Koneoppimismenetelmien käyttö aikasarjojen ennustamiseen

Pro Gradu -tutkielma
Eetu Leinonen
2506405
Matemaattisten tieteiden yksikkö
Oulun yliopisto
Kevät 2022

Sisältö

Tiivistelmä

Tutkimuksessa tarkastellaan eri koneoppimismenetelmien toimivuutta kahden rakennuksen sähkönkulutuksesta kerättyjen aikasarjojen arvojen ennustamisessa. Rakennusten sähkönkulutusta on pyritty ennustamaan useissa tutkimuksissa, mutta erityisesti kaupparakennusten sähkönkulutusta on ennustettu vähemmän kuin esimerkiksi toimistorakennusten sähkönkulutusta.

Tutkimuksen aineistona käytettiin kahdesta Oululaisesta ruokakaupasta kerättyä sähkönkulutustietoa vuosilta 2017 ja 2018. Ensimmäisen kaupan aineistoa on kerätty vuoden 2017 toukokuusta vuoden 2018 kesäkuuhun ja toisen kaupan aineiston kerääminen on aloitettu vuoden 2017 lokakuussa ja kerätty vuoden 2018 kesäkuuhun asti. Aineistoa on kerätty yhden minuutin resoluutiolla.

Aineiston ennustekykyä arvioitiin useiden mallien avulla. Valittuja malleja olivat keinotekoiset neuroverkot, tukivektoriregressio ja ARIMA. Jokaisesta malliperheestä sovitettiin aineistoon useita malleja ja jokaisen sovitetun mallin ennusteen tarkkuutta arvioitiin keskineliövirheen neliöjuurta (RMSE) käyttäen. Jokaisen mallin valintaan käytettiin aikasarjamenetelmiin sopivaa validointitapaa, jossa aineiston alkuosaa käytetään mallin opettamiseen ja loppuosaa mallin tarkkuuden mittaamiseen. Mallinvalinnassa jokaisen mallin hyperparametrit valittiin hilaetsinnän avulla.

Tutkimuksessa selvisi, että kaupan 1 aineistolle parhaiten toimiva malli on tukivektoriregressio ja lisäksi tavalliset eteenpäinsyöttävät neuroverkot toimivat hyvin osassa ennustustapauksista. Kaupassa 2 parhaiten toiminut malli oli myös tukivektoriregressio, mutta toisaalta ARIMA-malli sekä tavalliset eteenpäinsyöttävät neuroverkot toimivat hyvin. Suurin osa mallien ennustusvirheistä molemmissa kaupoissa sijoittuivat välille 8–20 %.

1 Johdanto

Tässä tutkielmassa tutkitaan koneoppimismenetelmien soveltuvuutta aikasarjojen ennustamiseen. Tutkittavia aikasarjoja ovat kahden kaupan sähkönkulutuksesta muodostetut aikasarjat. Aikasarjoja tutkitaan pääosin koneoppimisen menetelmin, mutta mukaan on otettu myös yksi hiukan perinteisempikin tilastollinen menetelmä, autoregressive integrated moving average -menetelmä (ARIMA). Koneoppiminen on tekoälyn osa-alue, jossa tilastollisia menetelmiä hyödyntäen pyritään antamaan tietokoneille taito havaita aineistosta uusia rakenteita ja säännöllisyyksiä ilman ihmisen väliintuloa. Koneoppimismenetelmiksi luetellaan perinteisesti esimerkiksi keinotekoiset neuroverkot (artificial neural network), vahvistusoppiminen (reinforcement learning) ja geneettiset algoritmit (genetic algorithms) (Marsland 2014).

Koneoppiminen, ja erityisesti syväoppiminen (deep learning) eli suurien ja monikerroksisten neuroverkkojen käyttäminen, on 1990-luvun jälkeen noussut tavalliseksi työkaluksi myös kaupallisissa sovelluksissa. Ensimmäiset keinotekoiset neuroverkot keksittiin jo 1950-luvulla, mutta puutteellisen laskentatehon takia ne eivät vielä silloin nousseet tärkeäksi työkaluksi. Kun 1990-luvulla tietokoneiden laskentateho kasvoi ja ryhdyttiin keräämään entistä suurempia määriä aineistoa, kasvoi myös syväoppimisen suosio ja nykyisin neuroverkkoja tutkitaan aktiivisesti ympäri maailmaa ja niitä hyödynnetään myös lukuisissa sovelluksissa (Goodfellow ym. 2016).

Tässä luvussa kuvataan lyhyesti kuinka sähkömarkkinat toimivat ja miten ne ovat tulevaisuudessa muuttumassa sekä sitä, miten koneoppimisen menetelmiä voidaan jaotella. Tämän jälkeen luvuissa 2–4 esitellään tutkielmassa käytettävät mallit. Luvussa 2 esitellään tukivektoriregressio, luvussa 3 esitellään erilaisia neuroverkkomalleja kuten konvoluutioneuroverkko ja takaisinkytkeytyvä neuroverkko. Luvussa 4 esitellään ARIMA-malliperheen rakenne. Luvussa 5 käydään läpi aineiston analysointimenetelmät sekä analyysien ohjelmistot ja periaatteet. Luvussa 6 esitellään malleista saadut tulokset, luvussa 7 esitetään tuloksien nojalla pohdintaa liittyen malleihin sekä aineiston rakenteeseen. Viimeisenä luvussa 8 pohditaan mahdollista tulevaa tutkimusta samaan aihepiiriin liittyen.

1.1 Sähkömarkkinat

Ilmastonmuutos ja sen tuomat uhat haastavat ihmiskuntaa muutoksiin toiminnassaan. Tämän takia Suomessakin ollaan siirtymässä vihreään sähköjärjestelmään. Sähköjärjestelmässä tulee millä tahansa hetkellä tuottaa täsmälleen yhtä paljon sähköä kuin sitä kulutetaan. Perinteisesti tuotantoa on säädetty kunkin hetken kysynnän mukaan, mutta nykyisin yhä suurempi osa

sähköntuotannosta riippuu säästä, jolloin jouston tarve sähköjärjestelmässä kasvaa etenkin kun joustavaa ja helposti säädettävää sähköntuotantoa on kadonnut markkinoilta. Tehokkaiden sähkön varastoinnin menetelmien puuttuessa tätä joustoa haetaan kuluttajilta. Tulevaisuudessa kuluttajatkin voivat osallistua kysyntäjoustoon, eli siirtää sähkönkulutustaan sähköntuotannon mukaan. Vielä nykyisin suurin osa kuluttajista toimii vuorokausimarkkinoilla, joissa myydään ja ostetaan sähköä seuraavalle päivälle. Tulevaisuudessa yhä useammalla kuluttajalla kuitenkin on mahdollisuus siirtyä päivänsisäisille markkinoille, jossa kuluttaja voi ostaa sähkönsä lähempänä kulutusajankohtaa. Tällöin kuluttajat voivat siirtää kulutustaan edullisempiin ajankohtiin joko itse tai palvelunvälittäjän kautta. Tällaiset palvelunvälittäjät säätelevät kuluttajien säätövaraa, kuten vaikka lämmitystä tai sähköautojen akkujen latausta sen hetken markkinoiden mukaan. Tämän toteuttaminen on jo mahdollista ilman että asumismukavuus kärsii (Fingrid 2018).

Tulevaisuuden sähkömarkkinoilla on uusiutuvasta energiasta koostuvan joustavan tuotannon sekä kuluttajien kulutusjouston lisäksi riittävästi siirtokapasiteettiä sekä tarpeeksi tehokkaita varastointimenetelmiä (Fingrid 2018). Erityisesti kulutusjouston toteuttamiseen tarvitaan tarkkoja sähkönkulutuksen ennustusmenetelmiä, jotta sähköjärjestelmä pystyttäisiin pitämään mahdollisimman vakaana. Tässä tutkielmassa tutkitaan juuri tämän kaltaisia ennustusmenetelmiä, jotta tulevaisuudessa voitaisiin siirtyä luotettavien joustopalveluiden piiriin.

1.2 Ohjattu ja ohjaamaton oppiminen

Eräs keino jaotella tilastollisia malleja on jakaa ne sen mukaan, miten mallien parametrien arvot estimoidaan eli miten malli opetetaan. Tälläinen jaottelu on ohjaamattoman ja ohjatun oppimisen välillä. Ohjatussa oppimisessa tilastollinen malli oppii eräänlaisen opettajan avulla. Malliin syötetään jokin syöte, jonka jälkeen malli antaa jonkin tuloksen tälle syötteelle. Sen jälkeen opettaja korjaa mallia käytettyä syötettä vastaavan havaitun arvon mukaan. Ohjatussa oppimisessa malli opetetaan siis havaituilla syöte-tulos -pareilla (x_i, y_i), missä x_i on syöte ja y_i tulos.

Ohjaamattomassa oppimisessa käsitellään tapauksia, joissa syöte-tulos -parien sijaan käytettävissä on N kappaletta havaintoja $(x_1, x_2, \ldots, x_N)$ jostain satunnaismuuttujasta X, jolla on tiheysfunktio f_X. Tästä otoksesta on tarkoitus suoraan päätellä em. tiheysfunktion ominaisuuksia. Tällöin ei ole käytössä opettajaa, joka korjaisi mallin tuloksia oikean havainnon tai virheen avulla. Ohjaamattomassa oppimisessa on hankalaa arvioida muodostetun mallin paikkaansapitävyyttä, sillä ohjaamattoman oppimisen asetelmassa ei ole välttämättä käytössä luotettavia mallin virheen tai onnistumisen

mittareita.

Esimerkkejä ohjatusta oppimisesta ovat mm. lineaarinen ja logistinen regressio sekä keinotekoiset neuroverkot. Ohjaamattomasta oppimisesta esimerkkejä taas ovat mm. pääkomponenttianalyysi sekä K-means ryhmittelymenetelmä (Hastie ym. 2001). Tässä tutkielmassa keskitytään pääosin keinotekoisiin neuroverkkoihin ja tukivektoriregressioon, jotka ovat ohjatun oppimisen menetelmiä.

2 Tukivektoriregressio

2.1 Tukivektorikone

Tukivektorikone (support vector machine, SVM) on aineiston ryhmittelykeino, jossa aineiston kahden eri ryhmän välille pyritään luomaan ns. optimaalinen hypertaso, joka osittaa aineiston havainnot. Kahta luokkaa ei kuitenkaan aina pystytä erottelemaan täysin, joten silloin joudutaan hyväksymään myös se tilanne, että osa luokiteltavista pisteistä joutuu hypertason väärälle puolelle. Tukivektorikoneen avulla tämänkaltainen luokittelu onnistuu helposti. Tukivektorikoneen avulla voidaan ratkaista sekä lineaarisia, että epälineaarisia tehtäviä. Luokiteltava aineisto kuvataan korkeaulotteisempaan avaruuteen ydinfunktion avulla, jolloin luokat pystytään helpommin erottelemaan yhden hypertason avulla (Hastie ym. 2001).

2.2 Tukivektoriregressio

Tukivektorikoneen periaatetta voidaan soveltaa myös regressioon, jolloin tuloksena on tukivektoriregressio (support vector regression, SVR). Tällöin etsitään mahdollisimman sileää funktiota f siten, että havaittujen arvojen y_i etäisyydet tästä funktiosta olisivat mahdollisimman pieniä. ε-regressiossa arvojen y_i etäisyydet ovat suurimmillaan arvon ε suuruisia. Monesti kuitenkin halutaan sallia myös suurempia virheitä, jolloin otetaan käyttöön ns. ε-insensitiivinen virhefunktio

$$|\xi|_\varepsilon = \begin{cases} 0, & \text{jos } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{muuten.} \end{cases}$$

Näiden ylimääräisten virheiden halutaan kuitenkin olevan mahdollisimman pieniä, joten tukivektoriregression optimointiongelmaan lisätään kaksi uutta muuttujaa ξ_i ja ξ_i^*. Nyt voidaan muotoilla optimointiongelma. Otetaan esimerkkifunktioksi lineaarinen funktio $f(x) = \langle w, x \rangle + b$, missä $w \in \mathcal{X}, b \in \mathbb{R}$ ja $\langle \cdot, \cdot \rangle$ kuvastaa pistetuloa joukossa $\mathcal{X}$. Sileys tässä tapauksessa tarkoittaa, että etsimme mahdollisimman pientä arvoa muuttujalle w. Yksi tapa tehdä niin on minimoida normia $||w||^2$. Kun yhdistämme esimerkkifunktiomme f ja virhefunktiomme, saamme seuraavan optimointiongelman:

$$\begin{aligned} \min_{w,\xi,\xi^*,b} \quad & \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \text{kun} \quad & y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ & \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \end{aligned}$$

missä normina käytetään euklidista normia $||x||_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$ ja $\langle x, y \rangle$ on pistetulo, eli $\langle x, y \rangle = \Sigma_{i=1}^n x_i y_i$. Kerroin $C > 0$ kuvastaa sitä, minkälaisen kompromissin suostumme tekemään sileyden ja yli ε suuruisten virheiden välillä.

