kuinka hyvin kukin toiminto auttaa saavuttamaan tavoitteen, jolloin se osaa valita seuraavan toiminnon paremmin.

Neuroverkot ja varsinkin syvät neuroverkot ovat koneoppisen tällä hetkellä suuren kiinnostuksen alaisia kehityskohteita. Tämä johtuu datamäärien kasvamisesta ja prosessorien suorituskyvyn kasvamisesta, jotka ovat mahdollistaneet monimutkaiset syvät neuroverkot. Syvillä neuroverkoilla on mahdollista saada tarkkoja tuloksia tai havainnoita monimutkaisia ilmiöitä.

Neuroverkkojen perusrakenne on hermosolua jäljittelevä neuroni. Neuroverkot muodostuvat useaan eri kerrokseen sijoitetuista neuroneista. Jokaisessa neuroverkossa on syötekerros, yksi tai useampi piilokerros ja tulostekerros.

Neuroverkoissa voi olla eri määrä kerroksia ja eri kerroksissa voi olla eri määrä neuroneita. Paljon neuroneita sisältävät neuroverkot ovat hitaita opettaa ja vaativat paljon dataa, siksi onkin tärkeää määritellä tehtävään sopiva riittävän pieni neuroverkko. Yksinkertaisin neuroverkko on täysin kytketty vain eteenpäin kytketty monikerros perseptroni, mutta neuroverkkossa voi olla myös takaisinkytkentää, jolloin saadaan paremmin mallinnettua toisistaan riippuvat peräkkäiset tapahtumat.

3 Tietoverkot

Nykyaikainen yhdyskunta tarvitsee tietoa kyetäkseen toimimaan tehokkaasti. Tietoa syntyy monin tavoin, monissa paikoissa ja sitä pitää pystyä nopeasti siirtämään sinne missä sitä tarvitaan. Internet mahdollistaa tehokkaan, nopean ja edullisen tavan siirtää tietoa.

Internet hyödyntää kerrosmalli, jossa kerrokset hyödyntävät alemman kerroksen palveluita ja tarjoaa vastaavasti palveluita ylemmälle kerrokselle. Internetissä alin peruskerros tarjoaa fyysisen siirtotien. Seuraavana oleva verkkokerros hoitaa datapakettien reitityksen käyttäen peruskerrosta. Kuljetuskerros vastaa datapakettien siirron päätelaitteiden välillä käyttäen verkkokerroksen palveluita. Ylimpänä pinossa on sovelluskerros, joka tiedonsiirtotarpeittensa mukaan käyttää kuljetuskerroksen palveluita.

Internet koostuu kokoelmasta protokollia, joista merkittävimmät ovat reititykseen liittyvä IP (Internet Protocol) ja pakettien kuljettamiseen liittyvät UDP (User Da-

tagram Protocol) ja TCP (Transmission Control Protocol). UDP siirtää paketteja päätteeltä toiselle ilman virheenkorjausta tai varmistuksia tiedonsiirron onnistumisesta. TCP puolestaan sisältää virheenkorjauksen ja ruuhkanhallinnan.

Tietoverkot ovat monimuotoisia laitteita, jotka koostuvat useista pienemmistä laitteista. Tietoverkoissa paitsi liikkuu dataa niin niissä myös syntyy dataa. Tämän datan analysoiminen on tärkeää tietoverkkojen toiminnan tehostamiseksi. Koska dataa on paljon, on mahdollista hyödyntää datan analysoimiseen koneoppimisen menetelmiä. Koneoppimista voidaan hyödyntää paitsi yksittäisen verkkolaitteen toiminnan analysointiin, myös koko tietoverkon toiminnan analysointiin.

Koneoppimisen hyödyntäminen edellyttää, että meillä on käytettävissä dataa, joka kuvaa ratkaistavaa ongelmaa. Tietoverkot sisältävät erilaisia tehtäviä, joissa voidaan hyödyntää koneoppimista, kuten liikenteen luokittelu, poikkeamien havaitseminen sekä siihen liittyvä tietoverkon turvallisuus, resurssien hallinta, reititys ja suorituskyvyn analysointi.

4 Liikenteen luokittelu

Tietoverkossa liikkuu monimuotoista tietoa, jolla on kullekin sovellukselle ominaiset piirteet. Esimerkiksi tekstipohjainen etäkäyttö SSH-yhteydellä saattaa siirtää vain muutaman merkin kerrallaan, kun taas tiedostojen siirtoon tarkoitettu FTP-yhteys siirtää useamman megatavun useassa perättäisessä paketissa. Kullakin sovelluksella on tietoliikenneyhteydelle asetetut vaatimukset, esimerkiksi pankkidatan siirto edellyttää virheettömyyttä ja videokeskustelu reaaliaikaisuutta. Jotta tietoverkko kykenisi mahdollisimman tehokkaasti hyödyntämään olemassa olevia resursseja on tarpeen luokitella siirrettävä tieto sen sovelluksen tarvitsemien ominaisuuksien tukemiseksi.

4.1 Taustaa

Aikaisemmin kukin palvelu sijaitsi omassa portissaan, jolloin liikenne oli helppo luokitella porttien perusteella. Monissa paikoin vertaisverkot, kuten torrentit, aiheuttivat kuormitusta ja sen käyttöä rajoitettiin. Rajoitus tehtiin estämällä vertaisverkkojen käyttämien porttien käyttö. Vastauksena tähän vertaisverkot alkoivat määrittää käyttämänsä portit dynaamisesti sekä käyttämällä muiden sovellusten yleisesti käyttämiä portteja. Tämä johti siihen, että liikenteen luokittelu porttitietoon perustuen ei ollut enää tarkkaa.

Eräänä ratkaisuna liikenteen luokitteluun on kehitetty hyötykuormaan perustuva menetelmä. Monien sovelluksien tietoliikenteellä on omat ominaispiirteensä, jotka näkyvät myös niiden hyötykuormassa. Analysoimalla eri sovellusten lähettämien pakettien hyötykuormia voidaan määritellä tietyt ominaispiirteet, joiden perusteella liikenne kyetään luokittelemaan. Vertaisverkoihin on kehitetty keinoja hyötykuormaan perustuvan luokittelun välttämiseksi. Vertaisverkot lisäävät pakettien loppuun vaih-

televan määrän tyhjää tietoa täytteeksi, tekeytyvät toisiksi sovelluksiksi tai salaavat sisällön [14]. Myös yksityisyyteen liittyvät lait saatavat rajoittaa pakettien hyötykuormien analysointia. Pakettien analysointiin tarvitaan paljon laskentakapasiteettia.

Yhtenä ratkaisuna on hyödyntää kuljetuskerroksen statistiikkaa liikenteen luokitteluun. Aivan kuten sovellusten hyötykuormissa on eroja niin on myös siinä miten ne käyttäytyvät kommunikoidessaan tietoverkossa. Analysoimalla sovellusten käyttäytymistä tilastojen avulla voidaan saatuja malleja hyödyntää sovellusten liikenteen luokittelussa. Tietoliikenneyhteydestä voidaan tilastoida mm. liikennevirrankesto, pakettien pituuden vaihtelu, pienin ja suurin segmentin koko, aikaikkunan koko, vasteaika, jne. Kerättyjä tilastoja voidaan hyödyntää automaattisesti rakentamalla koneoppimisen menetelmillä luokittelija. Tällä tavoin rakennettu luokittelija tarvitsee vähemmän laskentaresursseja sekä kykenee tunnistamaan salatun liikenteen.

4.2 Menetelmät

Liikenteen luokitteluun koneoppimisen avulla voidaan käyttää sekä ohjatun, että ohjaamattoman oppimisen menetelmiä. Luokittelussa on tavoitteena löytää funktio kuvaamaan tulevat liikennenäytteet määriteltyihin kategorioihin. Ohjaamatonta oppimista voidaan myös hyödyntää etsittäessä olemassa olevien kategorioiden lukumäärää.

Ohjatun oppimisen menetelmistä on liikenteen luokitteluun sovellettu mm. tukivektorikoneita (Support Vector Machines - SVM) [16]. Olennainen osa tukivektorikonetta on kernelifunktio. Sen avulla monimutkainen moniulotteinen operaatio voidaan välttää ja suorittaa laskenta alempi ulotteisessa hyperavaruudessa.

Kernelifunktio on reaaliarvoinen funktio, joka saa argumenteiksi kaksi objektia vektoreina [29]. Funktion arvoin voidaan ajatella kuvaavan kuinka samankaltaisia objektit ovat. Kernelifunktion valinnalla on suuri vaikutus luokittelijan suorituskykyyn. Z. Fan and R. Liu [16] ovat vertailleet neljää yleisesti käytettyä kernelifunktiota:

- Lineaarinen
- Polynominen
- Radial basis
- Sigmoid

Lineaarinen kernelifunktio on käyttökelpoinen, kun ratkaisun päätösraja voidaan todennäköisesti esittää lineaarisella yhdistelmällä alkuperäisiä ominaispiirteitä.

$$\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'.$$

Polynominen kernelifunktio

$$\kappa(\mathbf{x}, \mathbf{x}') = (\gamma \mathbf{x}^T \mathbf{x}' + r)^M,$$

missä r>0 ja M on polynomin aste, sopii ongelmiin, joissa päätösraja on monimutkaisempi kuin suora.

Sigmoid kernelifunktio

$$\kappa(\mathbf{x}, \mathbf{x}') = tanh(\gamma \mathbf{x}^T \mathbf{x}' + r).$$

Sekä polynominen että sigmoid kernelifunktio ovat ns. Mercer kerneleitä. Mercer kernelin Gram matriisi on definiitti positiivinen, jolloin kernelifunktioden laskenta yksinkertaistuu.

Radial basis kernelifunktio on vain termin $||\mathbf{x} - \mathbf{x}'||$ funktio, kuten esimerkiksi

$$\kappa(\mathbf{x}, \mathbf{x}') = exp\Big(- \frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2} \Big),$$

jossa termiä σ^2 kutsutaan kaistanleveydeksi.

Rangaistuskertoimen C avulla määritetään hyväksymis- ja sietotasot väärinluokittelulle. Suurempi rangaistuskerroin sietää huonommin virheitä ja kaventaa marginaalia. C valitaan yleensä ristiinvalidoinnin avulla, mutta sen riippuvuus kernelifunktion parametreista vaikeuttaa tehtävää.

Ohjaamattoman oppimisen menetelmistä on liikenteen luokitteluun sovellettu mm. K-means, joka on partitiointiin perustuva menetelmä kohdejoukon klusterointiin [14]. Partitiointiin perustuvia menetelmiä on useita, mutta K-means on yksi nopeimpia ja yksinkertainen toteuttaa. Menetelmälle annetaan parametri K, joka kertoo kuinka moneen klusteriin kohdejoukko jaetaan.

Kohteet jaetaan klustereihin niiden samankaltaisuuden perusteella. Tarkoituksena on, että klusterin sisällä olevat kohteet ovat mahdollisimman samankaltaisia ja vastaavasti eri klustereissa olevien kohteiden erot ovat mahdollisimman suuret. Samankaltaisuus määritellään kohteiden ominaispiirteiden välisen etäisyyden avulla. Etäisyyttä voidaan mitata monella tavalla, joista yleisin on euklidinen etäisyys.

Samankaltaisuutta mitataan yleisimmin jollain seuraavista neljästä mitasta:

- Manhattan etäisyys
- Minkowski etäisyys
- Canberra etäisyys
- Euklidinen etäisyys

Yleisin näistä on Euklidinen etäisyys, joka lasketaan

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} \Big(p_i - q_i\Big)^2},$$

jossa $\mathbf{p}$ ja $\mathbf{q}$ ovat ominaispiirteiden lukumäärän pituisia vektoreita ja p_i ja q_i niiden alkioita. Euklidinen etäisyyden tulkinta on pisteiden välisen suoran pituus.

Manhattan etäisyys puolestaan lasketaan

$$d(\mathbf{p}, \mathbf{q}) = ||\mathbf{p} - \mathbf{q}||_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

ja sen voidaan ajatella kuvaavan liikkumista kaupungissa kohtisuorassa toisiaan kohtaan olevia katuja pitkin.

Minkowski etäisyyden voidaan ajatella olevan yleistys sekä Euklidisesta että Manhattan etäisyydestä. Ja se lasketaan

$$d(\mathbf{p}, \mathbf{q}) = \Big(\sum_{i=1}^{n} |q_i - p_i|^p\Big)^{1/p},$$

joka, jos p on yksi vastaa Manhattan etäisyyttä ja jos p on 2 vastaa Euklidista etäisyyttä. Yleensä p on yhden ja kahden välillä.

Canberra etäisyys on painotettu versio Manhattan etäisyydestä ja se lasketaan

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}.$$

Käytettävä etäisyysmitta valitaan sovelluksen tarpeiden mukaan.

K-means pyrkii maksimoimaan samankaltaisuuden klustereiden sisällä minimoimalla neliövirheen (Square-error, SE)

$$SE = \sum_{i=1}^{K} \sum_{j=1}^{n} |d(x_j, c_i)|^2,$$

jossa x_j on luokiteltava kohde ja c_i on klusterin keskus.

DBSCAN (Density Based Spatial Clustering of Applications with Noise) on kohteiden esiintymistiheyteen perustuva menetelmä [14]. Menetelmä määrittää klusteriksi alueen, jossa on tiheästi kohteita ja näitä alueita erottavat harvemmat alueet. Menetelmä sallii monimuotoiset klusterit eikä rajoita niitä pallomaisiksi, kuten esimerkiksi tukivektorikoneet.

Menetelmällä on kaksi parametria: epsilon ja minPts. Epsilon määrittää etäisyyden kohteen ympärillä, joka on epsilon-naapurusto. MinPts määrittää kohteiden vähimmäismäärän epsilon-naapurustossa, jonka jälkeen kohde määritellään ydinkohteeksi.

Menetelmä perustuu kahteen määritelmään: tiheys-saavutettavuus (density-reachability) ja tiheys-yhdistettävyys (density-connectivity). Jos kohteen q epsilon-naapurustossa on vähintään minPts lukumäärä kohteita sanotaan q:ta ydinkohteeksi. Kaikkia epsilon-naapurustossa olevien kohteiden sanotaan olevan suoraan tiheys-saavutettavissa q:sta. Kohde p on tiheys-saavutettavissa q:sta, jos se on sellaisen kohteen epsilon-naapurustossa, joka on suoraan tiheys-saavutettavissa tai tiheys-saavutettavissa q:sta. p ja q ovat tiheys-kytketyt, jos on olemassa kohde o, joka on tiheys-saavutettavissa sekä p:stä että q:sta.

Klusteri on määritelmän mukaan se joukko kohteita aineistosta, jotka ovat tiheys-kytkettyjä tiettyyn ydinkohteeseen. Tosin kuin monissa muissa menetelmissä, kuten tukivektorikone tai K-means, kaikkia kohteita ei välttämättä luokitella kuuluvaksi johonkin klusteriin. Luokittelemattomat kohteet, jotka eivät siis kuulu mihinkään klusteriin, luokitellaan lopuksi kohinaksi.

AutoClass klusterointimenetelmä perustuu todennäköisyysmalliin [14]. Toisin kuin monet muut klusterointi menetelmät AutoClass osaa automaattisesti valita klustereiden lukumäärän. Lisäksi AutoClass tekee ns. pehmeän klusteroinnin eli jakaa kohteen osittain useampaan klusteriin.

Todennäköisyysmallin rakentamisen yhteydessä algoritmi määrittelee klustereiden lukumäärän ja parametrit, jotka ohjaavat kunkin klusterin ominaista todennäköisyysjakaumaa. AutoClass perustuu kahteen askeleeseen: Odotusarvoaskel (Expectation) ja Maksimointiaskel.