Tämä optimointiongelma on helpompi ratkaista ns. duaalimuodossaan. Muodostetaan ensin Lagrangen funktio

$$\begin{aligned} L = \frac{1}{2}||w||_2^2 + C\sum_{i_0}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ - \sum_{i=1}^{l}\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) \\ - \sum_{i=1}^{l}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b). \end{aligned}$$

L on Lagrangen funktio ja $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ ovat Lagrangen kertoimia, joille pätee ehto

$$\alpha_i^{(*)}, \eta_i^{(*)} \geq 0.$$

Merkinnällä $\alpha_i^{(*)}$ viitataan molempiin muuttujiin α_i ja α_i^*. Lagrangen funktiolla L on minimi gradientin nollakohdassa, joten täytyy olla

$$\begin{aligned} \frac{\partial L}{\partial b} = &\quad \sum_{i=1}^{l}(\alpha_i^* - \alpha_i) = 0 \\ \frac{\partial L}{\partial w} = &\quad w - \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)x_i = 0 \\ \frac{\partial L}{\partial \xi_i^{(*)}} = &\quad C - \alpha_i^{(*)} - \eta_i^{(*)} = 0. \end{aligned}$$

Näiden ehtojen avulla saadaan Lagrangen funktiosta esitettyä optimointiongelman duaalimuotoinen tehtävä:

$$\begin{aligned} \max_{\alpha,\alpha^*} \quad & -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ & -\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \qquad (1) \\ \text{kun} \quad & \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \text{ ja } \alpha_i, \alpha_i^* \in [0, C]. \end{aligned}$$

Ratkaisemalla tämä optimointiongelma saadaan selville haluttu regressiofunktio. Tukivektoriregression ennalta määritettäviä hyperparametreja ovat ε ja C. Hyperparametri ε määrittää optimoinnin tuloksena olevan funktion molemmille puolille tulevien kaistojen leveyden. Näiden kaistojen sisälle sijoittuu suurin osa havainnoista. C taas kuvaa sitä, kuinka suuri määrä havaintoja saa olla tämän ε-putken sisällä. Toisin sanottuna C määrittää tuloksena saatavan regressiofunktion sileyttä. (Smola ja Schölkopf 2004).

2.3 Ydinfunktiot

Edellisessä alaluvussa määriteltiin tukivektoriregression idea ja esitettiin regressiofunktion ratkaisemiseen liittyvä optimointiongelma. Kuitenkin edellisessä alaluvussa esitetty ongelma toimii vain lineaarisissa tapauksissa. Tukivektoriregressio voidaan kuitenkin laajentaa myös epälineaarisiin tapauksiin. Yksinkertainen ratkaisu tähän ongelmaan olisi esikäsitellä aineistoa ja kuvata se kuvauksella $\Phi : X \to F$ sopivaan avaruuteen, jossa voisimme suoraan hyödyntää lineaarista tukivektoriregressiota. Tämänkaltaisesta toimenpiteestä tulee kuitenkin helposti laskennallisesti raskas.

Optimointiongelman (1) määrittelystä huomataan, että tukivektoriregressio ei riipu suoraan aineiston havainnoista x_i vaan useiden havaintojen välisistä sisätuloista $\langle x_i, x_j \rangle$. Epälineaarisen ongelman ratkaisussa ei tarvitse tietää funktiota Φ, joka kuvaa havainnot sopivaan avaruuteen, vaan tulee tietää pelkästään havaintojen väliset sisätulot tässä avaruudessa $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$. Tätä funktiota k sanotaan myös ydinfunktioksi (kernel). Tällöin optimointiongelman duaalimuoto voidaan muotoilla uudestaan seuraavanlaisesti

$$\begin{aligned} \text{maksimoi} \quad & -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ & -\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \\ \text{kun} \quad & \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \text{ ja } \alpha_i, \alpha_i^* \in [0, C] \end{aligned}$$

(Smola ja Schölkopf 2004).

Ydinfunktiona voidaan käyttää esimerkiksi polynomista funktiota

$$k(x_i, x_j) = (\langle x_i, x_j \rangle + c)^p$$

tai radiaalikantafunktiota

$$k(x_i, x_j) = \exp\{-\gamma |x_i - x_j|^2\}$$

(Vapnik 2000).

3 Neuroverkot

Tässä kappaleessa pääasiallisena lähteenä on käytetty lähdettä (Rojas 2013).

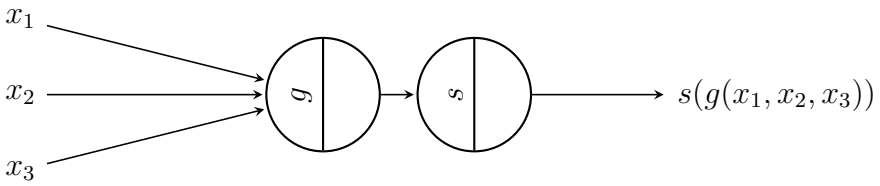
Warren McCulloch ja Walter Pitts tekivät ensimmäiset mallit keinotekoisista neuroneista vuonna 1943. Keinotekoiset neuronit matkivat keskeiseltä arkkitehtuuriltaan ihmisen hermosoluja. Ihmisen hermosolujen keskeisimmät rakenteet ovat dendriitit, sooma sekä aksoni ja synapsit. Dendriitit vastaanottavat signaaleja muilta neuroneilta. Sooma taas on solun vartalo, jossa sijaitsee mm. tuma ja mitokondriot. Soomassa siis tapahtuu energian vapauttaminen solun käyttöön sekä proteiinien valmistaminen. Aksonien tehtävänä on kuljettaa ulospäin meneviä signaaleja muille soluille synapsien, eli kahden hermosolun välisen liitoksen kautta. Dendriitit ovat siis hermosolun syötepaikkoja, sooma muodostaa solun vartalon, aksoni siirtää signaaleja ja synapsit siirtävät signaalin toiselle hermosolulle.

Keinotekoiset neuronit noudattavat samanlaista rakennetta kuin ihmisenkin hermosolut. Keinotekoisella neuronilla on syötteen vastaanottopaikkoja, vartalo ja syötteen ulostulopaikkoja. Näiden neuronien väliset yhteydet ovat kuin aksoneita ja kohdat joissa nämä yhteydet kohtaavat neuronin vartalon tai vaihtoehtoisesti syötteen vastaanotto- tai ulostulopaikan ovat synapseja.

Keinotekoiset neuroverkot koostuvat yksiköistä, joiden väliset yhteydet kuljettavat informaatiota verkon sisällä. Näitä neuroverkkoja voidaan kuvata hyvin yksinkertaisesti ja havainnollisesti graafein. Graafin solmut ovat verkon yksiköitä ja näitä yksiköitä yhdistävät kaaret taas yksiköiden välisiä yhteyksiä. Solmun sisällöllä taas kuvataan, mitä yksittäisen yksikön sisällä tapahtuu. Kuvassa 1 havainnollistetaan keinotekoisen neuronin rakennetta. Yleensä yksi yksikkö sisältää syötteiden yhdistämiseen tarkoitetun summaajafunktion g sekä aktivaatiofunktion s, joka laskee summaajafunktion perusteella yksikön ulostulon. Tämä summaajafunktio on yleensä syötteiden summa, tosin muitakin summaajafunktioita voidaan käyttää.

Keinotekoisia neuroverkkoja on olemassa painotettuina sekä painottamattomina. Tässä tutkielmassa käsitellään vain painotettuja verkkoja, sillä suurin osa nykyisin käytettävistä keinotekoisista neuroverkoista on painotettuja verkkoja, joiden opetus tapahtuu verkon painoja muokkaamalla. Painotetun verkon jokaisella kaarella on oma painonsa w, jolla kerrotaan kyseisen kaaren kautta kulkevaa informaatiota.

Keinotekoinen neuroverkko yhdistää yksittäisten yksiköiden funktioita tietyillä painotuksilla. Verkko siis loppujen lopuksi kuvaa tiettyä verkkofunktiota φ, joka riippuu verkon laskentayksiköiden summaaja- ja aktivaatiofunktioista sekä painotettujen yhteyksien painokertoimista. Neuroverkkojen avulla voidaankin sen parametreja muuttelemalla pyrkiä estimoimaan jotain epälineaarista funktiota f.



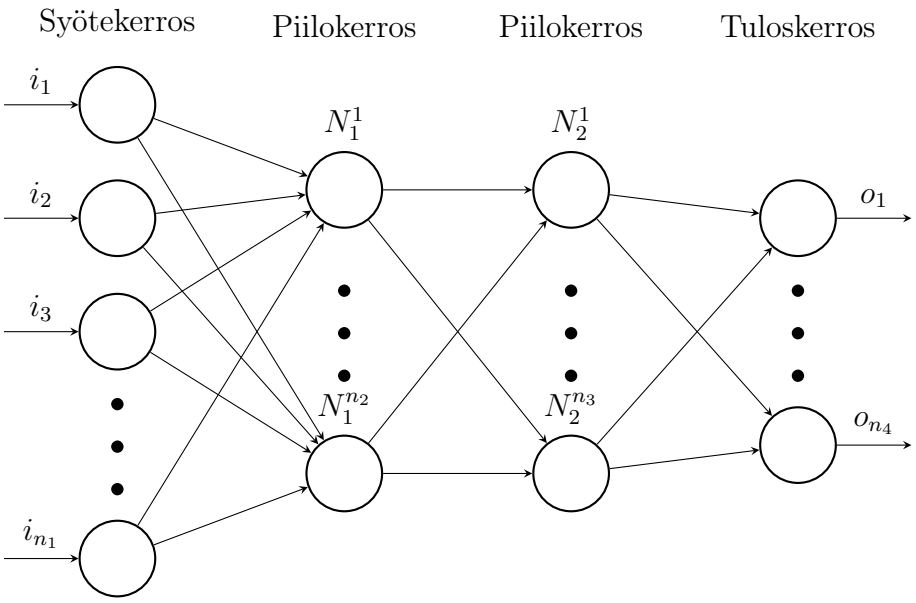
Kuva 1: Keinotekoinen neuroni, jolla on syötteet x_1, x_2 ja x_3, summaajafunktio $g : \mathbb{R}^3 \to \mathbb{R}$ ja aktivaatiofunktio $s : \mathbb{R} \to \mathbb{R}$.

Määritelmä 3.1. Keinotekoisen neuroverkon arkkitehtuuri on järjestetty monikko (I, N, O, E), missä I on verkon syötepaikkojen joukko, N on verkon yksikköjen, eli solujen joukko, O on verkon tulospaikkojen joukko sekä E on verkon painotettujen, suunnattujen kaarien joukko. Jokainen kaari on järjestetty monikko (u, v, w), jossa $u \in I \cup N, v \in N \cup O$ ja $w \in \mathbb{R}$. Verkon yksiköt taas ovat muotoa (g, s), missä $g : \mathbb{R}^n \to R$ ja $s : \mathbb{R} \to \mathbb{R}$ ja $n \in \mathbb{N}$ on kyseisen yksikön saamien syötteiden lukumäärä.

Määritelmä 3.2. Eteenpäinsyöttävä neuroverkko (feed forward neural network, FF) on keinotekoinen neuroverkko (I, N, O, E), jonka yksiköt jakautuvat kerroksiin $N_1, N_2, \ldots, N_l$. Eteenpäinsyöttävän neuroverkon yksiköiden välillä on yhteyksiä vain kahden perättäisen kerroksen välillä. Yhteyksiä on myös syötepaikkojen ja ensimmäisen kerroksen sekä tulospaikkojen ja viimeisen kerroksen N_l välillä.

Huomautus 3.3. Keinotekoisen neuroverkon syötepaikkoja kutsutaan yleensä syötekerrokseksi, yksiköitä, joista verkon tulos luetaan, tuloskerrokseksi, joka koostuu tulosyksiköistä. Syöte- ja tuloskerroksen välillä olevia kerroksia kutsutaan piilokerroksiksi (hidden layer).

Tässä tutkielmassa käsitellään eteenpäinsyöttäviä sekä takaisinkytkeytyviä neuroverkkoja. Eteenpäinsyöttävässä neuroverkossa yksiköiden välillä yhteyksiä on vain vierekkäisten kerrosten välillä. Takaisinkytkeytyvä neuroverkko taas on laajennos eteenpäinsyöttävästä neuroverkosta. Eteenpäinsyöttävien yhteyksien lisäksi takaisinkytkeytyvään neuroverkkoon lisätään yhteyksiä piilotetussa kerroksessa olevien yksiköiden eri tilojen välille. Piilokerroksessa olevat yksiköt saavat syötteenä edelliseltä kerrokselta olevan syötteen lisäksi saman yksikön aikaisemman tuloksen. Kuvassa 2 on tavallinen eteenpäinsyöttävä neuroverkko. Tavallisessa eteenpäinsyöttävässä neuroverkossa jokainen yhden kerroksen yksikkö on yhteydessä jokaiseen seuraavan kerroksen yksikköön ja kaikki yhteydet ovat kahden peräkkäisen kerroksen välillä.



Kuva 2: Tavallinen eteenpäinsyöttävä neuroverkko, jossa on kaksi piilokerrosta. Jokainen edellisen kerroksen yksikkö on yhteydessä jokaiseen seuraavan kerroksen yksikköön.

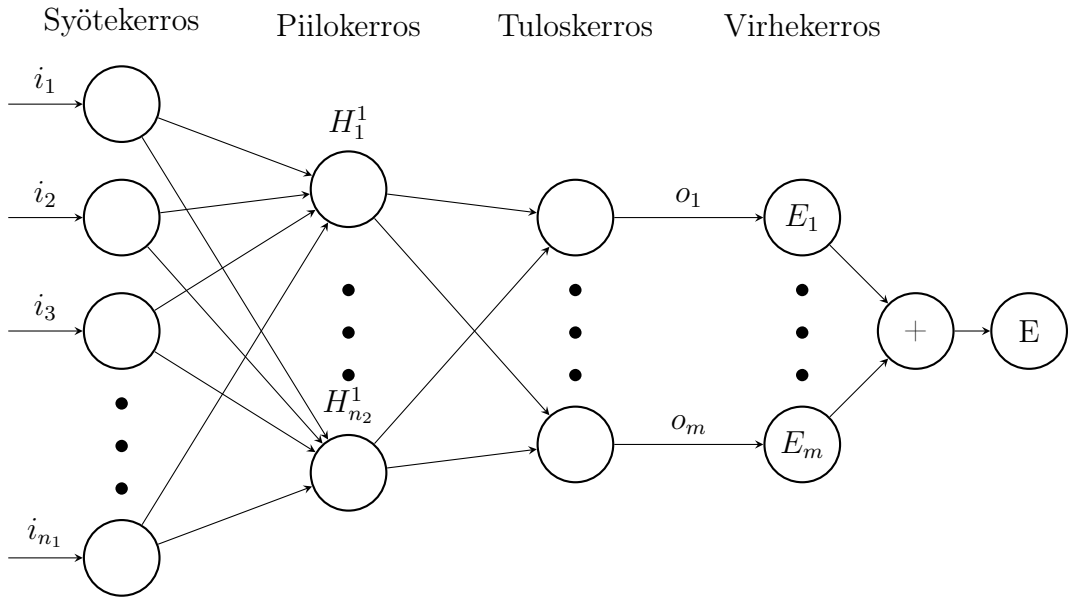
3.1 Neuroverkon opettaminen

Neuroverkkojen opettaminen tapahtuu yleensä vain verkon yksikköjen välisten yhteyksien painokertoimia muokkaamalla. Tällöin neuroverkon arkkitehtuuri on vakio kaarien painokertoimia lukuun ottamatta. Verkkoja opetetaan yleensä gradienttia hyödyntävillä optimointimenetelmillä. Verkon painokertoimet alustetaan ensin satunnaisiksi arvoiksi, jonka jälkeen opetusaineistosta $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ valitaan yksi pari (x_p, y_p). Luku x_p on neuroverkon syöte ja y_p verkon odotettu tulos em. syötteelle. Tämän jälkeen x_p syötetään neuroverkkoon ja lasketaan neuroverkon tulos. Tulosta verrataan arvoon y_p, josta voidaan laskea verkon virhe e. Tämän virhearvon avulla voidaan sen jälkeen muokata verkon painokertoimia sopimaan paremmin opetusaineistoon. Sama toistetaan myös muille (x, y)-pareille opetusaineistossa.

Verkon opettamisessa tärkeitä käsitteitä ovat aikakausi (epoch) ja erä (batch). Yhden aikakauden aikana koko opetusaineisto on syötetty verkkoon yhden kerran ja verkon painokertoimia muutettu syötteiden virheiden mukaan. Erä kertoo sen, kuinka monen opetushavainnon virheet lasketaan ennen kuin verkon painokertoimia muutetaan. Jos verkon opetusvaiheessa eräkoko on 32, niin opetuksessa käydään läpi 32 havaintoa ja lasketaan niiden virheet, jonka jälkeen verkon painokertoimia muutetaan näiden 32:en havainnon yh-

teisen virheen mukaan.

Verkon opettamisessa tehtävänä on siis löytää optimaaliset painotukset siten, että neuroverkon verkkofunktio φ on mahdollisimman lähellä estimoitavaa funktiota f, kun estimoitava funktio f on annettu epäsuorasti esimerkkiaineiston avulla. Nämä painotukset voidaan etsiä esimerkiksi tutuilla optimointimenetelmillä kuten gradienttimenetelmällä tai stokastisella gradienttimenetelmällä.



Kuva 3: Laajennettu eteenpäinsyöttävä verkko, jossa on aluksi verkon syötekerros, piilokerrokset ja tuloskerros. Tämän jälkeen verkosta lasketaan koko verkon virhefunktio laskemalla jokaisen tulosyksikön virhe erikseen ja summaamalla ne.

Aluksi laajennamme neuroverkkoa siten, että neuroverkko laskee tuloksen sijaan suoraan verkon tuloksen ja odotetun tuloksen välisen virheen. Virheeksi voidaan valita vaikka keskineliövirhe (mean squared error, MSE), jota hyödynnetään tässä tutkielmassa. Jos verkossa on useita tulosyksiköitä, lasketaan jokaisen yksikön virhe erikseen ja summataan nämä virheet $E_1, \dots E_m$, jolloin saadaan koko verkon virhefunktio E. Jotta verkon painokertoimia voisi muokata gradienttimenetelmän avulla, tulee laskea em. virhefunktion osittaisderivaatat kaikkien painokertoimien suhteen erikseen. Tällöin saamme virhefunktion E gradientin

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_L} \right).$$

Keinotekoinen neuroverkko on vain ketju yhdistettyjä funktioita, joten analyysista tuttu ketjusääntö on suuressa roolissa gradientin laskemisessa.

3.1.1 Vastavirta-algoritmi

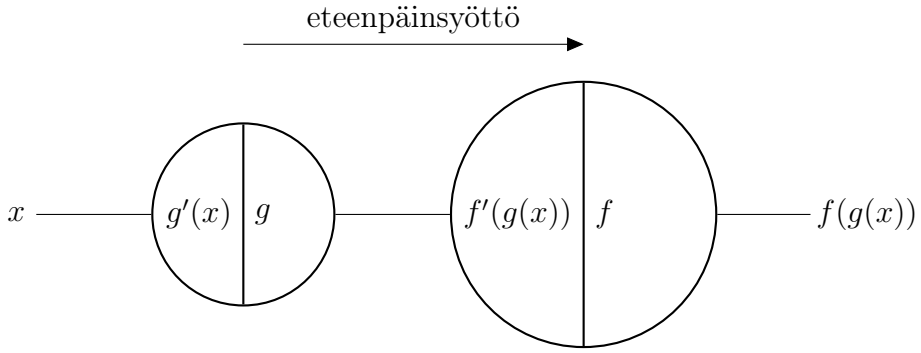
Gradientin määrittämisessä hyödynnetään vastavirta-algoritmia (backpropagation algorithm), jossa verkko evaluoidaan kahteen kertaan. Ensimmäistä vaihetta kutsutaan eteenpäinsyöttö- ja toista vastavirtavaiheeksi. Eteenpäinsyötössä verkkoon syötetään syöte vasemmalta ja jokaisen verkon yksikön oikealla puolella lasketaan yksikön funktion arvo syötteen antamassa pisteessä sekä määritetään myös yksikön funktion derivaatta sen vasemmalla puolella. Vastavirtavaiheessa verkkoa käytetään takaperin, jolloin hyödynnetään edellisessä vaiheessa laskettuja arvoja. Vastavirtavaihetta tarkastellessa tulee vastaan kolmenlaisia tilanteita.

Kuva 4: Verkko joka muodostaa kahden funktion yhdisteen. Eteenpäinsyöttövaiheessa verkon jokaisen yksikön oikealla puolella lasketaan yksikön funktion arvo ja vasemmalla puolella määritetään yksikön funktion derivaatan arvo.

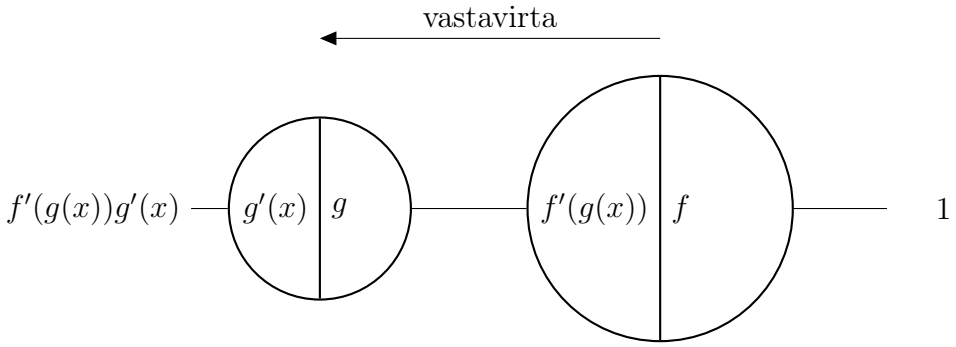
Ensimmäisessä tilanteessa verkossa luodaan eteenpäinsyöttövaiheessa yhdistetty funktio. Eteenpäinsyöttövaiheessa siis verkkoon syötetään vasemmalta syöte, joka kulkee verkon läpi. Kun syöte saapuu verkon yksikköön, lasketaan yksikön vasemman ja oikean puolen funktioiden arvot, jonka jälkeen yksikön oikean puolen arvo siirtyy verkossa eteenpäin. Lasketut arvot tallennetaan yksikköön. Kuvassa 5 näytetään verkon tulos, kun verkolle annetaan syöte x sen vasemmalta puolelta.

Vastavirtavaiheessa verkkoon annetaan syöte 1 sen oikealta puolelta. Kun syöte saapuu verkon yksikköön, kerrotaan syötettä verkon yksikön vasemman puolen arvolla, jonka jälkeen tulo siirtyy verkossa vasemmalle. Kuvassa 6 näkyy, että vastavirtavaiheen tulokseksi saadaan $f'(g(x))g'(x)$, joka on eteenpäinsyöttövaiheessa tulokseksi saadun yhdistetyn funktion $f(g(x))$ derivaatta. Samalla tavalla voidaan laskea minkä tahansa yhdistetyn funktion derivaatta. Vastavirtavaihe on siis yhtäpitävä derivaatan ketjusäännön kanssa.