Odotusarvoaskeleessa menetelmä käyttää näennäissatunnaisia lukuja käytettäville parametreille. Maksimointiaskeleessa keskiarvon ja varianssin avulla uudelleen arvioidaan parametreja, kunnes ne suppenevat paikalliseen maksimiin. Nämä paikalliset maksimit tallennetaan ja prosessi suoritetaan uudelleen, kunnes riittävä määrä parametreja on tallessa. Tämän jälkeen käytettään Bayes-pistemäärää (Bayesian score) määrittelemään parhaan parametrijoukon käytettäväksi todennäköisyysmalliin. Bayes-pistemäärää rankaisee malleja, joissa on enemmän klustereita ja täten minimoi mahdollista ylisovittamista. AutoClass vaatii varsin paljon laskentaresursseja, esimerkiksi Erman et al. [14] vertailivat K-means, DBSCAN ja AutoClass menetelmiä ja havaitsivat, että saman aineiston läpikäyminen vei K-means:ltä minuutin, DBSCAN:lta kolme minuuttia ja AutoClass:lta neljä ja puoli tuntia.

Klustereiden lukumäärä vaikuttaa laskentaan käytettävään aikaan, jolloin on laskentakapasiteetin kannalta edullista vähentää klustereiden lukumäärää. Jos aineistosta suurin osa kohteista on vain pienessä joukossa klustereita voidaan keskittyä luokittelemaan vain nämä klusterit [14].

4.3 Vertailu

Eri koneoppimismetodien kykyä luokitella liikennettä Fan et al. [16] arvioi kolmella mittarilla: tarkkuus (precision), herkkyys (recall) ja täsmällisyys (accuracy). Tarkkuus on oikeiden positiivisten luokittelujen osuus kohteista, jotka on ennustettu po-

sitiivisiksi. Herkkyys on oikeiden positiivisten luokittelujen osuus kohteista, jotka ovat oikeasti positiivisia. Täsmällisyys on oikein luokiteltujen osuus kaikista näytteistä. Tämän lisäksi määritellään tasapainotettu F1-mitta

$$F1 = \frac{2 \cdot tarkkuus \cdot herkkyys}{tarkkuus + herkkyys},$$

jota käytetään arvioimaan kompromissia tarkkuuden ja muistamisen väliltä.

Tukivektorikoneen säätämiseksi mahdollisimman tarkaksi Fan et al. [16] vertasivat eri kernelifunktioiden vaikutusta tarkkuuteen. Laskennan tarkkuuden lisäksi on olennaista huomioida laskennan vaativuus, jota kuvataan tarvittavien tukivektoreiden lukumäärällä. Vertailussa (Taulukko 1) havaittiin, että lineaarinen kernelifunktio vaati laskennallisesti vähiten, mutta sen kanssa saavutettu täsmällisyys oli vain 78 %, kun taas lähes yhtä vähällä laskennalla selvisi radial basis kernelifunktio, jota käyttämällä saavutettiin lähes 98 % täsmällisyys.

Taulukko 1: Kernelifunktioiden vaikutus tarkkuuteen.

	Täsmällisyys	Tukivektorien lukumäärä
Radial basis	98%	1768
Polynomiaalinen	91%	2008
Sigmoid	88%	2550
Lineaarinen	78%	1733

Mallin virityksen yhteydessä kokeiltiin myös rangaistuskertoimen vaikutusta tarkkuuteen ja käytetyllä aineistolla saatiin paras tulos arvolla 50, jolloin täsmällisyys oli 98,8%, kun kertoimen arvolla yksi se oli 98%.

Jeffrey Erman et al. [14] on vertaillut K-means, DBSCAN ja Autoclass menetelmiä liikenteen luokittelussa ja käyttänyt tehokkuuden mittaamiseen kokonaistarkkuutta, joka saadaan

$$kokonaistarkkuus = \frac{\sum_{i=1}^{n} Oikeat\ positiiviset\ kaikista\ klustereista}{yhteyksien\ kokonaismaara}.$$

Kokonaistarkkuus määrittää kuinka hyvin klusterointi on onnistunut luomaan klustereita, joissa on vain yhtä liikenneluokkaa. Klusterien luokka määräytyy siinä olevien objektien enemmistön luokkien perusteella. Yhteyksiä, jotka on luokiteltu oikein kutsutaan oikeiksi positiivisiksi, vastaavasti väärin luokiteltuja yhteyksiä kutsutaan vääriksi positiivisiksi ja luokittelemattomia yhteyksiä kutsutaan kohinaksi.

K-means menetelmälle annetaan klustereiden määrä K lähtöarvona. Lähtökohtana voidaan pitää, että eri protokollat, kuten HTTP, FTP jne. luokitellaan omiin kategorioihin, mutta tämän lisäksi kullakin protokollalla voi olla sisäistä hajautumista eri luokkiin, kuten esimerkiksi HTTP liikenne voi sisältää selailua, tiedoston latausta tai vaikka videoiden suoratoistoa. Tutkimuksessa oli testattu klustereiden määrän

vaikutusta luokittelun tarkkuuteen kasvattamalla K:n arvoa kymmenestä ylöspäin kymmenen välein 500:aan asti.

Menetelmiä testattiin kahdella eri kokeellisella aineistolla, jotka on saatu Aucklandin yliopistosta ja Calgaryn yliopistosta. K-means menetelmällä kokonaistarkkuudet ovat aluksi 49% Auckland IV -aineistolla ja 67% Calgary -aineistolla. Klustereiden määrän K kasvaessa myös kokonaistarkkuudet kasvavat niin, että K:n ollessa 100 kokonaistarkkuudet ovat 79% ja 84% vastaavasti. Tästä eteenpäin kokonaistarkkuuden kasvu edelleen hidastuu klustereiden määrän kasvaessa niin, että K:n ollessa 500 kokonaistarkkuudet jäävät alle 90%. Isoilla klustereiden määrillä kyseessä on ylisovittamisesta, jossa aineistoa jaetaan useampaan klusteriin, kuin niitä todellisuudessa aineistossa on. Tutkimuksessa ei otettu kantaa mitkä ovat aineistojen oikeat liikenneluokkien määrät tai miten se voidaan selvittää K-means menetelmää käytettäessä, jolloin ei voida arvioida menetelmän tarkkuutta täsmällisemmin.

DBSCAN menetelmälle annetaan kaksi parametria minPts ja eps, joiden vaikutusta kokonaistarkkuuteen on tutkittu. MinPts parametria on vaihdeltu 3 ja 24 välillä ja eps etäisyyttä 0,005:an ja 0,040:an välillä. MinPts:n ollessa 3 saatiin parempia tuloksia kuin sen ollessa 24, koska klustereiden koko on pienempi, tyypillisesti kolmesta viiteen yhteyttä. Pitämällä minPts kolmessa ja kasvattamalla eps etäisyyttä 0,005:sta 0,020:aan DBSCAN menetelmän kokonaistarkkuus paranee 59,5%:sta 75,6%:iin Auckland IV -aineistolla ja Calgary -aineistolla 32,0%:sta 72,0%:iin. Kasvattamalla eps etäisyyttä edelleen kokonaistarkkuus alkaa merkittävästi pienenemään. Tutkijat arvioivat, että tämä johtuu eri liikenneluokkien yhdistymisestä yhdeksi isommaksi klusteriksi. Klusteri koostui yhteyksistä, jotka koostuivat muutamista paketeista, muutamista siirretyistä tavuista ja lyhyistä kestoista. Tämä sisältö muodostui SMTP yhteyksistä, joissa vastaanottaja hylätään ja yhteys suljetaan, POP3 yhteyksissä, joissa postilaatikossa ei ole uusia sähköposteja tai Gnutella -asiakkaan yhteydenotto yritykset hylätään.

Autoclass:ia käyttäen saatiin keskimääräisiksi kokonaistarkkuuksiksi Auckland IV -aineistolla 92,4% ja Calgary -aineistolla 88,7%. Toisin kuin K-means Autoclass kykenee määrittämään parhaan klustereiden määrän, joka Auckland IV -aineistolla on 167 klusteria ja Calgary -aineistolla 247 klusteria.

Liikenteen luokittelussa syntyvien klustereiden määrä on huomionarvoinen asia, koska kullekin klusterille on löydyttävä sitä kuvaava nimi. Samalla on huomioitava, että pienempi klustereiden määrä nopeuttaa laskentaa. Eräs tapa vähentää klustereiden määrää on tutkia kuinka monta yhteyttä niissä on. Jos vaikka klusteroinnin tuloksena ollaan saatu tarkka klusterointi, jossa suurin osa yhteyksistä on pienessä osajoukossa klustereita, voidaan analysoida vain tämä osajoukko ja nimetä nämä klusterit.

Tutkijat havaitsivat, että DBSCAN luokittelee yhteydet herkemmin vähempään määrään klustereita kuin K-means ja Autoclass. Esimerkiksi ajettaessa Auckland IV -aineistolla noin 200 klusterin parametreilla K-means ja Autoclass luokittelee 15 klusteriin 50% yhteyksistä, kun taas DBSCAN luokittelee 5 klusteriin 50% yhteyksistä. Nämä viisi luokkaa tunnisti 75,4% NNTP, POP3, SOCKS, DNS ja IRC yhteyk-

sistä kokonaistarkkuudella 97,6%. Tämä on huomattavaa, kun käytetään vain viittä noin 200:sta klusterista. Samankaltaiset tulokset saatiin myös Calgary -aineistolla.

Tutkituista kolmesta menetelmästä vain DBSCAN voi jättää osan yhteyksistä luokittelematta ts. luokittelee ne kohinaksi. Tämä hieman heikentää DBSCAN menetelmän kokonaistarkkuutta, joka onkin heikompi kuin K-meansin tai Autoclassin. Toisaalta tutkijat havaitsivat, että DBSCAN kykenee tuottamaan erittäin tarkkoja klustereita.

Autoclass oli tarkin, mutta sen laskenta-ajat ovat paljon pidemmät kuin muilla tutkituilla menetelmillä, joten se ei ole kovin käyttökelpoinen. Sen sijaan K-means saavuttaa lähes saman tarkkuuden paljon pienemmällä laskenta-ajalla, joka tekee siitä erittäin käyttökelpoisen. DBSCANin vahvuus on siinä, että se luokittelee valtaosan yhteyksistä pieneen joukkoon klustereita.

5 Suorituskyvyn analysointi

Tietoverkon suorituskyky kuvaa minkälaista siirtokapasiteettia se tarjoaa käyttäjälle. Tietoverkoissa liikkuu dataa eri muodoissa ja eri tarkoituksiin. Erilaisilla tehtävillä on erilaiset vaatimukset verkolle ja sen suorituskyvylle. Suuren datavirran vaativat reaaliaikaiset videopuhelut vaativat suurta kaistanleveyttä sekä lyhyitä kulkuaikaviipeitä ja sallivat jonkin verran virheitä sen vaikuttamatta koettuun palvelun laatuun. Raakadatan, kuten tietokantojen, siirto ei toisaalta salli lainkaan virheitä, mutta siirto voidaan tehdä pienemmällä kaistanleveydellä löysillä kulkuaikaviivevaatimuksilla. Käyttämällä reitin varrella virheenkorjausta voidaan osasta virheistä päästä eroon, mutta olennaista on, että data on perillä saapuessaan täysin vastaava kuin lähtiessään.

5.1 Taustaa

Tietomäärien kasvaessa on verkon kaikissa olosuhteissa pystyttävä tarjoamaan asiakkailleen riittävän nopeat ja luotettavat tiedonsiirtopalvelut. Kustannusten säästämiseksi tiedonsiirtokapasiteetti pyritään mitoittamaan tarpeiden mukaan. Jotta tietoverkoista saataisiin kaikissa olosuhteissa paras mahdollinen siirtokapasiteetti, mittaavat verkot suorituskykyään ja sen muutoksia sekä optimoivat sitä.

5.2 Menetelmät

Tietoverkossa esiintyy pitkäkestoisten vikojen lisäksi yksittäisiä lyhytkestoisia häiriöitä. Nämä häiriöt ovat yleisiä, eikä niiden syitä tai käyttäytymistä täysin tunneta. Osa näistä häiriöistä ei ole toistuvia, jolloin niihin on hankala puuttua. Osa häiriöistä taas on toistuvia, kroonisia. Krooniset häiriöt eivät vain hetkellisesti heikennä palvelun laatua vaan voivat indikoida verkkolaitteiden tai linkkien hajoamista. Häiriöiden ollessa toistuvia ne voidaan tunnistaa ja siten löytää niihin syy.

R. Sasisekharan et al. [33] on tutkinut koneoppimisen hyödyntämistä häiriöiden luokitteluun kroonisiksi, jolloin häiriöiden aiheuttajat on helpompi löytää ja niihin voidaan puuttua ajoissa ennen kuin häiriön synnyttävä laite hajoaa ja aiheuttaa verkkovian. Kroonisten häiriöiden havainnointi perustuu suureen määrään historiallista dataa verkon toiminnasta sekä monimutkaisten käyttäytymismallien analysointiin.

Kehitettävän menetelmän tavoitteena on kyetä ennustamaan jatkaako nykyinen häiriö toistuvana. Häiriöt ovat usein lyhytkestoisia, jolloin on helpompi mitata aikajaksojen häiriötrendejä. Häiriöitä mitataan aikaikkunassa, joka koostuu useasta aikayksikköä, ja jonka perusteella pyritään ennustamaan häiriöiden esiintyminen toisessa aikaikkunassa. Mittausaineisto koostuu ajallisista yhteenvedoista verkon toiminnasta. Näitä yhteenvetoja talletetaan aina häiriön ilmaantuessa.

Mittausdata koostuu 30 ominaispiirteestä, joista osa liittyy häiriöiden esiintymiseen ja osa verkon ominaisuuksiin. Häiriöihin liittyviä ominaispiirteitä ovat häiriön esiintymiskertojen lukumäärä, häiriöiden keskimääräinen esiintyminen aikayksikössä ja aikayksikköjen lukumäärä, joissa häiriö esiintyi. Verkkoa koskevia ominaispiirteitä on mm. sijainti, reitin tyyppi ja verkon rakenne.

Tutkimuksessa vertailtiin erilaisia menetelmiä, kuten lineaarinen diskriminanttianalyysi (LDA), neuroverkko, päätöspuu, päätössäännöt ja lähin naapuri. Ensin pienemmällä datalla pyrittiin selvittämään olisiko joku menetelmä ylivertainen muihin verrattuna. Tässä yhteydessä huomattiin, että LDA ja lähin naapuri toimivat huonosti. Neuroverkko ja päätöspuu toimivat varsin kilpailukykyisesti kuitenkin niin, että päätöspuu oli hieman parempi. Lopullisessa vertailussa tulokset ovat samankaltaiset kuin alustavalla, pienemmällä datalla tehdyllä vertailulla.

Tutkimuksessa vertailukohtana on luokittelija, joka yksinkertaisesti luokittelee kaikki isoimpaan luokkaan, jolloin virheellisesti luokiteltuja on 6,4%. LDA:n tulos ei ole yhtään parempi. Heikko suoriutuminen tässä luokittelussa on tyypillistä tälle menetelmälle tilanteissa, joissa luokan alkioiden esiintyminen on alhainen. Lähin naapuri -menetelmä antaa lähes prosenttiyksikön verran paremman tuloksen, muttei pärjää neuroverkolle, päätöspuulle tai päätössäännöille (taulukko 2).

Tutkimuksen tuloksena saatu menetelmä onnistui tuottamaan krooniset häiriöt ennustavan sääntöjoukon. Sääntöjoukko voidaan pelkistää joukoksi olosuhteita, joiden toteutuessa esiintyvä häiriö on todennäköisesti krooninen.