Toisessa tilanteessa verkko laskee kahden funktion arvot yhteen. Kuvassa 7 kuvataan kyseisen verkon eteenpäinsyöttövaihe, jossa tulokseksi verkko laskee oikein funktioiden f_1 ja f_2 summan pisteessä x, joka on $f_1(x) + f_2(x)$.



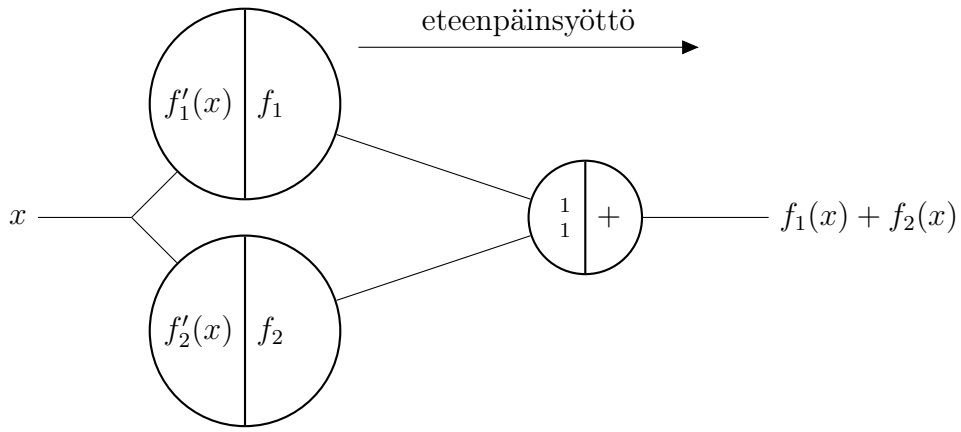
Kuva 5: Eteenpäinsyötön lopputulos, kun verkon syöte on x ja verkon yksiköiden funktiot ovat g ja f.



Kuva 6: Vastavirtavaiheen lopputulos, kun verkkoon annetaan syöte 1 verkon oikealta puolelta. Tällöin saadaan verkon oikealta puolelta yhdistetyn funktion, eli koko verkon, derivaatta.

Vastavirtavaiheessa annetaan verkon syötteeksi sen vasemmalta puolelta jälleen luku 1. Kohdissa, joissa kaksi verkon kaarta kohtaavat toisensa, lasketaan molemmissa kaarissa kulkeva informaatio yhteen. Kuvassa 8 lasketaan vastavirtavaiheen tulos $f_1'(x) + f_2'(x)$, joka on eteenpäinsyöttövaiheessa saadun summan derivaatta pisteessä x. Vastaavalla tavalla voidaan määrittää useammankin funktion summan derivaatta.

Kolmannessa tilanteessa käsitellään painotettuja kaaria. Eteenpäinsyöttövaiheessa kaaren kautta kulkeva syöte kerrotaan kaaren painolla w, jolloin tulos on wx. Vastavirtavaiheessa verkon oikealta puolelta syötettävää syötettä 1 kerrotaan kaaren painolla w. Tulokseksi saadaan tällöin w, joka on eteenpäinsyöttövaiheen tuloksen wx derivaatta muuttujan x suhteen. Painotettuja kaaria voidaan siis käsitellä samalla tavalla riippumatta siitä, onko kyseessä eteenpäinsyöttö- vai vastavirtavaihe.



Kuva 7: Kuvan verkko laskee funktioiden f_1 ja f_2 summafunktion eteenpäinsyöttövaiheessa.

Nyt on osoitettu, että vastavirta-algoritmi toimii hyvin yksinkertaisilla keinotekoisilla neuroverkoilla, jotka laskevat pelkästään funktioiden summia, yhdistettyjä funktioita tai joka laskee tuloksen, kun verkossa informaatio kulkee painotetun kaaren läpi. Seuraavassa lauseessa todistetaan, että algoritmi 1 todellakin määrittää verkkofunktion derivaatan pisteessä x myös monimutkaisemmissa verkoissa, joissa näitä edellä esiteltyjä tapauksia yhdistellään.

Lause 3.4. *Algoritmi 1 määrittää verkkofunktion F derivaatan verkon syötteen x suhteen.*

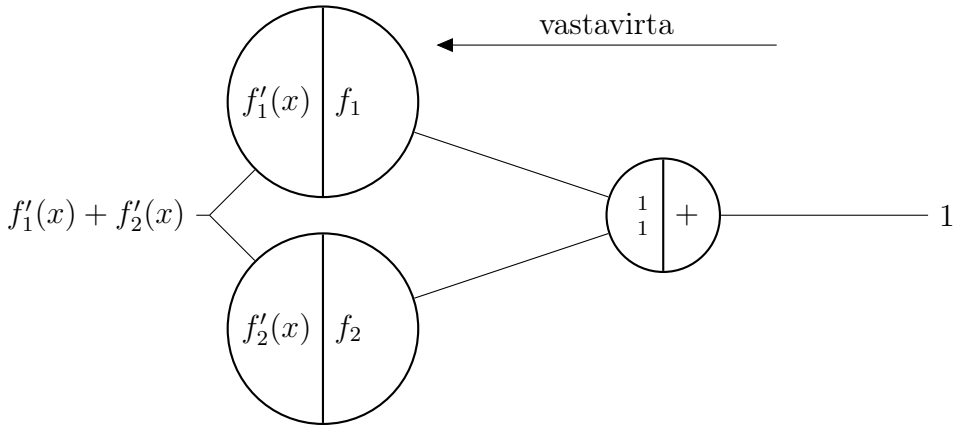
Todistus. Aikaisemmin on jo näytetty, että algoritmi toimii verkoilla, joissa on yksikköjä sarjassa tai rinnan ja kun verkon kaaret ovat painotettuja. Tehdään induktio-oletus, että algoritmi toimii kaikilla eteenpäinsyöttävillä verkoilla joissa on n tai vähemmän yksikköä. Käsitellään nyt kuvan 10 verkkoa, jossa on $n+1$ yksikköä. Suoritetaan ensin eteenpäinsyöttövaihe ja lasketaan verkkofunktion F arvo pisteessä x. Oletetaan, että m yksikköä, joiden tulokset ovat $F_1(x), \ldots, F_m(x)$, ovat yhteydessä tulosyksikköön. Koska φ on tulosyksikön sisältämä funktio, tiedetään että

$$F(x) = \varphi(w_1F_1(x) + w_2F_2(x) + \ldots + w_mF_m(x)).$$

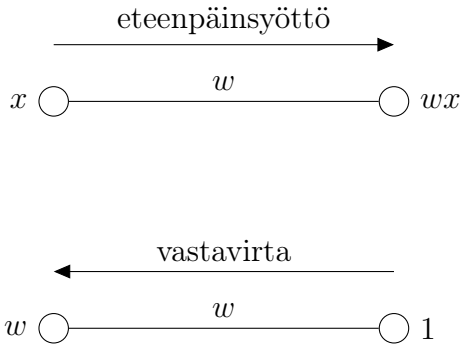
Siten verkkofunktion F derivaatta pisteessä x on

$$F'(x) = \varphi'(s)(w_1F_1'(x) + w_2F_2'(x) + \ldots + w_mF_m'(x)),$$

missä $s = w_1F_1(x) + \ldots + w_mF_m(x)$. Käsiteltävän graafin aligraafi, joka sisältää kaikki reitit syötepaikasta yksikköön jonka tulos on $F_1(x)$, määrittää



Kuva 8: Vastavirtavaihe verkossa, jossa lasketaan eteenpäinsyöttövaiheessa kahden funktion yhteenlasku. Tuloksena saadaan summafunktion derivaatta.



Kuva 9: Verkon kaaren eteenpäinsyöttö- ja vastavirtavaihe. Eteenpäinsyöttövaiheessa kaaren syöte kerrotaan kaaren painolla. Vastavirtavaiheessa lasketaan eteenpäinsyöttövaiheen muodostaman funktion derivaatta.

aliverkon, jonka verkkofunktio on F_1 ja jossa on vähemmän kuin n yksikköä. Induktio-oletuksen nojalla aligraafin derivaatta voidaan määrittää algoritmin 1 mukaisesti. Vastaavasti voidaan luoda $m-1$ muuta aliverkkoa, jotka vastaavat yksiköitä, joiden tulokset ovat $F_2(x), \ldots, F_m(x)$. Jos arvon 1 sijasta syötämme vastavirtavaiheessa aliverkkoon i vakion $\varphi'(s)$ kerrottuna vastaavalla painolla w_i, saadaan aliverkkojen syötepaikoista vastavirtavaiheiden tuloksiksi $w_1F_1'(x)\varphi'(s), w_2F_2'(x)\varphi'(s), \ldots, w_mF_m'(x)\varphi'(s)$. Koko verkon vastavirtavaiheessa nämä m tulosta lasketaan yhteen, jolloin koko vastavirtavaiheen tulokseksi saadaan

$$\varphi'(s)(w_1F_1'(x) + w_2F_2'(x) + \ldots + w_mF_m'(x)),$$

joka on verkkofunktion F derivaatta pisteessä x. Sen sijaan että verkon tulos-

Algoritmi 1 Vastavirta-algoritmi

Käsitellään verkkoa, jolla on yksi syöte $x \in \mathbb{R}$ ja verkkofunktio $F : \mathbb{R} \to \mathbb{R}$. Derivaatta F' määritetään kahdessa vaiheessa.

Eteenpäinsyöttö:
syöte x syötetään verkkoon. Verkon yksiköiden sisältämät funktiot ja niiden derivaatat lasketaan yksikön syötteen määräämässä pisteessä. Määritetyt derivaattojen arvot säilytetään yksikössä.

Vastavirta:
vakio 1 syötetään verkon tulospaikkaan, jonka jälkeen verkkoa käytetään takaperin. Verkon yksikköön tulevat syötteet lasketaan yhteen ja summaa kerrotaan yksikön vasemmalle puolelle varastoidulla arvolla. Laskettu tulo jatkaa kulkemista yksikön vasemmalle puolelle. Vaiheen tuloksena saadaan verkon syötepaikasta verkkofunktion derivaatta muuttujan x suhteen.

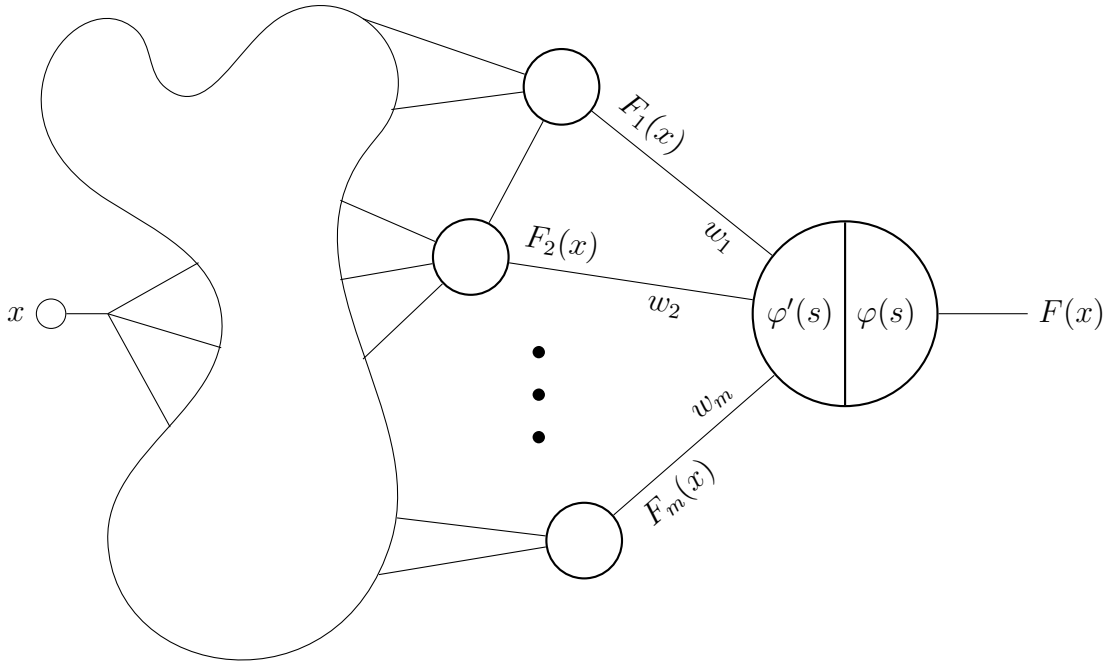
yksikköön yhteydessä oleviin yksikköihin syötetään vakiot $w_1\varphi'(s), w_2\varphi'(s), \ldots, w_m\varphi'(s)$, saadaan sama tulos syöttämällä koko verkon tulosyksikköön vakio 1, jota kerrotaan vakiolla $\varphi'(s)$. Tämä tulo siirretään sen jälkeen tulosyksikön jokaisen painotetun yhteyden kautta vasemmalle. Tällöin toteutetaan itse asiassa algoritmia 1. Siten algoritmi toimii verkoilla, joissa on $n+1$ yksikköä, eli väite on tosi. □

Huomautus 3.5. Algoritmi 1 toimii myös verkoille, joissa on useampia syötteitä. Alkuperäisestä verkosta erotellaan aliverkot siten, että yhden aliverkon määrittää kaikki reitit yhdestä alkuperäisen verkon syötteestä verkon tulosyksikköön. Näin muodostetuilla verkoilla on vain yksi syöte ja yksi tulosyksikkö, jolloin algoritmia 1 voidaan soveltaa suoraan.

Huomautus 3.6. Verkon yksiköissä voidaan myös käyttää useamman muuttujan aktivaatiofunktioita. Tällöin yksikön vasemmalla puolella määritetään ja säilötään aktivaatiofunktion osittaisderivaatat yksikön eri syötteiden suhteen. Algoritmin 1 vastavirtavaiheessa yksikköön saapuva syöte kerrotaan yhdellä osittaisderivaatalla ja tämä tulo siirtyy verkon kaarelle, josta tulleen syötteen suhteen kyseinen osittaisderivaatta oli laskettu.

Huomautus 3.7. Algoritmi 1 voidaan laajentaa koskemaan myös takaisinkytkeytyviä neuroverkkoja ns. levittämällä nämä takaisinkytkeytyvät neuroverkot siten, että ne näyttävät tavallisilta eteenpäinsyöttäviltä neuroverkoilta.

Opettaessamme keinotekoista neuroverkkoa minimoimme verkon virhefunktiota E. tällöin tulee käsitellä jokaista verkon painoarvoa w samanaikaisesti. Opetusta varten joudumme laajentamaan verkkoa jälleen siten, että



Kuva 10: Vastavirtavaihe viimeisessä neuronissa.

verkko laskee virheen suoraan kuten näimme kuvassa 3. Keskitymme yhteen painoon w_{ij}, joka on verkon i:nnen ja j:nnen yksikön välinen yhteys. Tämä painotettu kaari määrittää aliverkon, johon kuuluu kaikki reitit kaaresta alkuperäisen verkon tulosyksikköön. Aliverkkoon annetaan syötteenä arvo o_i, joka on verkon i:nnen yksikön tulos. Aliverkon ensimmäisen yksikön koko syöte on siis $o_i w_{ij}$.

Vastavirta-algoritmi määrittää verkkofunktion osittaisderivaatan verkon syötteen suhteen, joten kun algoritmia 1 sovelletaan em. aliverkkoon, saadaan tuloksena $\frac{\partial E}{\partial o_i w_{ij}}$. Koska arvoa o_i voidaan käsitellä vakiona, saadaan verkon virhefunktion osittaisderivaatta painon w_{ij} suhteen määritettyä yksinkertaisesti. Tällöin

$$\frac{\partial E}{\partial w_{ij}} = o_i \frac{\partial E}{\partial o_i w_{ij}}.$$

Aliverkon vastavirtavaiheen tulosta $\frac{\partial E}{\partial o_i w_{ij}}$ sanotaan vastavirtavirheeksi ja verkon yksikön j vastavirtavirhe on δ_j. Siten

$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j.$$

Keinotekoista neuroverkkoa opetettaessa tallennetaan algoritmin 1 eteenpäinsyöttövaiheen aikana jokaiseen yksikköön sen tulos ja vastavirtavaiheen aikana vastavirtavirhe. Näiden tietojen avulla voidaan määrittää verkon virhefunktion gradientti ∇E. Gradienttia voidaan hyödyntää sitten painoarvojen päivityksessä. Gradienttimenetelmässä painoarvojen muutokset saadaan yksinkertaisesta laskukaavasta

$$\delta w_{ij} = -\gamma \frac{\partial E}{\partial w_{ij}} = -\gamma o_i \delta_j,$$

missä γ on opetuksen nopeuteen vaikuttava vakio. Jos γ on liian suuri, tehdään painokertoimille joka kerta liian suuria päivityksiä, jolloin minimiä ei löydetä. Jos taas γ on liian pieni, ovat painokertoimien päivityksetkin hyvin pieniä, jolloin neuroverkon opetukseen menee hyvin paljon aikaa.

3.1.2 Stokastinen gradienttimenetelmä

Algoritmi 2 SGD

Require: jono askelpituuksia ε_k
Require: parametrin θ alkuarvo
  $k \leftarrow 1$
  while θ ei ole supennut **do**
    valitse satunnaisesti m kpl havaintoa opetusaineistosta $\{x^{(1)}, \ldots, x^{(m)}\}$ ja niitä vastaavat tavoitearvot $y^{(i)}$
    laske gradientin estimaatti $\hat{g} \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
    päivitä parametrin θ arvo $\theta \leftarrow \theta - \varepsilon_k \hat{g}$
    $k \leftarrow k + 1$
  end while

Stokastinen gradienttimenetelmä (stochastic gradient descent, SGD) on yksi yleisimmistä nykyisin käytössä olevista optimointialgoritmeista koneoppimisessa, erityisesti syväoppimisessa. Stokastisen gradienttimenetelmän suurin ero tavalliseen gradienttimenetelmään on se, että tavallisessa gradienttimenetelmässä käytetään yleensä koko opetusaineisto gradentin arvon estimoimiseen, kun taas stokastisessa gradienttimenetelmässä valitaan koko opetusaineistosta pienempi otos, jonka avulla gradientti estimoidaan. Tämän estimaatin avulla voidaan sitten laskea parametrin päivitys. Stokastinen gradienttimenetelmä esitellään tarkemmin algoritmissa 2. Stokastisen gradienttimenetelmän etu on se, että opetusaineiston kasvaessa yhden päivityksen laskeminen ei hidastu.

Kun optimoinnissa käytetään stokastista gradienttimenetelmää, tulee oppimisen askelpituutta pienentää opetuksen edetessä, sillä gradientin estimoimiseen käytetty menetelmä tuo opetukseen virhettä joka ei katoa, vaikka minimi saavutettaisiinkin. Riittävä ehto algoritmin 2 suppenemiselle on

$$\sum_{k=1}^{\infty} \varepsilon_k = \infty \quad \text{ja} \quad \sum_{k=1}^{\infty} \varepsilon_k^2 < \infty.$$

Askelpituudelle voidaan käyttää esimerkiksi kaavaa $\varepsilon_k = (1 - \alpha)\varepsilon_0 + \alpha\varepsilon_\tau$, missä $\alpha = \frac{k}{\tau}$ ja ε_0 on ensimmäinen askelpituus. Askelpituus ε pienenee siis kierrokseen τ asti, jonka jälkeen se pysyy vakiona ε_τ (Goodfellow ym. 2016).

3.1.3 ADAM

ADAM (Adaptive Moment Estimation) kehitettiin vuonnna 2014 kahden aikaisemman optimointimenetelmän, AdaGradin sekä RMSPRopin pohjalta (Kingma ja Ba 2014). ADAM perustuu tavoitefunktion ensimmäisen kahden momentin estimoimiseen kyseisen tavoitefunktion gradientin avulla. Näistä momenteista lasketaan ensin virhettä sisältävät estimaatit, joiden arvot myöhemmin korjataan virheen varalta. ADAM kuvataan tarkemmin algoritmissa 3.

3.2 Konvoluutioneuroverkko

Konvoluutioverkko (convolutional neural network, CNN) on suunniteltu erityisesti sellaisen aineiston käsittelyyn, jossa on tunnettu ruudukkomainen rakenne. Esimerkkejä tällaisesta aineistosta ovat mm. aikasarjat sekä kuvat. Konvoluutioverkossa syötteelle x tehdään konvoluutio ytimen w kanssa

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t - a).$$

Kun konvoluutiota toistetaan tarpeeksi monta kertaa, että koko syöte on käyty läpi, saadaan tulokseksi piirrekartta, joka siirtyy eteenpäin neuroverkossa. Konvoluution ansiosta verkossa on vähemmän optimoitavia parametreja, sillä konvoluution seurauksena verkossa on harvoja yhteyksiä sekä samoja yhteyksiä käytetään useassa eri kohdassa verkkoa.

Konvoluutiokerroksen jälkeen konvoluutioverkossa tulee usein koontikerros (pooling layer), jossa lasketaan jokin yksi arvo lähekkäin olevien neuroneiden tuloksista. Yleisiä koontikerroksia ovat mm. maksimi, keskiarvo,

Algoritmi 3 ADAM

Require: α: askelpituus
Require: $\beta_1, \beta_2 \in [0, 1[$: momenttien estimaattien eksponentiaaliset hajoamisvakiot
Require: $f(\theta)$: estimoitava stokastinen funktio, jolla on parametri θ
Require: θ_0: parametrivektorin θ lähtöarvo
  $m_0 \leftarrow 0$ (ensimmäisen momenttivektorin lähtöarvo)
  $v_0 \leftarrow 0$ (toisen momenttivektorin lähtöarvo)
  $t \leftarrow 0$
  while θ_t ei ole supennut **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) g_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) g_t^2$
    $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
    $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$
  end while
  return θ

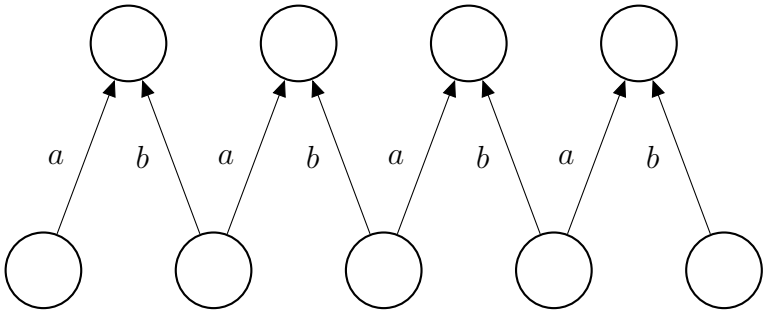
L^2-normi sekä painotettu keskiarvo. Nämä koontikerrokset vaikuttavat verkkoon siten, että jos verkon syöte muuttuu hiukan, ei verkon tulos muutu ollenkaan. Verkosta tulee siis invariantti syötteen pienten muutosten suhteen. Tämä ominaisuus on hyödyllinen kun tärkeämpää on jonkin piirteen olemassaolo kuin tarkka paikka. Koontikerrokset kaventavat verkkoa, eli pudottavat aineiston resoluutiota.

Konvoluutiokerros kaventaa verkkoa, varsinkin jos käytetään suurta ydintä. Tämän kaventumisen voi välttää lisäämällä nollia syötteen reunoille, jolloin voidaan pitää kerroksen syöte ja tulos saman kokoisina (Goodfellow ym. 2016).

Konvoluutiokerroksia, koontikerroksia sekä tavallisia kerroksia yhdistelemällä voidaan luoda syviä verkkoja jotka pystyvät havaitsemaan aineistosta monimutkaisiakin piirteitä. Konvoluutioverkkoja on käytetty mm. esineiden ja olentojen tunnistamiseen kuvista sekä kirjoituksen koneelliseen lukemiseen.