Taulukko 2: Luokittelu krooniseksi verkkohäiriöksi

Menetelmä	Virheosuus %
Priori	6.4
LDA	6.4
Lähin naapuri	5.6
Neuroverkko	4.7
Päätöspuu	4.6
Päätössäännöt	4.5

Eräs ongelma koneoppisen hyödyntämisessä on löytää sopiva menetelmä ratkaistavaan ongelmaan. Casas et al. on kehittänyt yleisen koneoppimismallin (Generic Machine Learning, GML) [7], joka perustuu kokoelmaoppimiseen (ensemble learning) ja on tarkoitettu käytettäviksi moniin tietoverkon mittausongelmiin. GML:n idea perustuu Superoppijaan (Super-Learner [27]), käyttäen kokoelmaoppimisen pinoamiseen(Stacking) perustuvaa lähestymistapaa.

Pinoamisen ideana on hyödyntää useampaa yksinkertaista koneoppimismallin tulosta ja yhdistää ne sopivalla kombinatoriikalla yhdeksi tulokseksi. Tällä tavalla toteutettu malli ratkaisee ongelman asymptoottiseksi yhtä hyvin kuin paras yksinkertainen koneoppimismalli eli ns. perusoppija. Koska perusoppijoiden kokoelma sisältää monenlaisia koneoppimismalleja, joilla kullakin on omat vahvuutensa voidaan sitä soveltaa erilaisten ongelmien ratkaisuun.

Yleisimmät tavat yhdistää perusoppijoita ovat Bagging [4], Boosting [18] ja Stacking [39]. Näitä kutsutaan myös meta-algoritmeiksi.

Bagging eli Bootsrap Aggregation generoi alkuperäisestä opetusdatasta lisää opetusdataa valitsemalla siitä satunnaisesti alijoukkoja. Kunkin satunnaisesti valitun alijoukon avulla opetetaan yksi kokoelman malleista, jotka sitten yhdistetään samanarvoisina niin, että eniten ääniä saanut tulos valitaan kokoelman tulokseksi. Bagging ei paranna tuloksen tarkkuutta, mutta pienentää varianssia. Satunnaismetsä (Random Forest, RF) on eräs Bagging:in sovellus.

Boosting -menetelmässä kokoelma rakennetaan asteittain, opettamalla jokainen uusi malli-instassi perustuen edellisen mallin kyvykkyyteen. Opettaminen tapahtuu kahdessa vaiheessa. Ensin käytetään alkuperäisen aineiston alijoukkoja, joilla tuotetaan uusia malleja. Sitten mallien kyvykkyyttä parannetaan enemmistön mallien tuloksen perusteella. Toisin kuin Bagging Boosting ei luo opetusdatan alijoukkoja satunnaisesti vaan perustuen edellisten mallien kyvykkyyteen. Kukin uusi alijoukko sisältää edellisten mallien väärin luokiteltuja instansseja.

Bagging ja Boosting käyttävät yleisesti samaa tyyppiä olevia malleja jokaisessa opetusvaiheessa, kun taas pinoamisen (Stacking) vahvuus on eri tyyppiset perusoppijat. Pinoamisen ideaa voidaan hahmottaa ongelma-avaruudella. Kukin perusoppijoista kykenee ratkaisemaan osan ongelma-avaruudesta, mutta ei koko ongelma-avaruutta. Yhdistämällä nämä eri perusoppijat saadaan aikaiseksi ratkaisu, joka löytää parhaan koko ongelma-avaruuden ratkaisun. Pinoaminen ei ole ollut yhtä suosittu kuin Bagging ja Boosting, mutta viime aikoina sitä on sovellettu menestyksekkäästi.

Yleisesti ottaen kokoelmaoppimisen lähestymistavat ovat alttiita ylisovitukselle. Kehitetyssä superoppijassa ristiinvalidoinnin muunnelma minimoi ylisovituksen todennäköisyyttä. Menetelmässä aineisto, jonka koko on n, jaetaan K:hon yhtä isoon joukkoon, joita kutakin käytetään validointiin ja joukon komplementtia K-1 opetukseen. Jokaisella jaolla J ensimmäisen tason oppijaa sovitetaan opetusjoukolla ja ennustetaan validointijoukolla. Yhdistämällä ennustukset jokaisessa vaiheessa saadaan uusi datajoukko Z, jonka koko on n*J. Tämä joukko sisältää ennustukset kaikista näytteistä epäjatkuvasta validointijoukosta jokaiselta perusoppijalta. Tätä uutta aineistoa Z käytetään syötematriisina opettamaan metaoppijan algoritmi, jota sitten

käytettään tekemään lopullinen ennustus.

Vaikka metaoppija voi olla mielivaltaisen monimutkainen, niin ongelman ollessa regressio-tyyppinen käytetään yksinkertaista lineaarista regressiota.

Casas et al. kehittämä GML [7] on toteutus superoppijasta. Ensimmäisen tason oppijoiden lähtöjen yhdistämiseen käytettään todennäköisyyksiin perustuvaa kerroinfunktiota. Jokaisen luokan onnistumistodennäköisyyttä käytetään rakentamaan eksponentiaalisesti hajoavaa kerroinfunktiota. Funktioon lisätään muuttuja, jonka tarkoituksena on vähentää matalan tarkkuuden omaavien mallien kokonaisvaikutusta lopulliseen ennustukseen.

Mallit voidaan niiden lähdön perusteella jakaa kovien ja pehmeiden päätösten malleihin. Kovien päätösten malleissa kynnysarvo on kiinteä, mutta pehmeissä malleissa voidaan päätöksen kynnysarvoa muuttamalla vaikuttaa mallin antamaan ennustukseen, jolloin malliin tulee yksi vapausaste lisää. GML hyödyntää pehmeitä malleja.

Kokoelmaoppijamallilla on taipumus antaa parempia tuloksia kuin yksittäiset perusoppijat, eräs merkittävä tekijä tässä on perusoppijoiden monimuotoisuus. GML käyttää perusoppijoina tukivektorikonetta lineaarisella kernelillä, päätöspuuta (Classification And Regression Trees, CART), k-lähintä naapuria (k-Nearest Neighbor, k-NN), neuroverkkoa (Multi Layer Perceptron, MLP) ja Naiivia Bayes -luokittelijaa (Naive Bayes, NB). SVM:n kanssa käytetään lineaarista kerneliä, koska se on nopeampi kuin oletuksena käytettävä Radial Basis Function (RBF) kernelit. K-NN:ssä aineisto klusteroidaan kymmeneen ryhmään eli parametri k on 10.

Valituilla perusoppijoilla on aikaisempien tutkimusten perusteella todettu hyvä suorituskyky, verkkoturvallisuudessa, poikkeamien havaitsemisessa, luokittelussa ja palvelukokemuksen ennustamisessa [8, 10, 9]. Mallien hyperparametrit valittiin kokeilemalla käyttäen lähtökohtana suositeltuja oletusarvoja.

Alkuperäisessä superoppijaa koskevassa artikkelissa [27] käytettiin esimerkkinä yksinkertaista lineaarista regressiota, mutta superoppija voidaan toteuttaa monella eri tavalla. Selvittääkseen kehittämänsä oppijan hyvyyttä sitä verrattiin muihin tapoihin toteuttaa kokoelmaoppija:

- GML
- MVuniform
- MVaccuracy
- Bagging
- ADABoost

Vertailussa mukana olevat GML, MVuniform ja MVaccuracy perustuvat superoppijaan, mutta ne kukin painottavat perusoppijoita eri lailla, tällä tavoin voidaan tutkia painokertoimien vaikutusta tulokseen. GML laskee painokertoimet oppijoille käyttäen eksponentiaalista luokittelutodennäköisyyttä.

$$w_j = \frac{e^{\lambda \alpha_j}}{\sum_{i=1}^{J} e^{\lambda \alpha_i}},$$

jossa λ:n tehtävänä on pienentää alhaisen tarkkuuden oppijoiden vaikutusta.

Tutkijat käyttävät termiä MVuniform menetelmälle, joka käyttää kaikille oppijoille samaa painokerrointa (1/J), jolloin yksikertaisesti ottaen valitaan eniten ääniä saanut ratkaisu. Vastaavasti termiä MVaccuracy käytetään menetelmälle, joka käyttää painokertoimena kullekin oppijalle kertoimena sen onnistumistodennäköisyyttä jaettuna kaikkien oppijoiden onnistumistodennäköisyyksien summalla.

$$w_j = \frac{\alpha_j}{\sum_{i=1}^{J} \alpha_i}.$$

Tutkimuksessa sekä bagging että boosting toteutettiin käyttämällä päätöspuuhun perustuvia malleja. Baggingin tapauksessa kokeiltiin sekä Bagging tree että satunnaismetsä algoritmeja. Näiden menetelmien ero on siinä mitä ominaispiirteitä käytetään uusiin oksiin jakauduttaessa. Bagging:ssä jakautumisessa käytetään kaikkia ominaispiirteitä, kun taas satunnaismetsässä käytettään vain satunnaisesti valittua ominaispiirteiden alijoukkoa.

Boostingia edustamaan oli valittu AdaBoost (Adaptive Boosting) [19]. ADABoost käyttää perusoppijoinaan päätöspuita, joita opetetaan käyttäen edellisten perusoppijoiden väärin luokittelemia instansseja.

Menetelmiä verrattiin viiden erityyppisen aineiston avulla: älypuhelinsovelluksen poikkeamien havaitseminen, verkkohyökkäysten havaitseminen, matkapuhelinverkon palvelun laadun ennustaminen, videon suoratoiston palvelun laadun mallintaminen ja Internetpolkujen dynaaminen seuranta.

Älypuhelinsovelluksen poikkeamien havaitsemisen vertailua varten käytettiin suurelta eurooppalaiselta matkapuhelinoperaattorilta kerättyjä DNS-jälkiä, joita yhdistelemällä luotiin uusia DNS-jälkiä, joihin lisättiin kolmea erityyppistä poikkeamaa. Nämä poikkeavat perustuvat oikeisiin mitattuihin poikkeamiin. Poikkeamat ovat useita tunteja kestävä intensiivinen, useita päiviä kestävä matalan intensiteetin omaava ja lyhytkestoinen vaihtelevan intensiteetin omaava.

Verkkohyökkäysten havaitsemisen vertailuun käytetään MAWILab -aineistoa [17], joka on yleisesti saatavilla. Aineisto on kerätty Japanin ja Yhdysvaltojen välisestä runkolinjasta. MAWILab-projekti on neljän perinteisen poikkeamien havaitsijan avulla osittain luokitellut aineiston. Tutkimuksessa oli keskitytty luokkaan, jossa olevat poikkeamat havaittiin samanaikaisesti kaikilla neljällä havaitsijalla.

Matkapuhelinverkon palvelun laadun ennustamisen vertailuun käytettiin kenttäkokeista kerättyä aineistoa. Kokeessa 30 käyttäjää arvioivat kolmen eri sovelluksen, Youtube, Facebook ja Gmaps, palvelun laatua omilla laitteillaan. Palvelun laatua arvioitiin viisiportaisella asteikolla, jolloin luokittelu ongelman tehtävänä on tietoliikenne tietojen perusteella ennustaa oikea palvelun laadun luokka.

Videon suoratoiston palvelun laadun mallintamisen vertailun aineistona käytettiin yleisesti saatavilla olevaa LIVE-Avvasi Mobile Video tietokantaa [21]. Aineisto koostuu 174:stä vääristyneestä videosta, jotka on generoitu 24 lähdevideosta ja joissa on 26 ainutlaatuista viivytystä. Aineistoon kuuluu myös 54 katsojan tekee 4830 viisiportaista luokittelua videoista. Videot sisältävät monipuolisesti erityyppistä sisältöä nopeasti muuttuvista urheiluvideoista hitaammin muuttuviin dokumenttivideoihin. Aineistosta on otettu erilleen 19 ominaispiirrettä kuvaamaan häiriöitä ja videon sisältöä.

Internetpolkujen vaihtelun ennustamisen vertailuun käytettiin aineistoa M-Lab avoin Internetin mittausaloite mittauksista (https://www.measurementlab.net/). Aineisto koostuu traceroute mittauksista ja se sisältää yli 450 000 eri polkua. Tulokset on kerätty yli 180 ympäri maapalloa sijoitetusta palvelimesta. koneoppimistehtävä on ennustaa kuinka monta polkumuutosta tiettynä päivänä tapahtuu edellispäivän tietojen perusteella. Datan ominaispiirteitä ovat mm. polun dynamiikan ja latenssin tilastolliset ominaisuudet. Tehtävä on muutettu luokitteluksi määrittelemällä kategoriat: staattinen polku, dynaaminen polku ja erittäin dynaaminen polku. Staattisessa polussa ei tapahdu yhtään polun muutosta, dynaamisessa polussa tapahtuu yhdestä kymmeneen muutosta ja erittäin dynaamisessa polussa muutoksia tapahtuu enemmän kuin kymmenen.

5.3 Vertailu

Verkon suorituskykyyn vaikuttaa siinä esiintyvät häiriöt, siksi onkin tärkeää kyetä tunnistamaan häiriöt ajoissa, jotta niiden vaikutus verkon suorituskykyyn jäisi mahdollisimman pieneksi. R. Sasisekharan et al. [33] tutki häiriöiden tunnistamista koneoppimisen avulla. Lineaarinen diskriminaattianalyysi (LDA) antoi yhtä huonon tuloksen, kuin jos kaikki tapaukset olisi luokiteltu isoimpaan luokkaan. Sen sijaan neuroverkot, päätöspuut ja päätössäännöt antoivat paremmat tulokset ja niiden perusteella voidaan tunnistaa olosuhteet, joiden esiintyessä kyseessä on todennäköisesti häiriö.

Tutkittaessa videon suoratoistopalvelun palvelun laadun ennustuksen tarkkuutta Casas et al. kehittämä kokoelmaoppija GML [7] saavuttaa hyvät tulokset kaikissa viidessä kategoriassa. Kategoriassa DMOS (Degradation MOS score) = 2 satunnaismetsä ja kategoriassa DMOS = 4 MVuniform ovat parempia, muutoin GML on tarkin menetelmä. Edellä mainituissa kahdessa kategoriassakin GML on aivan parhaiden menetelmiä takana.

Yleisesti otettuna kokoelmaoppijat, kuten satunnaismetsä, bagging, boosting ja erityisesti tutkimuksessa parhaiten suoriutunut GML, antavat tarkempia tuloksia kuin perusoppijat, kuten päätöspuu, Naiivi Bayes, neuroverkot, tukivektorikone ja k-Lähin naapuri.

6 Poikkeamien havaitseminen ja tietoverkon turvallisuus

Tietoverkoissa liikkuu kasvavassa määrin enemmän tietoa. Koska nämä tietomassat on saatava nopeasti oikeisiin kohteisiinsa on tietoverkoissa olevien laitteiden määrä kasvanut ja tehnyt tietoverkkojen rakenteesta monimutkaisia. Verkon toiminnan kannalta on tärkeää, että sitä valvotaan, mutta koska verkot ovat isoja ja niissä liikkuu paljon dataa, niiden valvonta synnyttää ison määrän dataa, josta ei ole helppoa nopeasti havaita verkon tilassa tapahtuvia muutoksia.

6.1 Taustaa

Eräs tärkeimmistä verkon tilan tarkkailun tavoitteista on automaattisesti havaita tietoliikenteen poikkeamat ja verkkohyökkäykset. Verkkoliikenteessä liikennevirrat alkavat ja loppuvat jatkuvina ketjuina, jolloin tästä muuttuvien signaalien joukosta on vaikeaa havaita ne merkit, jotka kertovat poikkeavasta toiminnasta tai verkkohyökkäyksestä.

On vaikeaa täsmälleen määritellä se tarkkailudatan muuttujien joukko, joiden muutosten perusteella tapahtuma voidaan luokitella poikkeamaksi. Yleisen poikkeaman havaitseminen on vaikeaa, koska on olemassa iso joukko erilaisia poikkeamia, joilla on varsin monipuolinen rakenne. Ratkaisua tähän ongelmaan on haettu koneoppimisesta. Koneoppimisella tarkkailudatasta opitaan poikkeamien perustana olevat ominaisuudet.