3.3 Long short-term memory -neuroverkko

Takaisinkytkeytyvät neuroverkot (recurrent neural network) ovat neuroverkkoja joissa osa verkon neuroneista saa syötteenä verkon edellisen kerroksen tuloksen lisäksi edellisen optimointikierroksen tuloksensa. Tämä mahdollis-



Kuva 11: Konvoluution soveltaminen syötteeseen.

taa sen, että verkon neuronit pystyvät muistamaan aikaisemmin verkon läpi kulkenutta informaatiota. Ominaisuus on hyödyllinen esimerkiksi konekäännöksissä, joissa tulee muistaa esimerkiksi aikaisemmin mainittu hahmon sukupuoli tai vaikka käännettävän sanan suku.

Tavallisen takaisinkytkeytyvän neuroverkon opettaminen on vaikeaa, sillä gradienttia laskettaessa voi gradientti lähestyä nollaa tai ääretöntä, jolloin neuroverkon opettaminen on mahdotonta. Tätä haastetta korjaamaan kehitettiinkin vuonna 1997 LSTM-verkko (pitkä lähimuisti -verkko, long short-term memory), jonka gradientin arvot pysyvät sellaisissa rajoissa, että verkko on helpompi opettaa (Hochreiter ja Schmidhuber 1997). LSTM-verkkoja paranneltiin myöhemmin lisäämällä ns. "unohdusportti", jonka avulla verkko voi itse oppia, mitä tietoa se säilöö ja kuinka kauan (Gers 2001).

4 ARIMA-malli

Tässä kappaleessa lähteenä on käytetty lähdettä (Box ja Jenkins 1976).

Määritelmä 4.1. Määritellään viiveoperaattori

$$Bz_t = z_{t-1}, B^m z_t = z_{t-m}$$

sekä differenssioperaattori

$$\nabla z_t = z_t - z_{t-1} = (1 - B)z_t.$$

4.1 Stationaarisuus

Määritelmä 4.2. Aikasarja on vahvasti stationaarinen, jos m:n havainnon $z_{t_1}, z_{t_2}, \ldots, z_{t-m}$ yhteinen reunajakauma $F_{z_{t_1}, z_{t_2}, \ldots, z_{t-m}}$ on sama kuin havaintojen $z_{t_1+k}, z_{t_2+k}, \ldots, z_{t-m+k}$ yhteinen reunajakauma $F_{z_{t_1+k}, z_{t_2+k}, \ldots, z_{t-m+k}}$.

Huomautus 4.3. Määritelmästä seuraa, että sarjan keskiarvo sekä varianssi ovat vakiota koko aikasarjassa.

Aikasarjan stationaarisuutta voidaan testata esimerkiksi augmentoidulla Dickey-Fullerin testillä (Greene 2003). Jos aikasarja ei täytä stationaarisuusehtoja, voidaan sitä pyrkiä muokkaamaan siten, että aikasarjasta tuleekin stationaarinen. Tälläisia muokkaustoimenpiteitä ovat esimerkiksi differensointi sekä trendin ja kausiluonteen mallintaminen ja poistaminen. Differensoinnissa aikasarjan arvosta ajanhetkellä t vähennetään saman aikasarjan arvo ajanhetkellä $t - 1$. Tämänkaltainen differenssi poistaa aikasarjasta trendin, jos sellainen aikasarjasta löytyy. On myös mahdollista suorittaa kaudellinen differensointi. Tällöin aikasarjan arvosta ajanhetkellä t vähennetään saman aikasarjan arvo ajanhetkellä $t - s$, missä s on kauden pituus. Jos siis tunnittaisessa aikasarjassa on viikon pituinen kausi, on kauden pituus $7 \cdot 24 = 168$. Differensointi voidaan purkaa lisäämällä aikasarjan arvoon ajanhetkellä t siitä poistettu arvo, eli $t - s$, jossa s on differensoinnin kauden pituus.

4.2 Autoregressiivinen prosessi

Määritelmä 4.4. Autoregressiivinen (autoregressive, AR) prosessi, jolla on kertaluku p, on prosessi, joka on muotoa

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + \ldots + \phi_p \tilde{z}_{t-p} + \varepsilon.$$

Malli voidaan esittää myös autoregressiivisen operaattorin ϕ avulla

$$\phi(B) = i - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p,$$

jolloin kertalukua p oleva prosessi saa muodon

$$\phi(B)\tilde{z}_t = \varepsilon_t$$

Autoregressiivisessa prosessissa nykyinen arvo esitetään aikasarjan aikasempien arvojen sekä satunnaisen virheen ε avulla. Termi autoregressiivinen viittaa siihen, että aikasarjan arvot selitetään saman aikasarjan aikaisempien arvojen avulla.

4.3 Liukuvan keskiarvon prosessi

Määritelmä 4.5. Liukuvan keskiarvon (moving average, MA) prosessi, jolla on kertaluku q määritellään AR-prosessissakin esiintyneiden virheiden ε_t avulla

$$\tilde{z}_t = \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q}.$$

Sama prosessi voidaan määritellä myös liukuva keskiarvo -operaattorin θ avulla

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q b^q,$$

jolloin kertalukua q oleva MA-prosessi saa muodon

$$\tilde{z}_t = \theta(B)\varepsilon_t.$$

4.4 ARIMA-prosessi

Määritelmä 4.6. Autoregressiivinen integroitu liukuvan keskiarvon prosessi (autoregressive integrated moving average, ARIMA) kertalukua (p, d, q) määritellään yleistetyn autoregressiivisen operaattorin $\varphi(B)$ avulla

$$\varphi(B) = \phi(B)(1 - B)^d.$$

Tällöin ARIMA-prosessi saa muodon

$$\varphi(B)z_t = \theta(B)\varepsilon_t,$$

joka on yhtäpitävä lausekkeen

$$\phi(B)w_t = \theta(B)\varepsilon_t$$

kanssa, missä $w_t = \nabla^d z_t$.

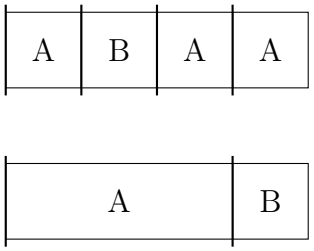
5 Aineiston analysointi

Tässä tutkielmassa analysoidaan kahdesta saman ketjun kaupasta kerättyä aineistoa. Kummastakin kaupasta on kerätty aineistoa, jossa on mitattu kaupan sähkönkulutusta. Kulutusta on mitattu usean mittarin avulla, joten sähkönkulutustietoja on esimerkiksi kaupan pakasteiden ja jääkaappien kylmennyksestä sekä ilmanvaihtolaitteista erikseen. Tutkielmassa analysoidaan molempien kauppojen sähkön kokonaiskulutusta sekä kylmäketjun ja ilmanvaihdon alimittarointeja. Sähkön kokonaiskulutusta ei mitattu kaupoissa suoraan, vaan kokonaiskulutus on laskettu summaamalla kaupoista mitatut alimittaroinnit yhteen.

Sähkönkulutusta on mitattu molemmissa kaupoissa yhden minuutin välein. Tämä aikaväli on kuitenkin hiukan liian lyhyt tutkielman tarkoitusta varten, joten kaksi uutta aineistoa muodostettiin, joissa aikaväli on 15 ja 60 minuuttia. Tämä muunnos määritettiin laskemalla vierekkäisistä havainnoista keskiarvot, jolloin saadaan muutettua aikaväliä suuremmaksi.

Kaupat ovat erilaisia, joten molempien kauppojenkin aineistot ovat eri näköisiä. Kauppa 2 on vanhempi kuin kauppa 1, joten kaupan 2 sähkönkulutuksen keskiarvo on korkeampi kuin kaupan 1. Kaupassa 1 on myös aurinkopaneeleja jotka todellisuudessa vaikuttavat kaupan sähkönkulutukseen. Tässä tutkielmassa ei kuitenkaan lähdetty tutkimaan aurinkopaneelien vaikutusta kaupan sähkönkulutukseen, joten aurinkopaneelien sähköntuotanto jätettiin huomiotta.

5.1 Mallinvalinta



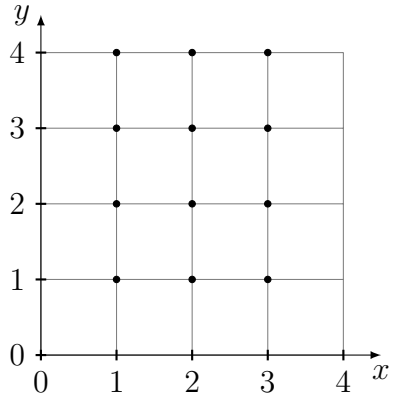
Kuva 12: Ylempi kuva kuvastaa ristiinvalidaatiota, jossa aineisto jaetaan satunnaisesti k osaan, joista yhtä käytetään mallin validointiin ja loppuja mallin opettamiseen. Alemmassa kuvassa kuvataan aikasarja-aineiston validointimenetelmää, jossa aineiston alkuosaa käytetään mallin opettamiseen ja loppuosaa mallin validointiin.

Tavallisissa tilastollisissa malleissa käytetään usein ristivalidointia, jossa

opetusaineisto jaetaan satunnaisesti k osaan. Tämän jälkeen malli opetetaan $k-1$ osalla opetusaineistosta ja viimeinen osa jää mallin validointiin. Tämä prosessi toistetaan jokaisella opetusaineiston k:lla osalla. Ristivalidointia kuvataan kuvassa 12. Ylempi kuva kuvastaa perinteistä ristivalidointia, jossa A:lla merkityllä osalla aineistoa opetetaan tarkasteltava malli ja kirjaimella B merkittyä osaa käytetään mallin validoinnissa (Hastie ym. 2001).

Perinteistä ristivalidointia ei kuitenkaan voida suorittaa aikasarja-aineistolle, sillä aikasarjassa on ajasta riippuvia osia, kuten vaikka kausittaisuus ja trendi, jonka takia aikasarjaa ei voida jakaa satunnaisesti k osaan. Ongelma voidaan ratkaista valitsemalla aikasarjan alusta tietty osa mallin opetukseen ja ylijäävä aineisto validointiin (Hastie ym. 2001). Kuvassa 12 alempi osa kuvastaa aikasarja-aineiston ristivalidointimenetelmää. Kirjaimella A merkitty osa on mallin opettamiseen ja kirjaimella B merkitty osa mallin validointiin varattua aineistoa. Aikasarja-aineiston validointimenetelmää voi muokata vielä lähemmäs perinteistä ristivalidointia suorittamalla edellä kuvailtu validointi useampaan kertaan. Ensimmäiseen validointiin käytetään esimerkiksi vain ensimmäiset 50 % aineistosta. Seuraavalla kerralla validointiin otetaan aikaisemman 50 %:n lisäksi seuraavat 10 % alkuperäisestä aineistosta ja validointia suoritetaan niin kauan, että koko aineisto on käytössä joko opetukseen tai validointiin. Tässä tutkielmassa päätettiin hyödyntää ensin kuvailtua aikasarja-aineiston validointimenetelmää.

Mallin hyperparametreilla tarkoitetaan niitä parametreja, joita ei määritetä mallin opetusvaiheessa vaan ennen sitä. Näitä hyperparametreja ovat mm. kerrosten lukumäärä ja leveys neuroverkoissa, ARIMA-mallin kertaluku ja SVR-mallissa C, ε sekä γ, jos tukivektoriregressiossa käytetään ydinfunktiona radiaalikantafunktiota.



Kuva 13: Hilaetsintä

Hilaetsintä (grid search) on menetelmä, jolla voidaan etsiä tarkasteltа-

vien mallien hyperparametreja. Hilaetsinnässä valitaan etukäteen tietyt arvot jokaiselle määritettävälle hyperparametrille. Etsintää suoritettaessa kaikki etukäteen valittujen arvojen yhdistelmät käydään läpi ja jokaiselle hyperparametrien yhdistelmistä lasketaan jokin mallin hyvyyden mittari. Tämän mittarin avulla valitaan lopuksi parhaiten menestyvä hyperparametrien yhdistelmä. (Hsu ym. 2003).

Malleja muodostettaessa aineistot jaettiin kolmeen osaan: opetus-, validointi- sekä testausosaan. Opetusosaa käytettiin nimensäkin mukaan mallien opettamiseen, validointiosaa mallien hyperparametrien määrittämiseen ja testausosan avulla arvioitiin valitun mallin toimivuutta, eli sitä, miten mallin ennustukset yleistyvät aineistoon, jota ei ole käytetty mallinvalinnassa. Opetusosaan otettiin aineiston ensimmäiset 60 % sekä validointi- ja testausosiin kumpaankin 20 % aineistosta siten, että validointiosaan kuuluu opetusosan jälkeiset 20 % ja testausosaan loput 20 % aineistosta. Tämä jako tehdään sen takia, että mallin opetukseen ei käytetä testiaineiston informaatiota. Validointiosan tarkoituksena on optimoida mallien niitä parametreja, joita mallin opettamiseen käytetty algoritmi ei pysty muuttamaan. Aineiston testiosaa ei siis käytetä ollenkaan minkään mallin parametrin viilaamiseen, vaan pelkästään mallin ennusteiden tarkkuuden määrittämiseen.

Hyperparametreja valittaessa muodostettiin ja opetettiin useita erilaisia malleja samasta malliluokasta ja laskettiin kaikkien muodostettujen mallien validointiosien aineiston virhe. Näistä muodostetuista malleista valittiin jatkoon se, jonka virhe oli pienin.

Neuroverkkomallien hyperparametreja valittaessa kokeiltiin yksi-, kaksi- ja kolmikerroksisia neuroverkkoja. Jokaisen kerroksen leveyksiä vaihdettiin mallinvalinnan aikana. Neuroverkkojen hyperparametreihin kuuluu myös aikakausien lukumäärä sekä opetuserän koko. Neuroverkkomallit ovat luonteeltaan stokastisia, joten jokaisen uuden opetettavan mallin tulos on eri kuin aikaisemmalla mallilla. Tämän vuoksi jokaisesta neuroverkkomallista opetettiin useampi yksittäinen malli joista paras otettiin mukaan arviointiin.

ARIMA-mallien kertaluvun valinnassa tehtiin hilaetsintä, jossa kokeiltiin useita kertalukuvaihtoehtoja ja näistä kertaluvuista valittiin lopuksi parhaiten toimiva. Hilaetsintään valittuja parametreja rajattiin aineistosta estimoidun autokorrelaatio- ja osittaisen autokorrelaatiokuvaajien avulla. Myös tukivektoriregression tapauksessa tehtiin hilaetsintä, jonka hilaan valittiin laaja määrä kokeiltavia lukuarvoja.

ARIMA-mallia sovitettaessa huomattiin, että mikään tarkastelluista aikasarjoista ei ollut stationaarinen. Koska aikasarjassa on sen luonteen takia selvä viikottainen kausi, tehtiin aikasarjoille kaudellinen differensointi, jolloin augmentoidun Dickey-Fullerin testin mukaan aikasarjat olivat stationaarisia. Malli sovitettiin tällöin differensoidulle aikasarjalle ja tehdyt ennustukset oli-

vat myös tämän aikasarjan arvoja. Differensoitu aikasarja muutettiin tulkintaa varten takaisin alkuperäiseksi purkamalla differensointi.

5.2 Mallien arviointi

Mallien arviointiin käytettiin kahta virhefunktiota, keskineliövirheen neliöjuurta (root mean square error, RMSE) ja tämän normalisoitua versiota (normalized root mean square error, NRMSE).

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}$$

$$\text{NRMSE} = \frac{\text{RMSE}}{y_{max} - y_{min}}$$

missä y on sähkönkulutuksen havaittu arvo, $\hat{y}$ sähkönkulutuksen ennustettu arvo y_{max} ja y_{min} ovat sähkonkulutuksen suurin ja pienin arvo. NRMSE:n määritelmän perusteella voidaan NRMSE:n arvo esittää myös prosentteina, jos sen arvo kerrotaan sadalla prosentilla. Tämä mahdollistaa jonkinasteisen vertailun eri aineistojen mallien välillä.

5.3 Analyysiin käytetty ohjelmisto

Tämän tutkielman analyysien suorittamiseen käytettiin Python-ohjelmointikieltä (Python Software Foundation). Aineiston muokkaukseen ja lataamiseen ohjelman käyttöön käytettiin pythonin kirjastoa pandas (McKinney 2010). ARIMA- ja SVR-mallien muodostamiseen käytettiin statsmodels-kirjastoa (Seabold ja Perktold 2010). Neuroverkkojen muodostamiseen taas käytettiin Keras-kirjastoa jota käytettiin TensorFlow-kirjaston päällä (Chollet 2015; Abadi ym. 2015).

5.4 Mallien toteutus

Ennen mallien muodostamista muokattiin kerätty aineisto sopivaan muotoon ohjattua oppimista varten. Aineisto muokattiin matriisiin, jossa yksi sarake sisältää ennustusta edeltäviä havaintoja viikon ajalta. Matriisiin muokattiin myös sarakkeet jotka sisältävät tehtävän ennustuksen oikeat, havaitut arvot. Ohjatun oppimisen periaatteen mukaan nämä ennustusta edeltävät arvot syötetään malliin, joka antaa ennusteen ja mallin toimintaa korjataan havaittujen arvojen avulla.

Neuroverkkojen yhteisiä hyperparametreja ovat aikakausien lukumäärä ja eräkoko. Aikakausien lukumäärä tarkoittaa sitä, kuinka monta kertaa opetukseen käytettävä aineisto käydään kokonaisuudessaan läpi. Eräkoko taas

tarkoittaa sitä, kuinka monta syötettä neuroverkolle annetaan, ennen kuin neuroverkon painoja muutetaan. Opetuksen aikana voidaan siis laskea usean opetusaineiston havainnon virhe neuroverkossa, jonka jälkeen voidaan suorittaa painojen muutos koko erän havaintojen virheiden avulla. Tämä vähentää opetuksen laskennallista taakkaa, joten suurempi eräkoko nopeuttaa yhden aikakauden läpikäymiseen tarvittavaa aikaa.

Tavallisessa eteenpäinsyöttävässä neuroverkossa sekä konvoluutioneuroverkossa neuroverkkojen syötteenä käytettiin yhden viikon sähkönkulutuksen havaintoarvoja. Näiden arvojen perusteella malli ennustaa tulevaisuutta 24 tai 2 tuntia riippuen siitä, onko kyseessä 60 vai 15 minuutin ainiesto. Tukivektoriregressiossa aineisto muotoillaan samalla tavalla. Tukivektoriregressiomalleissa tosin jokaisen ennustuksen syötteenä käytettiin kahden edellisen viikon arvoja aikasarjasta, sillä se tuotti parempia tuloksia kuin pelkän yhden viikon käyttäminen.

LSTM-mallissa LSTM-neuroneilla on eräänlainen sisäinen tila, johon ne voivat tallentaa tietoa. Halusinkin kokeilla, voisiko LSTM-verkko oppia aikasarjan rakenteen, jos sille annetaan vain yksi havainto kerrallaan. Ohjelmiston rakenteen vuoksi tämä asettaa rajoituksia erän koolle. Jokaisen opetuserän jälkeen neuroneiden sisäinen tila nollataan, joten eräkoon tulee olla suhteellisen suuri, että oppiminen edes olisi mahdollista.

ARIMA-mallin opetuksessa mallille syötettiin koko opetusaineisto kerralla mallin luonteesta johtuen. ARIMA-malli opetetaan suurimman uskottavuuden menetelmällä, joten koko opetusaineisto tarvitaan kerralla mallin opettamiseen.

Kaikki muut mallit paitsi tukivektoriregressio pystyvät suoraa ennustamaan useita askelia eteenpäin. Tukivektoriregressio antaa tuloksena vain yhden ennustuksen, sillä se on regressiomalli. Tukivektoriregression käyttämiseksi usean askeleen ennustamiseen tulee siis kehittää erilainen lähestymistapa. Tähän voidaan käyttää joko suoraa tai toistettua ennustamismallia. Suorassa ennustamismallissa luodaan useita malleja, joista jokainen ennustaa yhden askeleen eteenpäin. Ensimmäinen malli ennustaa suoraan seuraavan askeleen, toinen malli ensimmäisen mallin ennustetta seuraavan askeleen ja niin edelleen, kunnes haluttu ennustuksen pituus on saavutettu. Malleja tulee siis opettaa sitä enemmän mitä pitempi ennustus halutaan. Toistetussa ennustamisessa opetetaan vain yksi malli koko opetusaineistolla, joka ennustaa aina seuraavan askeleen. Tällä mallilla ennustettaessa käytetään seuraavan ennusteen saamiseksi aina edellistä ennustetta. Toisen ennustusaskeleen ennustamiseen käytetään siis osin havaittuja arvoja, mutta viimeisin havainto onkin ensimmäinen ennustusaskel.

6 Tulokset

Mallien tulokset esitellään taulukoissa 3 ja 4. Taulukoissa esitetään keskineliövirheen neliöjuuri ja tämän normalisoitu versio jokaisesta mallista. Aineiston jokaiselle alimittaroinnille esitetään tulokset sekä 60 minuutin että 15 minuutin aineistolle jokaisella käytössä olleella mallilla. Taulukossa 1 esitellään tulostaulukoissa käytetyt nimitykset eri alimittaroinneille. Vertailtavien menetelmien lyhenteet on esitelty taulukossa 2.

Nimitys	Tarkoitus
total	Kaupan kokonaisenergiankulutus
IV	Kaupan iv-laitteiden energiankulutus
kylmälaitteet	Kaupan kylmälaitteiden energiankulutus
jääkaapit	Kaupan jääkaappien energiankulutus
pakastimet	Kaupan pakastimien energiankulutus

Taulukko 1: Selitteet aineistossa käytetyille alimittaroinneille

Lyhenne	Selite
FF	Tavallinen eteenpäinsyöttävä neuroverkko
LSTM	Pitkän lähimuistin verkko (Long short-term memory)
CNN	Konvoluutioneuroverkko
ARIMA	Autoregressiivinen integroitu liukuvan keskiarvon prosessi
SVR	Tukivektoriregressio

Taulukko 2: Selitteet tutkielmassa käytetyille menetelmille

Taulukosta 3 huomataan, että kaupassa 1 parhaiten pärjännyt malli oli selkeästi epälineaarinen tukivektoriregressio. Se selvisi parhaimmaksi malliksi kokonaiskulutuksen ennustamisessa sekä molempien alimittarointien osalta. Tukivektoriregressio ei kuitenkaan ollut kaupan 1 ainoa hyvin selviytynyt malli. Kokonaiskulutuksen ennustamisessa hyvin toimivat myös tavallinen eteenpäinsyöttävä neuroverkko erityisesti 15 minuutin aineistolla.

Kaupan 2 tulokset näkyvät taulukosta 4. Taulukosta huomataan, että kaupassa 2 paras malli melkein jokaiselle alimittaroinnille oli tukivektoriregressio. Yhdessä ennustustilanteessa ARIMA-mallin tuottama ennustus oli tarkempi kuin tukivektoriregression ennustus. Toisaalta tukivektoriregressio ei ollut paras malli kaupan kokonaiskulutusta ennustettaessa, jossa parhaaksi malliksi selvisivät tavallinen eteenpäinsyöttävä neuroverkko sekä ARIMA-malli.