Verkon tilan tarkkailusta syntyvän suuren datamäärän analysointi on vaikeaa, koska dataa syntyy nopeasti ja se on monimuotoista. Analysoitava data käsittää mm. pakettien otsikko- ja kokotietoja, puskureiden kuormitustilastoja ja linkkien toteutuneita siirtonopeuksia. Suurten datamäärien analysointia varten on kehitetty erilaisia alustoja, joita voidaan hyödyntää myös poikkeamien havaitsemiseen.

6.2 Menetelmät

Tietoverkon tilan tarkkailu vaatii sekä suuren tietovuon reaaliaikaista analyysia että suuren lokiaineiston tehokasta käsittelyä erikseen. Suurten tietomäärien hallintaa ja analysointia varten Casas et al. [6] on kehittänyt big datan analyysialustan, joka on erityisesti suunniteltu tietoverkkojen monitorointisovelluksille. Alustan kehittämisen lähtökohtana on ollut luoda yleiskäyttöinen alusta verkon hallinnan avuksi. Aikaisemmin on toteutettu moninaisia ratkaisuja johtuen erilaisista vaatimuksista, tavoitteista ja tarpeista. Tällöin soveltajat ovat joutuneet käymään läpi ison joukon ratkaisuja etsiessään itselleen sopivia.

Tutkimuksessaan Casas et al. [6] testasivat kuinka hyvin viisi eri koneoppimismenetelmää kykenevät havaitsemaan viisi eri verkkohyökkäystä ja kuinka nopeasti ne suoriutuvat laskennasta. Testaukseen he käyttivät julkisesti saatavilla olevaa

MAWILab-aineistoa [17], joka on peräisin Yhdysvaltojen ja Japanin välisestä WIDE runkoverkosta. Käytetty aineisto on peräisin loppuvuodelta 2015 ja koostuu 15 minuutin aikajaksoista liikenteen jälkiä.

Havaittavat verkkohyökkäykset ovat hajautettu palvelunestohyökkäys (Distributed Denial-of-Service, DDoS), HTTP flash crowds (mptp-la), Flooding -hyökkäys (PING-flood), sekä kaksi hajautettuun verkkoskannaukseen perustuvaa hyökkäystä, joissa toisessa hyödynnettiin UDP:tä ja toisessa TCP-ACK:ta. Tutkimuksessa kutakin hyökkäystä varten opetetaan oma erillinen mallinsa, jolloin hyökkäystyypit voidaan havaita suorittamalla kuhunkin malliin liittyvät laskennat rinnakkain.

Poikkeaminen havaitsemiseen kokeillaan viittä ohjatun koneoppimisen menetelmää, jotka on valittu aikaisempien havaintojen perusteella [12, 30]. Menetelmät ovat päätöspuu, satunnaismetsä, tukivektorikone, Naiivi Bayes ja neuroverkko (MLP).

Aineistossa olevat liikennejäljet pilkottiin sekunnin mittaisiin pätkiin, joista laskettiin kullekin aikavälille ominaispiirteet. Ominaispiirteet koostuivat mm. pakettien ja tavujen lukumääristä, pakettien ko'oista, IP-osoitteista, siirtoprotokollista, porteista jne. Ominaispiirteisiin laskettiin lisäksi erilaisia aikavälikohtaisia tilastollisia suureita, kuten keskiarvo, keskihajonta jne. Yhteensä ominaispiirteitä oli kaiken kaikkiaan 245.

Poikkeamien havaitsemiskykyä arvioitiin vertaamalla oikein poikkeamaksi luokittelujen tapahtumien osuutta kaikista poikkeamista eli herkkyys (sensitivity) ja väärin poikkeamaksi luokittelujen tapahtumien osuutta kaikista ei-poikkeamista. Näiden perusteella arvioidaan oikein ja väärin poikkeamiksi luokiteltujen tapahtumien osuutta kaikista poikkeamiksi luokitelluista tapahtumista.

Testissä huomattiin kaikkien menetelmien, Naiivia Bayesia lukuun ottamatta, antavan hyvän tarkkuuden kaikkiin hyökkäyksiin, havaintotarkkuuden ollessa DDoS:lle hieman heikompi. Parhaiten kokeilussa suoriutuivat neuroverkko ja satunnaismetsä, jotka kykenivät luokittelemaan poikkeamat oikein noin 80% tarkkuudella ilman, että mukana olisi väärin poikkeamiksi luokiteltuja tapahtumia.

Käytännön sovelluksissa algoritmin vaatima laskentakapasiteetti vaikuttaa sen tarkkuuden kanssa algoritmin käytettävyyteen. Vertailtaessa menetelmien opetusaikoja havaittiin, että neuroverkon opetus kestää kolme kertaluokkaa (10^3) kauemmin kuin satunnaismetsän, joten tutkituista menetelmistä tarkkuuden ja opetusajan perusteella satunnaismetsä on käyttökelpoisin.

Lisäksi tutkittiin ominaispiirteiden määrän vaikutusta ennustustarkkuuteen ja laskenta-aikaan. Ensinnäkin ominaispiirteiden määrän lisääminen kasvattaa laskenta-aikaa. Toisaalta havaintojen mukaan useimmilla ohjatun oppimisen menetelmillä ennustustarkkuus paranee, kun ominaispiirteiden määrä lisääntyy. Ominaispiirteiden joukossa saattaa olla merkityksettömiä tai tarpeettomia ominaispiirteitä, joiden käyttö paitsi kasvattaa laskentaan käytettyä aikaa, mutta saattavat myös heikentää ennustustarkkuutta [11].

Laskenta-aikaa voidaan lyhentää valitsemalla vain olennaisimmat ominaispiirteet. Tällaisia ovat ominaispiirteet, jotka korreloivat parhaiten tuloksen kanssa. Tutki-

muksessa vertailtiin kahta eri tapaa valikoida laskentaan käytettävät ominaispiirteet: Plain-top (PLCC, Pearson Linear Correlation Coefficient) ja Sub-set search selection (FS, Feature Selection). PLCC:ssä valitaan ne ominaispiirteet, jotka korreloivat lineaarisesti kaikkein eniten. FS:ssä valitaan ominaispiirteiden alijoukkoja, jotka korreloivat heikosti toistensa kanssa, mutta korreloivat hyvin tavoitteiden kanssa.

Ominaispiirteet korreloivat heikosti hyökkäyksiin, esim. PLCC:llä korrelaatiokertoimet ovat yleisesti alle puolen. Mielenkiintoinen huomio on, että verrattuna muihin hyökkäyksiin DDoS -hyökkäyksellä harvemmat ominaispiirteistä ovat hyvin korreloivia. Tämä sopii yhteen edellisen kohdan havainnon kanssa, jossa DDoS:n havaintotarkkuus oli heikoin.

PLCC:llä merkittävän korrelaation raja-arvoksi valittiin 0,2, jolloin eri hyökkäyksien havaitsijoille saatiin seuraavat merkittävien ominaispiirteiden lukumäärät: DDoS 11, HTTP flashcrowd 29, Ping flood 51, UDP skannaus 45 ja TCP skannaus 47. Vastaavasti FS:llä saatiin merkittävien ominaispiirteiden lukumääriksi: DDoS 13, HTTP flashcrowd 19, Ping flood 19, UDP skannaus 21 ja TCP skannaus 19. Kun alkuperäinen ominaispiirteiden lukumäärä oli 245 niin nyt saadut merkittävien ominaispiirteiden lukumäärät ovat noin kymmenesosa alkuperäisestä.

Laskentaan käytettävien ominaispiirteiden vähentämisen ansiosta kaikissa tapauksissa laskenta-ajat odotetusti pienenevät, joissain tapauksissa merkittävästikin. Esimerkiksi parhaimmat havaintotarkkuudet edelleenkin omaavien neuroverkkojen ja satunnaismetsän laskenta-aikojen ero ei ole enää kuin 10-20 kertainen. Havaintotarkkuudet muutamaa poikkeusta lukuun ottamatta heikkenivät odotetusti, mutta CART:in havaintotarkkuus jopa hieman parani.

Tutkimuksessa tarkasteltiin vielä miten merkittävimmät ominaispiirteet suhtautuivat hyökkäystyyppien kanssa. Havaintona todettiin, että valitut merkittävimmät ominaispiirteet ovat hyvinkin sopusuhdassa hyökkäystyypin tunnusmerkkien kanssa, esim. palvelunestohyökkäyksessä lähetetään suuri määrä paketteja kohteen IP-osoitteeseen ja palvelun porttiin, joten nämä näkyvät myös merkittävimpien ominaispiirteiden joukossa.

6.3 Vertailu

Poikkeamien havaitsemista oli testattu viidellä erilaisella ohjatun koneoppimisen menetelmällä: Päätöspuilla, satunnaismetsällä, Tukivektorikoneella, naiivilla Bayesilla ja neuroverkolla. Heikoimman tuloksen antoi naiivi Bayes. Vastaavasti parhaat tulokset saatiin neuroverkolla ja satunnaismetsällä.

Neuroverkkojen opettaminen vei paljon kauemmin kuin satunnaismetsän opettaminen. Opetusaikoja voi lyhentää vähentämällä käytettävien ominaispiirteiden lukumäärää. Eri ominaispiirteet korreloivat sekä toistensa että tuloksen kanssa eri lailla. Valitsemalla käytettäväksi ne ominaispiirteet, jotka korreloivat sopivasti voidaan ominaispiirteiden määrää vähentää ilman, että menetelmän tarkkuus merkittävästi heikkenee.

Tutkimuksessaan Casas et al. [6] olivat vertailleet PLCC ja FS menetelmiä, joista FS valikoi käytettäväksi hieman vähemmän ominaispiirteitä kuin PLCC ilman, että niiden välillä on merkittävää eroa tarkkuudessa. Opetusaikojen lyheneminen korreloi ominaispiirteiden vähenemisen kanssa kuitenkin niin, että neuroverkoilla päästiin opetusaikoihin, jotka olivat enää korkeintaan kertaluokkaa suurempia kuin muilla menetelmillä, erityisesti satunnaismetsällä.

Kaikilla menetelmillä tarkkuus oli heikoin havaittaessa palvelunestohyökkäyksiä. Erilaisten poikkeamien välillä olevat erot tarkkuudessa korreloi merkittävien ominaispiirteiden määrään niin, että mitä vähemmän merkittäviä ominaispiirteitä sitä huonompi tarkkuus. Tutkimustulosten perusteella ei voi päätellä, että toiset koneoppimismenetelmät sopisivat paremmin joihinkin poikkeamiin ja toisen menetelmät joihinkin muihin poikkeamiin, vaan varsin yleisesti satunnaismetsä ja neuroverkko ovat tarkimmat, ja näistä vielä satunnaismetsä on kertaluokkaa nopeampi opetettaessa.

7 Reititys tietoverkossa

Tietoverkossa on useita laitteita, jotka kommunikoivat keskenään. Jotta laitteiden lähettämä tieto löytäisi perille ja tiedonvälitys olisi tehokasta, tarvitaan reititystä, joka ohjaa lähetetyt viestit kohti vastaanottajaksi tarkoitettua laitetta. Reitityksestä vastaavat siihen tarkoitetut laitteet, reitittimet. Jotta reititin tietää mikä on paras reitti, sen on kerättävä ja ylläpidettävä tietoa verkon rakenteesta. Isot verkot ovat kuitenkin harvoin staattisia vaan osa yhteyksistä voi ruuhkautua tai jopa katketa, siksi ei aina voida käyttää yksinkertaista Lyhin Polku -strategiaa, vaan reittien arvioimiseen on käytettävä monenlaista tieto reittien ja reitin varrella olevien laitteiden tilasta. Reittien määrittäminen ja ylläpito on dynaamista toimintaa, joka kuluttaa varsin paljon laskentaresursseja.

7.1 Taustaa

Isossa tietoverkossa on paljon reitittimiä, jolloin reititystauluista tulee isoja. Reittien laskeminen ja jatkuva päivittäminen isoihin reititystauluihin on aikaa kuluttavaa. Ratkaisuna on ollut toteuttaa verkon reititys hierarkkisesti jakamalla verkko autonomisiin järjestelmiin (Autonomous System, AS). Autonomisten järjestelmien sisällä käytetään RIP (Routing Information Protocol) ja OSPF (Open Shortest Path First) protokollia. Autonomisten järjestelmien välillä käytetään BGP (Border Gateway Protocol) protokollaa.

RIP:ssä reititin välittää reititystaulun lähettämällä sen säännöllisesti kaikille, joten se soveltuu vain pieniin tietoverkkoihin. OSPF:n toiminta perustuu linkkien tilaan ja siinä reititystaulu lähetetään muille reitittimille vain, kun linkkien tilassa tapahtuu muutos. OSPF soveltuu isompiin verkkoihin ja sitä käytettään mm. yritysten isoissa tietoverkoissa.

Kasvavat tietoverkot ja liikennemäärät vaativat verkkolaitteilta lisääntyvää suorituskykyä, jota varten sekä teollisuus että akateeminen maailma kehittävät uusia reititysalgoritmeja. Reitittimet on kuitenkin usein valmistettu niin, että ohjelmisto on niissä laitekohtainen, jolloin uusia reititysalgoritmeja varten on tehtävä myös uudet laitteet. Jotta uudet reititysalgoritmit saadaan nopeasti käyttöön, tarvitaan ohjelmoitavia reitittimiä (Software Defined Routers, SDR). Nämä reititysalustat eivät pelkästään huolehti reitityksestä vaan myös kykenevät suorittamaan muitakin funktioita.

Tietoverkot ovat perusluonteeltaan hajautettuja järjestelmiä, joiden kontrolli on myös yleensä hajautettu. Toisaalta tietoverkkojen tuottama data on usein paikallista esim. yhden reitittimen tai palvelimen tuottamaa. Koneoppimismallin hyödyntäminen hajautetun järjestelmän kontroloimiseen paikallisten tietojen perusteella on haastavaa. Jotta verkon kapasiteetti saataisiin täysin hyödynnettyä tarvitaan näkemystä verkon kokonaistilanteesta.

Ohjelmisto määritellyt verkot (Software Defined Networking, SDN) on malli, jossa laitteiden kontrollikerrokset on yhdistetty verkon kontrollikerrokseksi, jolla on kokonaisnäkemys verkon tilanteesta ja osaa sen perusteella hyödyntää verkon kapasiteettia täysipainoisesti.

Tavanomaisesti reititystaulut ovat toteutettu sääntöperusteisesti, esimerkiksi Lyhyin Polku -algoritmilla, joka valitsee reitin jonkin mittarin minimiarvon perusteella. Tällainen strategia konvergoi hitaasti, eivätkä ne välttämättä sovellu monen mittarin perusteella laskettavaan reititykseen, sillä mittareiden välisiä yhteyksiä on vaikea ennustaa. Käyttämällä koneoppimisen menetelmiä saadaan hyödynnettyä monen eri mittarin tietoja reittejä laskettaessa.

7.2 Menetelmät

Ensimmäiset reititykseen liittyvät koneoppimissovelluksista toteutettiin tavanomaisilla koneoppimistekniikoilla. Nämä sovellukset toteuttivat reitityksen sääntöihin perustuvilla menetelmillä, jotka ovat tehottomia käsittelemään useita verkkoparametreja ja toisaalta niillä on vaikeuksia karakterisoida syötteitä ja tulosteita. Eräänä ratkaisuna on käyttää syväoppimista. Mao et al. [28] esittää syväoppimiseen perustuvaa reititystaulun muodostusta.

Ohjelmoitavia reitittimiä (SDR, Software Defined Routers) käyttämällä voidaan hyödyntää ohjatun syväoppisen (Supervised Deep Learning) menetelmiä tietoverkon reitityksen tehostamiseksi. Mao et al. [28] toteuttama reititys käyttää opetusmateriaalina liikennemalleja, joista se ohjatun syväoppisen avulla ratkaisee reitit.

Artikkelin mukaan syväoppimiseen perustuvan reititysstrategian hyödyt verrattuna OSPF:ään ovat matalammat signalointikustannukset, suurempi läpimeno ja pienempi keskimääräinen linkkikohtainen viive sekä nopeampi konvergoituvuus, jolloin liikennettä on helpompi kontrolloida.

Eräs tapa hyödyntää koneoppimusta on imitoida laskennallisesti raskasta heuris-

tista algoritmia. L. Yanjun et al. [40] laskivat tehokkaalla, mutta monimutkaisella, heuristisella algoritmilla eri pyynnöille reitityksiä, joista he sitten muodostivat opetusjoukon ohjatulle koneoppimiselle, jonka avulla muodostettiin malli jäljittelemään monimutkaista heuristista algoritmia. Käytännön ratkaisussa monimutkainen heuristinen algoritmi korvattiin nopeammin laskettavalla koneoppimismallilla.

Heuristisena algoritmina käytettiin MMAS (max-min ant system), joka on parannettu versio Ant Colony Optimoinnista. Tutkimuksessa verrattiin Lyhyin Polku Ensin (Shortest Path First, SPF) -algoritmia MMAS -algoritmiin sekä siitä laskettuun koneoppimismalliin. Simuloinnit tehtiin kahdella eri verkolla, joissa toisessa oli 15 solmua ja 27 linkkiä (KL-verkko) ja toisessa 14 solmua ja 21 linkkiä (NSFNet). Koneoppimiseen käytettiin neuroverkkoa, jossa oli joko 54 tai 42 ominaispiirrettä ja piilokerroksissa 600 tai 2000 neuronia tarkasteltavan verkon mukaan. Lisäksi lähdöissä 8 - 10 lähtöneuronia.

Tutkimuksen vertailtiin verkkoviiveitä ja havaittiin, että verkon ollessa kevyesti kuormitettu, ei SPF:n ja MMAS:n välillä ollut suurta eroa, mutta kuormituksen kasvaessa viiveet SPF:llä reititetyssä verkossa kasvavat nopeammin kuin MMAS:llä. MMAS:n ja sen koneoppisella korvatussa versiossa viiveet ovat lähes samat.

Reitinlaskemiseen SPF:llä meni noin 0,01 ms, MMAS:llä 250 ms ja sen koneoppimisversiolla 0,04 ms. Laskentatuloksista näkee, että MMAS heuristisena algoritmina on liian hidas käytettäväksi reaaliaikaiseen reititykseen, mutta sen koneoppimisversio on jo toimiva, vaikka se ei olekaan yhtä nopea kuin SPF-algoritmi. Käyttämällä MMAS:n koneoppimisversiota hyödynnetään monimutkaisen raskasta laskentaa vaativan algoritmin reititystehokkuus vähemmällä laskennalla niin, että sitä voidaan käyttää reaaliaikaisessa reitityksessä.

Myös Azzouni et al. [2] on tutkinut dynaamisen reitityksen toteuttavan heuristisen algoritmin korvaamista koneoppimisella. Vaikka heuristiset ratkaisut dynaamisen reitityksen ratkaisemiseksi ovatkin tehokkaita, ne eivät ole laskennallisesti useinkaan käyttökelpoisia. Tutkimuksen ongelmassa annetaan pyyntömatriisi, josta on tavoitteena maksimoida verkon kokonaisläpäisy minimoiden reitityskustannukset. Artikkelissa on osoitettu, että ongelma on NP-täydellinen, joten erilaisia polynomiaikaisia approksimaatiota on kehitetty.

Tutkimuksessa on ollut käytettävissä 10 000 näytteen aineisto, jossa kukin näyte koostuu liikennematriisista, verkontilasta ja lähes optimaalisesta reitistä. Lähes optimaalinen reitti on laskettu liikennematriisista ja verkontilasta käyttäen heuristista algoritmia.

Koneoppimiseen käytettävän neuroverkon rakenne on valittu kokeilemalla eri kokoonpanoja ja niiden suorituskykyä. Mallien rakenteen monimutkaistuessa niiden tarkkuus paranee, mutta oppimiseen käytetty aika kasvaa, samoin kuin myös laskenta-aika mallia käytettäessä. Mallien laskentaan käytettyä aikaa on rajoitettu kahteen minuuttiin ja niiden tarkkuutta on arvioitu keskineliö virheellä. Tällä tavoin on valittu neuroverkolle kuusi piilokerrosta, joissa jokaisessa on 100 solmua.

Näytteet jaetaan opetusta varten 100 näytteen eriin. Kukin näyte on 544-ulotteinen

vektori. 506 ulottuvuutta koostuu liikennematriisista (23 solmua * 22 solmua) ja 38 ulottuvuutta saadaan verkontilasta (38 linkkiä). Malli antaa vektorin, jossa on liikennematriisi (23*22) ja kuusi mahdollista reittiä.

Menetelmässä on käytetty Dropout-menetelmää ylisovittamisen ehkäisemiseksi. Dropout:ssa opetuksen aikana satunnaisesti valittuja neuroneja pudotetaan väliaikaisesti laskettaessa painokertoimia eteenpäin. Eikä niitä myöskään päivitetä edettäessä takaisin päin. Näin neuroverkosta tulee vähemmän altis tietyille neuronien painokertoimille, jolloin malli yleistyy paremmin.

Malli osaa ennustaa heuristiseen menetelmään perustuvan lähes parhaan reitin alle 0,05% virheellä (Kuvaajan perusteella oikea virhe on alle 5%.). Mallin oppimiseen meni aikaa kolme minuuttia. Mallin etu tulee sen nopeudesta löytää lähes paras reitti neljäsosalla siitä ajasta mikä meni heuristisella ratkaisulla.

A. Valadarsky et al. [36] olivat havainneet, että epäsäännöllisessä liikenteessä ohjattu oppiminen tuotti tehottomasti reitittäviä malleja. Ratkaisuna he lähtivät tutkimaan vahvistusoppimisen soveltamista. Lähtökohtana oli, ettei erikseen opiskella tulevan liikenteen pyyntöjä ja sitten niiden perusteella optimoida konfiguraatioita, vaan opiskellaan hyvä kuvaus havaituista pyynnöistä reitityskonfiguraatioihin. Oletuksena oli, että vaikka verkon aikaisemmat tilat eivät suoraan määritä tulevia tiloja, niin siinä on kuitenkin jotain tietoa tulevista tiloista. Esimerkkinä voidaan mainita liikenteen vuorokauden ajan mukaiset vaihtelut tai tietyille asiakkaille tyypilliset liikennöintitavat.

Ohjattu oppiminen on luonnollinen lähestymistapa, kun yritetään ratkaista reititystä koneoppimisella. Havaintoina tällöin käytettään menneen liikenteen pyyntöjä, joista koneoppimista soveltamalla ennustetaan tulevan liikenteen pyynnöt. Näitä ennustettuja tulevia liikenteen pyyntöjä käyttäen ennustetaan reititys. Tutkijat arvioivat kolmea ohjatun oppimisen malleja: Täysin kytketty 3-kerroksinen neuroverkko (FCN), 4-kerroksinen konvoluutioneuroverkko (CNN) sekä 4-kerroksinen epälineaarinen auto-regressiivinen neuroverkko (Nonlinear AutoRegressive Neural Network, NAR-NN) ja pitivät ohjattua oppimista tehottomana ratkaisuna dynaamiseen reititykseen ellei liikenne ole erittäin säännöllistä.

Vahvistusoppimisessa ei erityisesti opiskella tulevan liikenteen pyyntöjä ja sitten optimoida reititystä niiden mukaan, vaan opitaan hyvä kuvaus havaitusta menneestä liikenteestä reitityskonfiguraatioihin. Toimija vuorovaikutuksessa ympäristön kanssa havainnoi nykyisen tilan ja valitsee mielestään tilanteeseen parhaiten sopivan toiminnon. Verkon tila muuttuu ja toimija saa palkkion sen perusteella kuinka hyvä sen valitsema toiminto oli. Tämän prosessin tarkoituksena on oppia kuvaus joukosta mahdollisia tiloja joukkoon toimintoja. Tavoitteena on kuvaus, joka maksimoi odotettavissa olevan alennetun palkkion (expected discounted reward).

Myös reititysstrategian valinta voidaan muuttaa vahvistusoppimistehtäväksi. Jokaisen aikajakson aluksi systeemi valitsee reititysstrategian, sille aikajaksolle perustuen tiettyyn määrään edellisiä reititysstrategioita ja pyyntömatriisiin eli havaittuun dataan. Tila muuttuu, kun uusi pyyntömatriisi paljastuu ja saadaan palkkio

$$r^{(t)} = -\frac{u^{(t)}}{OPT^{(t)}},$$

jossa $u^{(t)}$ on max-link-utilization tietyllä reititysstrategialla pyyntömatriisille $D^{(t)}$ ja $OPT^{(t)}$ on optimaalinen max-link-utilization samalle pyyntömatriisille $D^{(t)}$. Palkkio $r^{(t)}$ on siis saavutetun tehokkuuden ja optimaalisen tehokkuuden suhde.

Tavoitteena on oppia kuvaus tietyn kokoisesta pyyntömatriisihistoriasta reititysstrategioihin, jotka maksimoivat odotettavissa olevan alennetun palkkion.

Tutkimusta varten toteutettiin jatkuvasti kontrolloiva vahvitusoppimisalgoritmi (Trust Region Policy Optimization, TRPO) [34] 3-kerroksisella täysin kytketyllä neuroverkolla (FCN). Menetelmän optimoinnin tavoitteena oli ruuhkautumisen minimointi. Vertailukohteiksi valittiin kolme eri menetelmää, jotka eivät perustu koneoppimiseen: 1) softmin reitityksen optimointi perustuen viimeisimpään pyyntömatriisiin, 2) softmin reitityksen optimointi perustuen k:hon viimeisimpään pyyntömatriisiin ja 3) Oblivious [1], optimaaliseen reititysstrategiaan perustuva.

Harvassa verkossa tutkittava vahvistusoppimisen menetelmä pääsee lähelle optimaalista Oblivious reititysstrategiaa ja on parempi kuin muut kaksi verrokkia. Tiheässä verkossa tutkittava vahvistusoppimisen menetelmä päihittää kaikki kolme verrokkimenetelmää.

Koneoppimista voidaan hyödyntää reititykseen luokittelemalla liikennettä sen vaatimusten perusteella. Pasca et al. [31] on tutkinut koneoppimiseen perustuvan luokittelun käyttämistä rajallisten liikennöintiresurssien parempaan hyödyntämiseen.

Erilaisilla sovelluksilla on erilaiset vaatimukset liikennöintiresurssien suhteen. Videopuhelusovellukset vaativat paitsi suuren kaistanleveyden myös pienen viiveen, toisaalta tekstipohjainen ohjaussovellus saattaa siirtää muutaman merkin kerrallaan, mutta vaatia pienen viiveen ja yksittäisiä tiedostoja siirrettäessä ei viiveellä ole suurta merkitystä, kunhan tiedosto saapuu nopeasti ehjänä perille.

Ongelmana on, ettei sovelluksesta saada helposti tietoa. Esimerkiksi suurin osa liikenteestä on HTTP- tai HTTPS-liikennettä, jonka sisällä on varsin erilaisia vaatimuksia omaavia sovelluksia. Sovellukset on pystyttävä luokittelemaan tarkemmin niiden tarvitsemien tietoliikenneresurssien, kuten kaistanleveyden, viiveen tai latenssin, mukaan.

Lisäksi verkko muuttaa tilaansa lähes jatkuvasti, toiset tiedonsiirrot alkavat ja toiset loppuvat, osa laitteista voi ruuhkautua tai vikaantua. Tämän vuoksi myös eri linkeillä olevat ominaisuudet, kuten kaistanleveys ja viive, vaihtelevat liikenteen mukana. Alati muuttuvan verkon tilan vuoksi ei ole järkevää reitittää tietovoita staattisesti.

Verkon tehokkuutta voidaan parantaa priorisoimalla eri sovellusten tietovuot ja sitten myöntää resursseja niiden edellyttämien vaatimusten mukaisesti. Tämä kuitenkin edellyttää tietovoiden ominaisuuksien tunnistamista siten, että vuot voidaan luokitella, tähän voidaan hyödyntää koneoppimista. Lisäksi tarvitaan älykästä reititystä, jolloin lähtökohtana on ohjelmisto määritelty verkko (SDN), jossa verkon

hallinta on keskitetty kontrollerille.

Tutkimuksen ratkaisussa on kontrollerissa sijaitsevat erilliset koneoppimisen opetusmoduuli ja kontrollimoduuli. Opetusmoduulia käytetään mallin luomiseen järjestelmän käytöstä erillisenä ja kontrollimoduli käytönaikaisesti luokittelee tietovoita saamiensa syötteiden perusteella. Lisäksi kontrolleri mittaa toistuvasti latenssia ja käytettävissä olevaa kaistanleveyttä linkeillä, joiden avulla lasketaan kullekin linkille reittikustannus reititystä varten.

Tutkimuksen [31] opetusjoukon syötteet koostuivat vektoreista, joissa on syötettä vastaava luokka ja 40 ominaispiirteestä, joita olivat mm. lähteen IP-osoite ja portti, kohteen IP-osoite ja portti, protokolla, sekä lähetettävien että saapuvien pakettien kokonaismäärä, tietovuon kokonaismäärä kumpaankin suuntaan, sekä molempien suuntien pakettien pituuden keskiarvo ja -hajonta jne. Parhaiten soveltuvimman luokittelijan löytämiseksi testattiin erilaisia ohjatun oppimisen luokittelijoita, kuten naiivia Bayesia, C4.5 päätöspuuta, Bayesilaista verkkoa ja tukivektorikonetta luokittelemaan liikennettä. Parhaimman tuloksen antoi C4.5 päätöspuu 98 % tarkkuudella, joka valittiin käytettäväksi. Myös menetelmiä puoliohjatun oppimisen hyödyntämisestä on tutkittu [38].

Tutkimuksissa havaittiin, että C4.5 päätöspuu käyttää neljää kenttää luokitteluun, joihin merkittävimpinä kuuluu pakettien välinen aikaväli ja pakettien pituus. Luokittelijan toiminta nopeutuu, kun luokittelu voidaan tehdä vain neljän ominaispiirteen avulla eikä tarvita kaikkia 40 ominaispiirrettä.

Kun luokittelija on opetettu, jokainen uusi tietovuo luokitellaan. Saatuaan luokittelijalta tietovuolle luokan, kontrolleri laskee Yen-K-Lyhyintä polkua -algoritmilla tietovuolle K kappaletta parasta käytettävissä olevaa reittiä ja valitsee niistä sopivimman. Valinta tehdään reitin kustannuksen ja sovelluksen tietovuon luokan perusteella. Lopuksi kontrolleri asettaa tietovuota koskevat reitityssäännöt reitillä oleviin kytkimiin.

Tutkimuksessa todettiin luokiteltujen sovellusten latenssin pienentyneen ja läpimenovuon parantuneen. Vastaavasti havaittiin, että luokittelemattomille sovelluksille määritetyt reitit eivät pystyneet täyttämään niiden sovellusten resurssitarpeita. Verkkoperusteisella luokittelulla on käyttöä, kun halutaan parantaa käyttäjien laatukokemusta.

Luokittelua koneoppimisella voidaan käyttää reitityksessä myös yksityisyyden ja määräystenmukaisuuden (Compliance) toteuttamiseksi [5]. Tietoverkkojen laajentuessa niihin lisätään uusia laitteita, jolloin verkon laitekanta voi olla hyvin monipuolinen. Kaikki verkossa olevat laitteet eivät välttämättä aina täytä kaikkia viranomaisten asettamia vaatimuksia esimerkiksi tiedon yksityisyyden salaamisesta. Jotta tietovuon määräysten mukainen siirtäminen onnistuu, on verkon kyettävä tunnistamaan luottamusta vaativat tietovuot ja kyettävä reitittämään ne sellaisten verkkolaitteiden kautta, jotka täyttävät tietovuon yksityisyydelle asetetut määräykset.

Sensitiivisen tiedon siirtäminen tiettyä samaa reittiä pitkin synnyttää tietoturvariskin. Reitti voi joutua palvelunestohyökkäyksen kohteeksi tai jokin reitillä oleva

pahansuopa laite voi pystyä keräämään paketeista tietoa ja hyödyntää sitä. Tämän puutteen korjaamiseksi tarvitaan dynaamista reititystä.

Tutkitussa menetelmässä Ohjelmisto Määritellyn Verkon (SDN) kontrolleri ensin analysoi datapaketit, klusteroi ne K-means -algoritmilla ja luokittelee ne niiden riskitason perusteella. Riskitason määrittelyyn se käyttää HIPAA (Health, Insurance, Portability and Accountability Act) riskiparametreja. Seuraavaksi menetelmä käyttää Ant Colony Optimization (ACO) -algoritmia reaaliaikaiseen reititykseen. ACO -algoritmi on valittu, koska se on laskenta-ajallisesti vaatimattomampi, kuin vastaavat perinteiset verkkoalgoritmit. Tutkijoiden toteavat menetelmän mahdollistavan SDN verkon yksityisyyden hallinnan ja määräystenmukaisuuden.

7.3 Vertailu

Koneoppimista voidaan hyödyntää reitityksessä korvaamalla raskasta laskentaa vaativat heuristiset algoritmit nopeammilla koneoppimismenetelmillä, jotka jäljittävät niiden toimintaa. L. Yanjun et al. [40] ja Azzouniet al. [2] ovat tutkimuksissaan korvanneet heuristiset algoritmit niitä matkivilla neuroverkoilla.

Molemmissa tutkimuksissa havaittiin, että reitityksen tehokkuuden pysyvän lähes samana kuin korvattavalla heuristisella algoritmilla lasketussa reitityksessä. Molemmissa tutkimuksissa reitityksen laskemiseen meni vain murto-osa verrattuna heuristisella algoritmilla laskettuun. L. Yanjun et al. [40] tutkimuksessa laskenta-aika lyheni alle viidestuhannesosaan ja Azzouniet al. [2] tutkimuksessa laskenta-aika pieneni neljäsosaan.

Erot laskenta-aikojen muutoksissa voivat johtua eroissa alkuperäisten heuristisen algoritmien monimutkaisuuksissa, niiden toteutuksissa sekä neuroverkkojen toteutuksista. Tutkimusten neuroverkoissa oli eroja rakenteissa. L. Yanjun et al. [40] neuroverkko koostui vain yhdestä piilokerroksesta, jossa oli 600 - 2000 solmua, kun taas Azzouniet al. [2] neuroverkossa oli kuusi piilokerrosta, joissa kussakin oli sata solmua.

Vaikka opetusaikakin vaihtelee verkkojen rakenteen mukaan, sillä ei ole suurta merkitystä käytännön soveltamisen kannalta, koska opetus voidaan tehdä ennen kuin mallia käytettään. Jatkoa ajatellen olisi mielenkiintoista perehtyä neuroverkkojen rakenteen vaikutusta reititysten laskenta-aikoihin ja reitityksen tehokkuuteen.

A. Valadarsky et al. [36] havaitsi, että dynaamisesti muuttuvan liikenteen reititys ohjatun oppimisen menetelmillä on tehotonta. Yhtenä vaihtoehtona ongelmaan on vahvistusoppiminen, jonka tehokkuutta tutkittiin vertaamalla täysin kytkettyä neuroverkko (FCN), kolmeen eri menetelmää, jotka eivät perustuneet koneoppimiseen. Tutkimuksessa havaittiin, että varsinkin tiheässä verkossa vahvistusoppiminen on tehokkaampi kuin verrokit ja harvoissakin verkoissa se oli lähellä parasta verrokkia.

Tietoverkkojen reititystä voidaan tehostaa luokittelemalla liikennevirrat ja reitittämällä eri liikenneluokat niiden ominaisuudet huomioiden. Pasca et al. [31] vertasi Naiivia Bayesia, Päätöspuuta, Bayesilaista verkkoa ja Tukivektorikonetta liikenteen

luokittelussa tavoitteena parantaa tietoverkon käyttökokemusta. Parhaiten vertailussa pärjäsi päätöspuu, jota sitten sovellettiin tietoliikennevoiden luokitteluun reititystä varten. Tutkimuksessa havaittiin luokitellun liikenteen latenssin pienentyneen ja läpimenovuon suurentuneen.

Siirrettävä tieto voi joskus sisältää luottamuksellista tietoa, jonka siirto edellyttää laitteita, jotka täyttävät tällaisen tiedon siirtoon liittyvät vaatimukset. Kaikkea liikennettä ei ole järkevää varmuuden vuoksi reitittää vaatimukset täyttävien laitteiden kautta, jolloin ruuhkaantuisivat helposti. Budhraja et al. [5] on tutkinut tietoliikenteen luokittelua koneoppimisen avulla, niin että liikenne reititetään sellaisten laitteiden kautta, jotka täyttävät liikenteelle asetetut vaatimukset.

Tutkimuksessa Ohjelmisto Määritellyn Verkon (SDN) tietoliikenne klusteroitiin K-means -menetelmällä viiteen riskiluokkaan, joita hyödynnettiin reityksessä turvaamaan tiedon yksityisyys. Johtopäätöksenä todettiin, että menetelmä täyttää yksityisyyden ja määräyksenmukaisuuden reititykselle asettamat vaatimukset.

8 Resurssien ja kuormituksen hallinta

Tietoverkot koostuvat laitteista, joiden avulla verkon käyttäjille tuotetaan palveluita, kuten tiedonsiirtoa, tallennus- ja laskentakapasiteettia. Monilla toimijoilla on käytössä organisaation sisäiset palvelut, mutta tietoliikenteen nopeuduttua on kustannustehokkaampaa tuottaa palvelut isoissa yksiköissä datakeskuksissa.

Datakeskuksissa on useita palvelimia, paljon tallennuskapasiteettia ja nopeat tietoliikenneyhteydet niin palvelimien välillä kuin ulospäinkin. Datakeskukset kuluttavat paljon energiaa, joten niiden sijoittamisessa on huomioitava varman ja edullisen energian saanti. Koska palvelimet kuluttavat paljon sähköä, ne myös tuottavat hukkalämpöä, joka on laitteiden toiminnan turvaamiseksi siirrettävä pois palvelinsaleista. Vaikka joissain tapauksissa syntynyttä lämpöä voidaan hyödyntää esimerkiksi lämmitykseen, niin yleisesti ottaen jäähdyttäminen on kuluerä.

Datakeskuksissa monet palvelut on toteutettu virtuaalikoneilla, jolloin samalla fyysisellä palvelimella voi samanaikaisesti olla useita virtuaalikonetta. Virtuaalikoneita on helppo monistaa, käynnistää, siirtää ja sammuttaa palvelutarpeen mukaan, mikä tekee niistä kustannustehokkaan tavan skaalata palveluita. Virtualisointitekniikan kehittyminen on lisännyt erilaisten palveluiden toteuttamista pilvipalveluna.

Pilvipalvelut toteutetaan datakeskuksissa, joista asiakkaat vuokraavat tarpeittensa mukaan palveluita. Palveluita on mahdollista vuokrata aina laitteistopalveluista kuten palvelimista, tallennus- ja laskentakapasiteetista, alustoihin ja ohjelmistoihin. Pilvipalveluiden etuna on keskittämisen tuoma säästö, joka syntyy pienemmistä palvelu kohtaisista laitteistokuluista ja siitä, että tarvitaan vähemmän ylläpitohenkilöstöä. Pilvipalvelut sijaitsevat Internetissä, jolloin työn tekeminen ei ole enää sidottuna paikkaan vaan työ voidaan tehdä mistä vain tietoliikenneyhteyden päästä.

8.1 Taustaa

Datakeskukset ovat volyymipalveluita, joissa määrä tarkoittaa paitsi suuria tuloja myös isoja kustannuksia. Jotta datakeskuksen toiminta olisi tehokasta on tarpeellista optimoida käytettäviä resursseja tarvittavien laitteistojen minimoimiseksi ja kulutettavan energian vähentämiseksi. Energian kulutusta voidaan pienentää siirtämällä virtuaalipalvelimia fyysisiin palvelimiin niin, että ylimääräiset fyysiset palvelimet voidaan sammuttaa.

Datakeskuksissa on usein paljon erilaisia käyttäjiä erilaisilla käyttöprofiileilla, joten myös resurssien tarve muuttuu jatkuvasti. Resurssien tilan jatkuvan muuttumisen takia ei niitä voida optimoida tehokkaasti perinteisillä menetelmillä, vaan tarvitaan menetelmiä, jotka pystyvät ennustamaan mitkä ovat resurssien tarpeet tulevaisuudessa. Koneoppimisen voimakas kehittyminen on tuonut sen myös datakeskusten resurssien hallinnan tehostamiseen.

Datakeskusten resurssit koostuvat laskentakapasiteetista eli CPU-ajasta, keskusmuistista, tallennusmuistista, tietoliikenteen kaistaleveydestä jne. Resurssienhallinnalla tarkoittaa näiden allokointia eri palveluille niiden tarpeiden mukaan. Palveluilla on toisistaan poikkeavia vaatimuksia, jotka palveluntarjoaja pyrkii toteutumaan käytettävissä olevien resurssien mukaan.

Datakeskukseen tulevaa liikennettä voidaan jakaa eri palvelimille eri tavoin, tavoitteena on yleensä tulevan kuormituksen tasapainotus eri resurssien kesken. Jos jotakin palvelua kuormitetaan paljon voidaan datakeskuksessa monistaa se, kopioimalla sen virtuaalikone toisille palvelimille ja näin jakaa siihen kohdistunut kuormitus useammalle palvelimelle. Tulevat pyynnöt voidaan esimerkiksi ohjata round-robin -periaatteella niin, että kullekin palvelimelle välitetään vuorotellen tulevat pyynnöt. Pyynnöt voidaan myös ohjata palvelimille niiden sen hetkisen kuormituksen mukaan niin, että tuleva pyyntö ohjataan palvelimelle, jonka kuormitus on vähäisintä.

Palvelu voidaan toteuttaa useammassa maantieteellisesti eri paikassa sijaitsevassa datakeskuksessa. Tällöin pyynnöt voidaan ohjata siihen datakeskukseen, joka mahdollistaa pienimmän vasteajan asiakkaalle.

Monet resurssien hallinnointimenetelmien tehostamiset keskittyvät energiankulutuksen minimointiin, koska energiankulutus aiheuttaa ison osan niistä datakeskuksen käyttökuluista, joihin voidaan vaikuttaa (2008: 15 % kokonaiskuluista [23]). Energiankulutusta voidaan pienentää käyttämällä uudempia paremman hyötysuhteen omaavia laitteita tai operoimalla käytettäviä laitteita tehokkaasti. Resurssienhallintaan datakeskuksissa on käytettävissä useita menetelmiä, mm. virtuaalikoneiden sijoittelu ja tehtävien aikataulutus.

Koneoppimista voidaan hyödyntää monin tavoin datakeskuksissa, kuten voidaan havaita Demirci et al. katsauksesta [13], jossa on tarkasteltu koneoppimista pilvilaskennassa. Yleensä tavoitteena on kustannusten alentaminen energian kulutusta pienentämällä. Tähän tavoitteeseen pyritään monenlaisilla menetelmillä: CPU-taajuuden säädöllä, älykkäällä aikatauluttamisella, kuormituksen ennustamisella, lämmönjakautumisen hallintaan perustuvalla sijoittelulla, kuormituksen luokittelulla ja pyyn-

töjen ennustamisella, jotta tarpeettomat palvelimet voidaan asettaa lepotilaan.

Koneoppimista käytetään resurssienhallinnassa pääasiassa tulevien resurssitarpeiden ennustamiseen. Usein tavoitteena on kulutettavan energian minimointi, johon ratkaisua haetaan optimoimalla aktiivisten palvelinten lukumäärää ennakoidun kulutustilanteen mukaisesti. Energiaa säästyy, kun tarpeettomia palvelimia sammutetaan tai asetetaan lepotilaan. Muita vaikuttavia tekijöitä, jotka on otettava huomioon optimoitaessa parasta ratkaisua, ovat mm. käytössä olevien laitteiden energiatehokkuus ja asiakkaiden kanssa kuhunkin palveluun liittyvät palvelutasosopimukset (Service Level Agreement, SLA).

8.2 Menetelmät

Vasić et al. tutkivat resurssienhallintajärjestelmää DejaVu [37], joka oppii aikaisempien allokointien tuloksista. Menetelmän ideana on tehostaa resurssien uudelleenallokointia kuormituksen muuttuessa. Aikaisemmat huippumenetelmät perustuivat analyyttisiin malleihin tai eri allokointien kokeilemisiin. Molemmat tavat tuottavat paljon ylimääräistä työtä kuormitustilanteen muuttuessa.

Tutkittavan menetelmän ideana on tunnistaa eri kuormitustilanteet ja luokitella ne pieneen määrää kuormitusluokkia, joita varten menetelmän on evaluoitava erilaisia resurssien allokoinnit. Käytettäessä järjestelmää kuormitustilanteen muuttuessa menetelmä luokittelee kuorman ja hakee DejaVu-välimuistista vastaavalla tunnusmerkistöllä olevaa resurssien allokointia. Jos vastaava allokointi löytyi, menetelmä käyttää sitä, muutoin se kutsuu palvelun, joka uudelleen konfiguroi allokoinnin. Menetelmä käyttää koneoppimista luokittelemaan kuormitukset.

Tutkimuksen tavoitteena oli menetelmän testaus eikä koneoppiminen, joten tutkijat käyttivät valmista WEKA -ohjelmistoa [24], eikä käytettyjä luokittelijoita nimetty tai niiden suorituskykyä kuormitustilanteiden luokitteluun tutkittu. Menetelmän todettiin oleva kymmenen kertaa aiempia menetelmiä nopeampi ja vähentävän resurssien uudelleenallokointiin liittyvää työtä 60 %:lla.

Resurssien allokoinnissa energian käytön minimoimiseksi on tärkeää tietää etukäteen mitä resursseja milloinkin tarvitaan. Koska datakeskuksiin tulevat pyynnöt ovat satunnaisia niin ei ole mahdollista tietää todellisia resurssitarpeita vaan joudutaan varautumaan tulevaan kuormitukseen ylimääräisellä kapasiteetilla. Aikaisemmat menetelmät resurssien käytön optimoimiseksi eivät ole ottaneet huomioon mm. resurssien käynnistymiseen ja sammumiseen käytettyjä aikoja, jolloin dynaamisesti muuttuva kuormitus aiheuttaa niille ongelmia.

Useita ennustavia menetelmiä on tutkittua ja ne voidaan jakaa diskreetteihin ja stokastisiin menetelmiin. Diskreetit menetelmät käyttävät kiinteitä käynnistys- ja sammumisaikoja, mutta todellisuudessa tämä ei ole pätevä oletus, koska esimerkiksi levyn välimuistin täyttöaste saattaa vaikuttaa siihen, kuinka pian levy voidaan sammuttaa. Stokastisissa menetelmissä myös järjestelmän nykyinen tila vaikuttaa tulokseen ja eri ajat ovat tilastollisesti jakautuneita.

Prevost et al. [32] on tutkinut koneoppimisen soveltamista pilvipalvelun datakeskuksen tulevan kuormituksen ennustamiseen, jota voidaan sitten hyödyntää optimaalisemman allokoinnin toteuttamiseksi. Tutkimuksessa tutkittiin stokastista menetelmää käyttäen neuroverkkoja ja autoregressiivista Weiner-suodatinta.

Datana tutkimuksessa käytettiin NASA:n ja EPA:n WWW-palvelimien HTTP -lokitietoja (http://ita.ee.lbl.gov/html/traces.html). Näytteet ovat tiettynä sekunnin aikana palvelimelle tulleiden pyyntöjen lukumäärä. Tutkimuksessa tutkitaan myös aikavälin pituuden vaikutusta ennustamisen tarkkuuteen yhdistämällä uudella aikavälillä olevat näytteet yhteen.

Neuroverkkona tutkimuksessa oli monikerros perseptroni, jonka kokoa ei artikkelissa mainita, jonka takia ei voida arvioida neuroverkon koon mahdollista vaikutusta menetelmän tarkkuuteen. Neuroverkon tuloja ovat peräkkäiset näytteet, joiden avulla ennustetaan tulevien pyyntöjen lukumäärä, toisin sanoen tuleva kuormitus. Neuroverkko opetettiin takaisinvirtausalgoritmilla.

Autoregressiivisen Weiner-suodatimen kanssa käytettiin näytteiden sekoittamista (dithering), jotta aineistossa esiintyvät numeeriset poikkeamat pehmenisivät. Ennustamista varten käytetään 65 näytteen dataikkunaa, joista lasketaan optimaaliset kertoimet FIR (Finite Impulse Response) -suodattimelle.

Menetelmien tarkkuuden arviointiin käytettään neliövirhesummaa (Mean Squared Error, MSE) ja sen neliöjuurta (Root Mean Squared Error, RMSE). Tutkimuksessa oltiin kiinnostuneita myös ennustusvälin pituuden vaikutuksesta tulokseen, jota tutkittiin tekemällä mittaukset usealla aikavälillä alkaen yhdestä sekunnista ja päätyen 90 sekunttiin.

Molemmat tutkitut menetelmät onnistuvat hyvin ennustamaan tulevaa kuormitusta, mutta Autoregressiivinen on parempi, esimerkiksi 90 sekunnin aikavälillä NASA-aineistolla sen RMSE on kahdeksasosa neuroverkon vastaavasta.

Toinen merkittävä tulos tutkimuksesta oli molempien menetelmien ennustuskyvyn käänteinen lineaarinen riippuvuus ennustusaikavälin kasvuun. Tämän avulla voidaan päätellä millä varmuudella pilven hallinointijärjestelmä voi hallita optimaalisen operoinnin vaatimat tilan muutokset, jolloin on helpompi saavuttaa tavoitteena oleva optimaalinen energiankulutus, kun samanaikaisesti täytettään voimassa olevat palvelutasosopimukset.

Isoissa tietojärjestelmissä on koneoppimiseen perustuvia järjestelmiä poikkeamien havaitsemiseen (Network Intrusion Detection System, NIDS). Nämä järjestelmät toimivat perinteisesti keskitetysti, mutta näin big datan aikakaudella, kun tietomäärät kasvavat, eivät keskitetyt ratkaisut aina enää riitä vaan tarvitaan hajautettuja ratkaisuja.

Koneoppimismenetelmät ovat perinteisesti olleet keskitettyjä ja siten niiden skaalautuvuus on ollut huono. Joitakin hajautettuja algoritmeja on kehitetty [25], joissa eri instanssit, erillään suoritettujen osaratkaisujen jälkeen, synkronoivat tulokset keskenään. Lisäksi koneoppimistehtävä voidaan ratkaistaan useasta eri instassista koostuvasta koneoppimiskoneesta, jossa kukin itsessään keskitetty instanssi ratkai-

see erikseen osajoukon kokonaistehtävästä, eikä tätä tulosta sitten enää jaeta muiden instanssien kanssa.

Useamman instanssin käytön ideana on jakaa kuormitusta useammalle resurssille. G. Frishman et al. [20] tutkivat kuormanjakamista koneoppimiseen perustuvassa hajautetussa väärinkäytön havaitsemisjärjestelmässä. Heidän ideana on jakaa kuormitusta liikenteen samankaltaisuuden perusteella, niin että samankaltaiset liikennevuot menevät samalle havaitsemisinstanssille. Menetelmän tavoitteena on maksimoida väärinkäytön havaitsemisen tehokkuus havaitsemisjärjestelmässä ja samanaikaisesti tasapainottaa kuormitusta instanssien kesken.

Tutkimuksen yhtenä vertailukohtana käytettiin keskitettyyn ratkaisuun perustuvaa väärinkäytön havaitsemista. Lisäksi menetelmän vertailua varten arvioitiin myös kahta hajautettua ratkaisua, joissa toisessa kuormitus tasapainotettiin round-robinilla ja toisessa kuormitus jaettiin tasaisesti kaikille instansseille (unified random distribution). Nämä menetelmät jakavat kuormituksen tasan kaikille instansseille, mutta soveltuvat huonosti koneoppimispohjaiseen väärinkäytösten havaitsemiseen, koska tällöin koneoppimisongelman kokonaistehokkuus heikkenee.

Tutkimuksessa käytettiin NSL-KDD -aineistoa [35], jossa kukin tietue kuvaa sarjaa paketteja TCP-sessiossa. Aineiston ominaispiirteet muodostuvat mm. tilastoarvoista kaikista yhteyksistä samaan kohteeseen kahden sekunnin aikana. Tietueet on luokiteltu Normaaliksi, jossa se ei sisällä väärinkäyttöä. Väärinkäyttöluokat ovat Pääsynesto (DOS), Skannaus (PROBE), Luvaton etäyhteys (R2L, remote-to-local) ja Luvaton paikallisyhteys (U2R, user-to-root). Lisäksi koska käytettiin vain numeerisia ominaispiirteitä jouduttiin aineiston esikäsittelyssä kategorisia ominaispiirteitä poistamaan tai muuttamaan one-hot enkoodauksella numeeriseksi niin, että kutakin kategoriaa vastaa oma ominaispiirre, joka saa arvon yksi, kun instanssi kuuluu kyseiseen kategoriaan, muutoin ominaispiirre saa arvon nolla. Vastaavasti osa numeerista ominaispiirteistä skaalattiin minimi- ja maksimiarvojen mukaan.

Erilaisista klusterointialgoritmeista, ominaispiirrejoukoista ja ominaispiirteiden esikäsittelyistä koostettiin klusterointimalleja, joista arvioitiin ne, joiden katsottiin sopivan tehtävään. Klustereiden määrä rajoitettiin kolmen ja yhdeksän väliin sekä suurimman ja pienimmän klusterin maksimisuhteeksi asetettiin 25.

Lopulta tutkimuksessa arvioitiin kolmea eri klusterointimallia, jotka kaikki perustuivat K-means -menetelmään erilaisilla ominaispiirrejoukoilla. Näiden kolmen klusterointimallin suurimman ja pienimmän klusterin suhteet ovat 13 ja 16 välillä, kun vertailukohdat jakoivat kuormituksen niin, että vastaavat suhdeluvut olivat lähes yksi. Erot klustereiden suhteellisissa koissa on isohko ja tutkijat toteavatkin, että kuormituksen parempi tasapainottaminen olisi hyvä menetelmän kehittämiskohta.

Tutkimuksessa menetelmiä mitattiin mm. F1 -mitan avulla. Kyseinen mittari oli valittu, koska se huomioi sekä tarkkuuden (precision) ja herkkyyden (recall) ja on hyvin lähellä täsmällisyyttä.

Tutkimuksessa [20] verrattiin myös klusterointiin pohjautuvan kuorman tasapainottamisen vaikutusta eri väärinkäytösten havaitsemiseen käytettyjen koneoppimisme-

netelmien suorituskykyyn. Osan menetelmistä suorituskyky parani hyvin merkittävästi ja osan vähemmän merkitsevästi, kun kuormitus jaettiin klusterointiin pohjautumalla menetelmällä. Parhaiten pärjäsivät päätöspuut ja satunnaismetsä, joiden F1 -mitan arvo nousi 0,8:sta yli 0,9:ään. heikoimmin pärjäsivät tukivektorikone ja lähinnaapuri-menetelmä, joiden nousi vain hieman. Näiden välissä oli vielä monikerrosperseptroni.

Keskitetyn ratkaisun oletettiin olevan lähtökohtaisesti tarkempi kuin hajautetut ratkaisut, koska sillä on kokonaiskuva tehtävästä, mutta tutkittava menetelmä, jossa näytteet ensin klusteroidaan antaa paremman tuloksen. Tutkijoiden näkemyksen mukaan klusterointi tavallaan tuo yhden lisätason varsinaiselle luokittelijalle ja siten parantaa lopullisen luokittelun tulosta.

Klusterointiin perustuva kuormituksen jakaminen parantaa erityisesti harvinaisia luokkia, esimerkiksi päätöspuulla U2R-luokan AUC (Area Under Curve) parani 10%. Harvinaiset luokat ovat haastavia opetettavia, koska niissä näytteiden osuus on pieni, jolloin niitä herkästi luokitellaan muihin luokkiin ja siten luokittelutarkkuus heikkenee.

8.3 Vertailu

Koneoppimista voidaan hyödyntää resurssienhallinnassa monella tapaa mm. luokittelemalla kuormitusta tai liikennettä sekä ennustamalla tulevaa liikennettä.

Vasić et al. tutkivat resurssienhallintajärjestelmää DejaVu [37], joka luokittelee kuormitustilanteita. Menetelmän todettiin oleva kymmenen kertaa nopeampi kuin aiemmat menetelmät vähentävän resurssien uudelleenallokointiin liittyvää laskentaa 60%. Tutkimus keskittyi kehitetyn konseptin testaamiseen, eikä antanut tarkempaa tietoa käytetyistä koneoppimismenetelmistä.

Prevost et al. [32] sovelsivat monikerrosperseptronia ja autoregressiivista Weiner-suodatinta ennustamaan datakeskukseen tulevaa liikennettä. Molemmat menetelmät antoivat käyttökelpoisia tuloksia, joiden avulla voidaan tehostaa resurssien allokointia datakeskuksissa. Tutkijat havaitsivat myös käänteisen lineaarisen riippuvuuden ennustusaikavälin pituuden ja ennustuksen tarkkuuden kanssa. Tätä tietoa voidaan hyödyntää ennustettaessa liikennettä ennustusaikavälien sisällä.

Koneoppimista voidaan hyödyntää myös kuormituksen tasapainottamiseen. G. Frishman et al. [20] tutki hajautettua menetelmää väärinkäytösten havaitsemiseen datakeskuksen liikenteessä. Menetelmässä datakeskukseen tuleva liikenne klusteroitiin K-means menetelmällä ja klusterit jaettiin eri instansseihin luokiteltaviksi. Luokittelussa kokeiltiin useita menetelmiä, joista parhaimmaksi osoittautuivat päätöspuu ja satunnaismetsä.

Tutkimuksessa todettiin hajautetun väärinkäyttöjen havaitsemisen toimivan paremmin kuin keskitetyn havaitsemisen. Erityisesti tämä nähtiin pienten luokkien kohdalla. Menetelmän todettiin tasapainottavan kuormitusta, mutta erot klustereiden kesken olivat yli kymmenkertaiset. Tutkijat totesivatkin kuormituksen tasapainot-

tamisen parantamiseksi tarvitaan lisää tutkimusta.

9 Yhteenveto

Internetin myötä tiedosta ja sen siirrosta on tullut tärkeä osa ihmisten elämää. Tämä ilmenee niin tärkeiden infrastruktuurin hallinnassa, liiketoiminnassa kuin ihmisten vapaa-ajassakin. Tietoverkot paitsi siirtävät dataa myös tuottavat dataa, jota käytetään tietoverkkojen hallinnassa. Internetin liikenne on kuitenkin kasvanut niin suureksi, että sen hallinta manuaalisesti on tehotonta, siksi tarvitaan automaattisia menetelmiä.

Koneoppiminen on eräs tekoälyn osa-alue, jonka avulla suurten tietomäärien analysointi voidaan automatisoida. Koneoppiminen koostuu joukosta menetelmiä, joissa järjestelmä hyödyntää suurta määrää dataa suorittaakseen jonkin tehtävän. Koneoppimismenetelmät on tyypillisesti jaettu ohjattuun oppimiseen, ohjaamattomaan oppimiseen ja vahvistusoppimiseen.

Menetelmät eroavat toisistaan siinä miten ne käyttävät dataa. Ohjatussa oppimisessa data sisältää syötedatan lisäksi sitä vastaavat tulosdatan, jolloin järjestelmä voidaan opettaa miten syötedata vaikuttaa tulosdataan. Ohjaamattomassa oppimisessa syötedatasta halutaan oppia siinä piilossa oleva tieto esimerkiksi kuinka monesta eri lailla jakautuneesta aineistosta data on muodostunut, jolloin voidaan käyttää klusterointia. Vahvistusoppimisessa toimija havainnoi järjestelmän tilaa ja suorittaa sen perusteella toiminnon, joka pyrkii saattamaan järjestelmän tilan kohti tavoitetta. Järjestelmä palauttaa toiminnon jälkeen toimijalle uuden tilansa sekä palkkion sen perusteella kuinka hyvin toiminto auttoi järjestelmää kohti tavoitettaan.

Internetin ja muidenkin tietoverkkojen tehokas ja luotettava toiminta koostuu useasta eri tehtävästä, joiden toiminnan tehostamiseen on hyödynnetty koneoppimisen menetelmiä. Tutkimuksessa läpi käydyt tehtävät on jaettu liikenteen luokitteluun, suorituskyvyn analysointiin, poikkeamien havaitsemiseen, reititykseen sekä resurssien ja kuormituksen hallintaan. Taulukosta 3 näkyy, että erilaisia menetelmiä on käytetty eri tehtäviin monipuolisesti samoin kuin se, että kaikkiin tehtäviin on käytetty myös useampaa menetelmää.

Luokittelu on eräs ohjatun oppimisen perustapaus ja siihen on useita erilaisia menetelmiä. Luokittelu onkin menetelmä, jota voidaan soveltaa kaikkiin tehtäviin joko suoraan tai välillisesti, esimerkiksi poikkeamat useimmin havaitaan luokittelemalla verkkotoiminnot normaalitapauksiin ja erilaisiin poikkeustapauksiin tai liikennettä reititetään luokittelemalla se erilaisiin kategorioihin sen tarpeiden mukaan. Ohjatun oppimisen toista yleistä menetelmää, regressiota, ei käytetty tutkimuksen aineistossa mihinkään.

Tutkimuksen aineistossa toinen varsin yleinen perustapaus on ohjaamattoman oppimisen klusterointi, jossa siinäkin aineistot ryhmitellään erilaisiin kategorioihin, joita sitten käsitellään eri tavoin. Muita ohjaamattoman oppimisen menetelmiä hyödynnettiin, kuten esimerkiksi puuttuvien arvojen imputointia datan esikäsittelyssä,

Taulukko 3: Menetelmät eri tehtävissä.

	Tehtävät					
Menetelmät	*Liikenteen luokittelu*	*Suorituskyvyn analysointi*	*Poikkeamien havaitseminen*	*Tietoverkon turvallisuus*	*Reititys tietoverkossa*	*Resurssien ja kuormituksen hallinta*
Naiivi Bayes	x	x	x	x	x	
SVM	x	x	x		x	x
Bayeslainen verkko	x				x	
Päätöspuu		x	x	x	x	
Satunnaismetsä		x	x	x		
K-means	x	x				x
DBSCAN	x					
Autoclass	x					
Vahvistusoppiminen					x	
Neuroverkko	x	x	x	x	x	x
Syväoppiminen	x					

mutta näitä menetelmiä ei varsinaisesti tutkittu.

Vahvistusoppimista hyödynnettiin reitityksessä. Tietoverkoissa liikennevirrat alkavat ja päättyvät jatkuvana ketjuna, jolloin reititin joutuu koko ajan hakemaan uuteen tilanteeseen parhaiten sopivinta reittiä. Tähän varsin dynaamiseen tehtävään vahvistusoppiminen sopii hyvin.

Eräs koneoppimisen ongelmista tietoverkoissa on tietoverkkojen hajautuneisuus. Koneoppiminen on perinteisesti toteutettu keskitetyillä ratkaisuilla, joissa yhtä yhtenäistä aineistoa käytettään opettamaan järjestelmä. Tietoverkoissa dataa syntyy ja sitä käytettään hajautuneesti eri verkkolaitteissa. Vaikka koneoppimisen hyödyntäminen tehokkaasti tietoverkossa onkin haastavaa, niin G. Frishman et al. [20] käyttää hajautettuja laskentaresursseja klusteroimalla ensin aineiston ja sitten jakamalla klusterit eri laskentaresursseille sisällön luokittelua varten.

Erilaisissa luokittelutehtävissä vahvoja menetelmiä olivat päätöspuut ja neuroverkot, jotka ovat keskenään varsin erilaisia menetelmiä. Päätöspuut on tehokas, koska se valikoi merkitsevimmät ominaispiirteet, lisäksi sen tekemät päätelmät ovat help-

poja tulkita verrattuna moniin muihin menetelmiin. Päätöspuut ovat ahne algoritmi, eli se etenee kussakin vaiheessa kohti lopullista ratkaisua valitsemalla sen hetkisen parhaan vaihtoehdon, tällöin ei välttämättä optimaaliseen ratkaisuun vaan laskenta voi päätyä paikalliseen minimiin. Neuroverkot vaativat paljon dataa tuottaakseen parempia tuloksia kuin ns. perinteiset koneoppimismenetelmät. Suuri datamäärän käsittely edellyttää myös riittäviä laskentaresursseja, jotta neuroverkot olisivat käyttökelpoisia. Toisin kuin päätöspuissa algoritmin tekemät päätelmät ovat vaikeasti tulkittavia.

Klusteroinnissa tutkimuksessa esiin nousi paljon käytetty menetelmä K-means. K-means on paitsi yksinkertainen algoritmi, niin se on myös riittävän tehokas, minkä vuoksi se onkin varsin yleinen menetelmä myös tietoverkkojen tehtäviin liittyvissä klusteroinneissa.

Syväoppiminen on eräs koneoppimisen viime aikoina eniten tutkittu menetelmä. Syväoppiminen vaatii suoriutuakseen hyvin paljon dataa ja suuren datamäärän käsittely puolestaan vaatii paljon laskentaresursseja, joiden lisääminen verkkolaitteisiin ei ole aina perusteltua, mutta joissakin valikoiduissa sovelluksissa lisätyt laskentaresurssit saattavat olla perusteltuja. Syväoppimisen hyödyntämisestä tietoverkoissa on toistaiseksi vain vähän tutkimusta, mutta menetelmien kehittymisen myötä myös sen soveltaminen tietoverkkoihin tarjoaa monia mahdollisuuksia tutkimukselle.

Vahvistusoppimisen hyödyntämisestä tietoverkoissa oli vähän tutkimuksia. Muuttuvissa olosuhteissa, kuten esimerkiksi tietoverkkojen reitityksessä, vahvistusoppimiselle voisi löytyä monia tutkimuskohteita.

Koneoppimisen yleistyessä myös todennäköisyys sen aiheuttamiin virheisiin kasvaa. Virheitä voivat tuottaa ongelmaan sopimattomat algoritmit, puutteellinen tai virheellinen data. Joissakin tapauksissa saattaa olla tahoja, jotka hyötyvät laitteiden virheellisestä toiminnasta ja siksi pyrkivät vaikuttamaan koneoppimisprosessiin.

Koneoppimismallin manipuloimiseksi on yleensä päästävä käsiksi opetusdataan tai prosessiin, joka tuottaa opetusdatan. Koneoppimismalleja voidaan manipuloida myös käyttövaiheessa esimerkiksi muokkaamalla niiden instanssien ominaispiirteitä, jotka halutaan luokitella väärään luokkaan. Koneoppimismenetelmien manipuloinnista on tehty tutkimuksia, kuten esimerkiksi Jagielski et al. [26], mutta tutkielmaa varten läpikäymässäni materiaalissa ei löytynyt esimerkkejä tietoverkoissa käytettyjen koneoppimismenetelmien manipuloinnista.

Tietoverkkojen luotettavuus ja tehokkuus on tärkeää digitalisoituneen yhteiskunnan toiminnan varmistamiseksi. Koneoppimisen menetelmät yleistyvät laajalti tietojärjestelmissä ja siten myös tietoverkoissa, jossa niitä käytetään mm. tietoverkkojen toiminnan tehostamisessa ja luotettavuuden varmistamisessa. Uusia menetelmiä otettaessa laajemmin käyttöön on syytä varmistaa, että ne toimivat luotettavasti. Koneoppimisen toiminnan virheettömyyden ja luotettavuuden parantamiseksi tietoverkoissa riittää jatkossakin tutkittavaa.

Lähteet

1 Azar, Yossi, Cohen, Edith, Fiat, Amos, Kaplan, Haim ja Racke, Harald: *Optimal Oblivious Routing in Polynomial Time.* Teoksessa *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, sivut 383–388, New York, NY, USA, 2003. ACM, ISBN 1-58113-674-9. http://doi.acm.org/10.1145/780542.780599.

2 Azzouni, A., Boutaba, R. ja Pujolle, G.: *NeuRoute: Predictive dynamic routing for software-defined networks.* Teoksessa *2017 13th International Conference on Network and Service Management (CNSM)*, sivut 1–6, Nov 2017.

3 Box, G. E. P.: *Science and Statistics.* Journal of the American Statistical Association, 71:791–799, 1976.

4 Breiman, Leo: *Bagging Predictors.* Mach. Learn., 24(2):123–140, elokuu 1996, ISSN 0885-6125. http://dx.doi.org/10.1023/A:1018054314350.

5 Budhraja, K. K., Malvankar, A., Bahrami, M., Kundu, C., Kundu, A. ja Singhal, M.: *Risk-Based Packet Routing for Privacy and Compliance-Preserving SDN.* Teoksessa *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, sivut 761–765, June 2017.

6 Casas, P., Soro, F., Vanerio, J., Settanni, G. ja D'Alconzo, A.: *Network security and anomaly detection with Big-DAMA, a big data analytics framework.* Teoksessa *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, sivut 1–7, Sept 2017.

7 Casas, P., Vanerio, J. ja Fukuda, K.: *GML learning, a generic machine learning model for network measurements analysis.* Teoksessa *2017 13th International Conference on Network and Service Management (CNSM)*, nide 00, sivut 1–9, Nov. 2017. doi.ieeecomputersociety.org/10.23919/CNSM.2017.8255998.

8 Casas, Pedro, D'Alconzo, Alessandro, Settanni, Giuseppe, Fiadino, Pierdomenico ja Skopik, Florian: *POSTER: (Semi)-Supervised Machine Learning Approaches for Network Security in High-Dimensional Network Data.* Teoksessa *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, sivut 1805–1807, New York, NY, USA, 2016. ACM, ISBN 978-1-4503-4139-4. http://doi.acm.org.libproxy.helsinki.fi/10.1145/2976749.2989069.

9 Casas, Pedro, D'Alconzo, Alessandro, Wamser, Florian, Seufert, Michael, Gardlo, Bruno, Schwind, Anika, Tran-Gia, Phuoc ja Schatz, Raimund: *Predicting QoE in cellular networks using machine learning and in-smartphone measurements.* Teoksessa *Ninth International Conference on Quality of Multimedia Experience, QoMEX 2017, Erfurt, Germany, May 31 - June 2, 2017*, sivut 1–6, 2017. https://doi.org/10.1109/QoMEX.2017.7965687.

10 Casas, Pedro, Fiadino, Pierdomenico ja D'Alconzo, Alessandro: *Machine-Learning Based Approaches for Anomaly Detection and Classification in Cellular Networks.* Teoksessa *Traffic Monitoring and Analysis - 8th International Workshop, TMA 2016, Louvain la Neuve, Belgium, April 7-8, 2016.*, 2016. http://dl.ifip.org/db/conf/tma/tma2016/tma2016-final50.pdf.

11 Casas, Pedro, Mazel, Johan ja Owezarski, Philippe: *Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge.* Computer Communications, 35(7):772 – 783, 2012, ISSN 0140-3664. http://www.sciencedirect.com/science/article/pii/S0140366412000266.

12 Chandola, Varun, Banerjee, Arindam ja Kumar, Vipin: *Anomaly Detection: A Survey.* ACM Comput. Surv., 41(3):15:1–15:58, heinäkuu 2009, ISSN 0360-0300. http://doi.acm.org.libproxy.helsinki.fi/10.1145/1541880.1541882.

13 Demirci, M.: *A Survey of Machine Learning Applications for Energy-Efficient Resource Management in Cloud Computing Environments.* Teoksessa *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, sivut 1185–1190, Dec 2015.

14 Erman, Jeffrey, Arlitt, Martin ja Mahanti, Anirban: *Traffic Classification Using Clustering Algorithms.* Teoksessa *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, MineNet '06, sivut 281–286, New York, NY, USA, 2006. ACM, ISBN 1-59593-569-X. http://doi.acm.org.libproxy.helsinki.fi/10.1145/1162678.1162679.

15 Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T. ja Mizutani, K.: *State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems.* IEEE Communications Surveys Tutorials, 19(4):2432–2455, Fourthquarter 2017.

16 Fan, Z. ja Liu, R.: *Investigation of machine learning based network traffic classification.* Teoksessa *2017 International Symposium on Wireless Communication Systems (ISWCS)*, sivut 1–6, Aug 2017.

17 Fontugne, Romain, Borgnat, Pierre, Abry, Patrice ja Fukuda, Kensuke: *MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking.* Teoksessa *Proceedings of the 6th International COnference*, Co-NEXT '10, sivut 8:1–8:12, New York, NY, USA, 2010. ACM, ISBN 978-1-4503-0448-1. http://doi.acm.org.libproxy.helsinki.fi/10.1145/1921168.1921179.

18 Freund, Yoav ja Schapire, Robert E.: *Experiments with a New Boosting Algorithm.* Teoksessa *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, sivut 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc., ISBN 1-55860-419-7. http://dl.acm.org.libproxy.helsinki.fi/citation.cfm?id=3091696.3091715.

19 Freund, Yoav ja Schapire, Robert E: *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*. Journal of Computer and System Sciences, 55(1):119 – 139, 1997, ISSN 0022-0000. http://www.sciencedirect.com/science/article/pii/S002200009791504X.

20 Frishman, Gal, Ben-Itzhak, Yaniv ja Margalit, Oded: *Cluster-Based Load Balancing for Better Network Security*. Teoksessa *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, Big-DAMA '17, sivut 7–12, New York, NY, USA, 2017. ACM, ISBN 978-1-4503-5054-9. http://doi.acm.org/10.1145/3098593.3098595.

21 Ghadiyaram, D., Pan, J. ja Bovik, A.C.: *LIVE Mobile Stall Video Database-II*. http://live.ece.utexas.edu/research/LIVEStallStudy/index.html, 2017.

22 Goodfellow, Ian, Bengio, Yoshua ja Courville, Aaron: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

23 Greenberg, Albert, Hamilton, James, Maltz, David A. ja Patel, Parveen: *The Cost of a Cloud: Research Problems in Data Center Networks*. SIGCOMM Comput. Commun. Rev., 39(1):68–73, joulukuu 2008, ISSN 0146-4833. http://doi.acm.org.libproxy.helsinki.fi/10.1145/1496091.1496103.

24 Hall, Mark, Frank, Eibe, Holmes, Geoffrey, Pfahringer, Bernhard, Reutemann, Peter ja Witten, Ian H.: *The WEKA Data Mining Software: An Update*. SIGKDD Explor. Newsl., 11(1):10–18, marraskuu 2009, ISSN 1931-0145. http://doi.acm.org.libproxy.helsinki.fi/10.1145/1656274.1656278.

25 Hu, W., Gao, J., Wang, Y., Wu, O. ja Maybank, S.: *Online Adaboost-Based Parameterized Methods for Dynamic Distributed Network Intrusion Detection*. IEEE Transactions on Cybernetics, 44(1):66–82, Jan 2014, ISSN 2168-2267.

26 Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C. ja Li, B.: *Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning*. Teoksessa *2018 IEEE Symposium on Security and Privacy (SP)*, sivut 19–35, May 2018.

27 Laan, Mark J. Van der, Polley, Eric C. ja Hubbart, Alan E.: *Super Learner*. Statistical Applications in Genetics and Molecular Biology, 6, 2007.

28 Mao, B., Fadlullah, Z. M., Tang, F., Kato, N., Akashi, O., Inoue, T. ja Mizutani, K.: *Routing or Computing? The Paradigm Shift Towards Intelligent Computer Network Packet Transmission Based on Deep Learning*. IEEE Transactions on Computers, 66(11):1946–1960, Nov 2017, ISSN 0018-9340.

29 Murphy, Kevin P.: *Machine Learning A Probabilistic Perspective*. The MIT Press, 2012.

30 Nguyen, T. T. T. ja Armitage, G.: *A survey of techniques for internet traffic classification using machine learning.* IEEE Communications Surveys Tutorials, 10(4):56–76, Fourth 2008, ISSN 1553-877X.

31 Pasca, S. T. V., Kodali, S. S. P. ja Kataoka, K.: *AMPS: Application aware multipath flow routing using machine learning in SDN.* Teoksessa *2017 Twenty-third National Conference on Communications (NCC)*, sivut 1–6, March 2017.

32 Prevost, J. J., Nagothu, K., Kelley, B. ja Jamshidi, M.: *Prediction of cloud data center networks loads using stochastic and neural models.* Teoksessa *2011 6th International Conference on System of Systems Engineering*, sivut 276–281, June 2011.

33 Sasisekharan, R., Seshadri, V. ja Weiss, S. M.: *Using machine learning to monitor network performance.* Teoksessa *Proceedings of the Tenth Conference on Artificial Intelligence for Applications*, sivut 92–98, Mar 1994.

34 Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael ja Moritz, Philipp: *Trust Region Policy Optimization.* Teoksessa Bach, Francis ja Blei, David (toimittajat): *Proceedings of the 32nd International Conference on Machine Learning*, nide 37 sarjassa *Proceedings of Machine Learning Research*, sivut 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. `http://proceedings.mlr.press/v37/schulman15.html`.

35 Tavallaee, M., Bagheri, E., Lu, W. ja Ghorbani, A. A.: *A detailed analysis of the KDD CUP 99 data set.* Teoksessa *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, sivut 1–6, July 2009.

36 Valadarsky, Asaf, Schapira, Michael, Shahaf, Dafna ja Tamar, Aviv: *Learning to Route.* Teoksessa *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, sivut 185–191, New York, NY, USA, 2017. ACM, ISBN 978-1-4503-5569-8. `http://doi.acm.org.libproxy.helsinki.fi/10.1145/3152434.3152441`.

37 Vasić, Nedeljko, Novaković, Dejan, Miučin, Svetozar, Kostić, Dejan ja Bianchini, Ricardo: *DejaVu: Accelerating Resource Allocation in Virtualized Environments.* SIGARCH Comput. Archit. News, 40(1):423–436, maaliskuu 2012, ISSN 0163-5964. `http://doi.acm.org.libproxy.helsinki.fi/10.1145/2189750.2151021`.

38 Wang, P., Lin, S. C. ja Luo, M.: *A Framework for QoS-aware Traffic Classification Using Semi-supervised Machine Learning in SDNs.* Teoksessa *2016 IEEE International Conference on Services Computing (SCC)*, sivut 760–765, June 2016.

39 Wolpert, David H.: *Original Contribution: Stacked Generalization.* Neural Netw., 5(2):241–259, helmikuu 1992, ISSN 0893-6080. `http://dx.doi.org.libproxy.helsinki.fi/10.1016/S0893-6080(05)80023-1`.

40 Yanjun, L., Xiaobo, L. ja Osamu, Y.: *Traffic engineering framework with machine learning based meta-layer in software-defined networks.* Teoksessa *2014 4th IEEE International Conference on Network Infrastructure and Digital Content*, sivut 121–125, Sept 2014.