		FF		LSTM		CNN		ARIMA		SVR	
		RMSE	NRMSE	RMSE	NRMSE	RMSE	NRMSE	RMSE	NRMSE	RMSE	NRMSE
total	60min → 2h	6.68	0.093	9.31	0.130	8.95	0.125	10.39	0.145	6.25	0.087
	60min → 24h	8.98	0.125	11.46	0.155	8.68	0.103	9.79	0.136	7.86	0.109
	60min → 36h	9.00	0.125	11.80	0.164	9.28	0.129	9.75	0.136	8.02	0.136
total	15min → 2h	13.28	0.147	15.02	0.166	11.61	0.129	14.40	0.159	10.01	0.111
	15min → 24h	11.54	0.128	16.77	0.186	12.78	0.141	14.26	0.158	11.73	0.130
IV	60min → 2h	0.89	0.094	1.01	0.107	1.89	0.199	2.22	0.224	0.73	0077
	60min → 24h	1.37	0.144	1.94	0.205	1.58	0.167	1.89	0.200	1.10	0.117
	60min → 36h	1.58	0.167	1.97	0.208	1.55	0.163	1.72	0.182	1.18	0.125
IV	15min → 2h	0.98	0.097	1.06	0.104	2.16	0.213	1.73	0.171	0.81	0.080
	15min → 24h	1.99	0.197	2.17	0.215	1.78	0.176	1.70	0.168	1.16	0.115
kylmälaitteet	60min → 2h	3.23	0.087	3.69	0.099	5.68	0.153	5.96	0.160	2.98	0.080
	60min → 24h	4.32	0.116	5.18	0.139	5.03	0.135	5.78	0.156	3.86	0.104
	60min → 36h	4.43	0.119	5.31	0.143	4.51	0.121	5.77	0.155	4.01	0.108
kylmälaitteet	15min → 2h	7.76	0.158	9.85	0.200	8.57	0.174	11.01	0.224	7.03	0.143
	15min → 24h	7.93	0.161	10.17	0.207	8.48	0.172	10.86	0.221	7.8	0.158

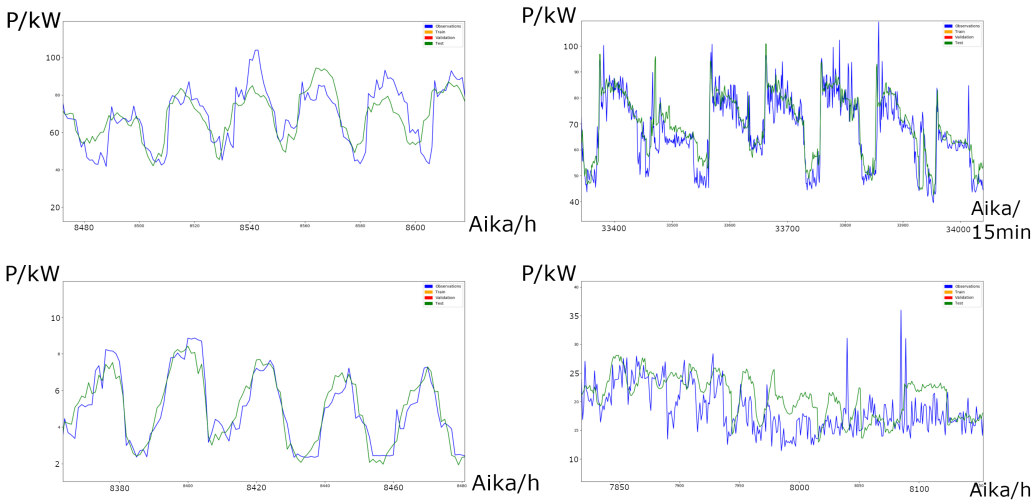
Taulukko 3: Kaupan 1 parhaiten toimivien mallien ennustetarkkuudet. Total on kaupan kokonaissähkönkulutus, IV kaupan ilmanvaihtojärjestelmän sähkönkulutus sekä kylmälaitteet kaupan kylmälaitteiden kylmänluontijärjestelmän kuluttama sähkö.

		FF		LSTM		CNN		ARIMA		SVR	
		RMSE	NRMSE	RMSE	NRMSE	RMSE	NRMSE	RMSE	NRMSE	RMSE	NRMSE
total	60min → 2h	9.51	0.085	20.34	0.181	17.48	0.156	12.05	0.107	16.95	0.151
	60min → 24h	11.65	0.104	24.45	0.218	13.75	0.123	11.64	0.104	17.41	0.155
	60min → 36h	12.06	0.108	25.37	0.226	13.93	0.104	11.66	0.104	17.74	0.158
total	15min → 2h	10.81	0.086	21.58	0.173	13.93	0.111	12.93	0.103	10.84	0.167
	15min → 24h	13.86	0.110	28.16	0.225	16.20	0.130	12.77	0.102	33.96	0.272
IV	60min → 2h	4.73	0.089	11.06	0.173	14.27	0.269	6.49	0.123	2.75	0.052
	60min → 24h	4.64	0.088	12.90	0.244	4.60	0.087	4.14	0.078	3.16	0.060
	60min → 36h	4.66	0.088	13.20	0.249	5.34	0.101	4.02	0.076	3.40	0.065
IV	15min → 2h	4.52	0.070	11.83	0.182	16.26	0.250	5.20	0.080	3.42	0.053
	15min → 24h	5.20	0.080	21.58	0.332	6.08	0.094	4.02	0.076	4.01	0.062
jääkaapit	60min → 2h	1.58	0.119	2.46	0.184	0.96	0.072	0.81	0.061	0.48	0.037
	60min → 24h	1.49	0.112	2.44	0.183	0.62	0.047	0.81	0.061	0.52	0.040
	60min → 36h	1.55	0.116	2.45	0.184	1.90	0.143	0.821	0.061	0.54	0.041
jääkaapit	15min → 2h	2.16	0.097	2.84	0.128	1.17	0.053	1.13	0.051	0.48	0.037
	15min → 24h	2.30	0.104	3.06	0.138	1.39	0.063	1.11	0.050	0.8	0.036
pakastimet	60min → 2h	3.07	0.078	6.72	0.172	5.97	0.152	3.08	0.079	2.30	0.059
	60min → 24h	3.76	0.096	10.60	0.271	4.27	0.109	3.06	0.078	2.71	0.069
	60min → 36h	4.21	0.108	10.52	0.258	3.77	0.096	3.07	0.078	2.71	0.069
pakastimet	15min → 2h	3.65	0.082	7.03	0.158	9.63	0.218	3.86	0.087	3.34	0.075
	15min → 24h	4.26	0.096	11.04	0.250	5.11	0.116	3.85	0.087	4.13	0.093

Taulukko 4: Kaupan 2 parhaiden mallien ennustustarkkuudet. Total on kaupan kokonaissähkönkulutus, IV kaupan ilmanvaihdon kuluttama sähkö sekä jääkaapit ja pakastimet ovat kaupan jääkaappien ja pakastimien kuluttama sähkö.

Parhaiten toimineiden mallien tarkastelun lisäksi mielenkiintoista on tarkastella toiseksi parhaiten toimineita malleja, sillä molempien kauppojen parhaimpia malleja oli monessa tapauksessa tukivektoriregressio. Kaupassa 1 toiseksi parhaaksi malleiksi selvisivät kokonaiskulutuksen ennustamisessa konvoluutioneuroverkko (CNN), eteenpäinsyöttävä neuroverkko (FF) sekä tukivektoriregressio (SVR). Ilmanvaihdon alimittaroinnissa toiseksi parhaita malleja olivat samalla tavalla CNN ja FF, mutta tässä alimittaroinnissa myös ARIMA oli toiseksi parhaiten ennustava malli yhdessä tapauksessa. Kylmän tuotannon alimittaroinnissa toiseksi paras malli oli luotettavasti FF.

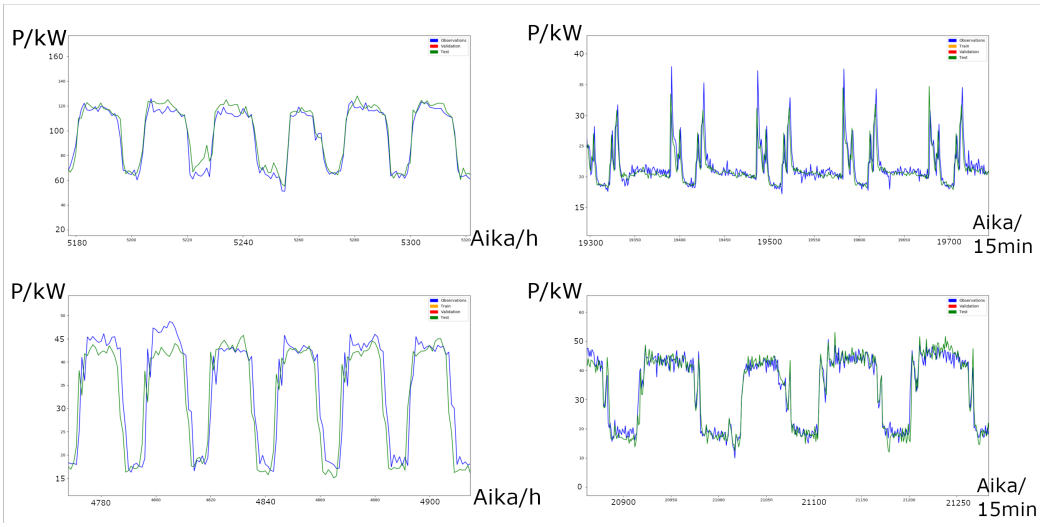
Kaupassa 2 toiseksi parhaita malleja kokonaiskulutuksen ennustamisessa olivat SVR-, ARIMA- sekä FF-mallit. Tukivektoriregressio ei siis jostain syystä onnistunut ennustamaan kaupan 2 kokonaiskulutusta kovinkaan tarkasti, sillä se ei ollut yhdessäkään tutkituista tilanteista paras malli, ja toiseksi paras malli vain yhdessä näistä tapauksista. Kaupan 2 ilmanvaihdon alimittaroinnin sähkönkulutuksen ennustamisessa FF- ja ARIMA-mallit olivat toiseksi parhaiten ennustaneita malleja. Jäähdytysalimittaroinneissa kaupan 2 toiseksi parhaiten ennustaneita malleja olivat FF-, CNN- ja ARIMA-mallit. Kaupan 2 tapauksessa ARIMA-malli oli yllättävän monta kertaa parhaiten tai toiseksi parhaiten ennustava malli verrattuna kauppaan 1, jossa ARIMA ei pärjännyt lähestulkoon ollenkaan.



Kuva 14: Käyrät kaupan 1 ennusteille ylävasemmalta alaoikealle: total 60min→24h, total 15min→2h, IV 60min→2h ja kylmälaitteet 60min→36h, missä aineiston mittausväli→ennustuksen pituus. Kuvaajien x-akseli on aika-akseli ja y-akseli energiankulutus.

Yksittäiset kuvat kuvassa 14 näyttävät muutaman kaupan 1 parhaiten

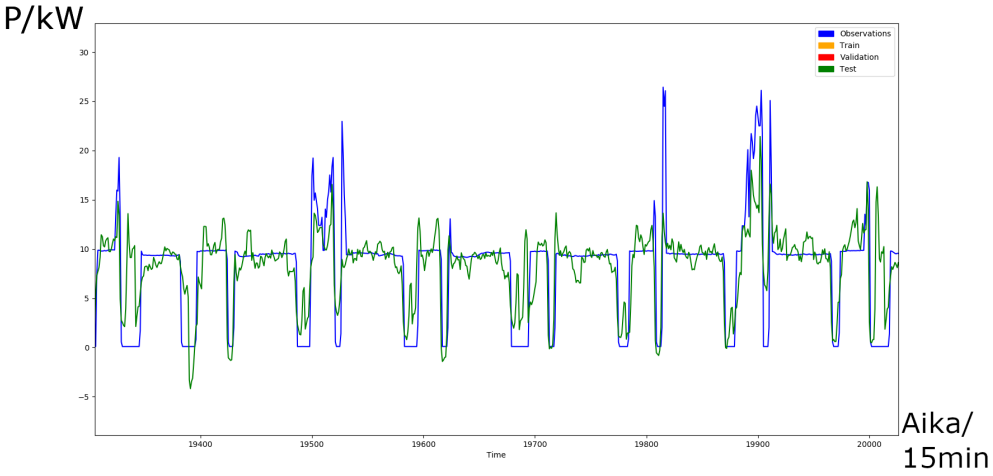
ennustavista malleista. Ylärivi kuvaa ennustuksia, jotka on tehty 60 ja 15 minuutin aineistolle 24 ja 2 tunnin ennustepituuksilla. Alarivillä ennustukset ovat molemmat 60 minuutin aineistolle, mutta 2 ja 36 tunnin ennustepituuksille.



Kuva 15: Käyrät kaupan 2 ennusteille ylävasemmalta alaoikealle: total 60min→36h, pakastimet 15min→2h, jääkaapit 60min→24h ja jääkaapit 15min→24h, missä aineiston mittausväli→ennustuksen pituus. Kuvaajien x-akseli on aika-akseli ja y-akseli energiankulutus.

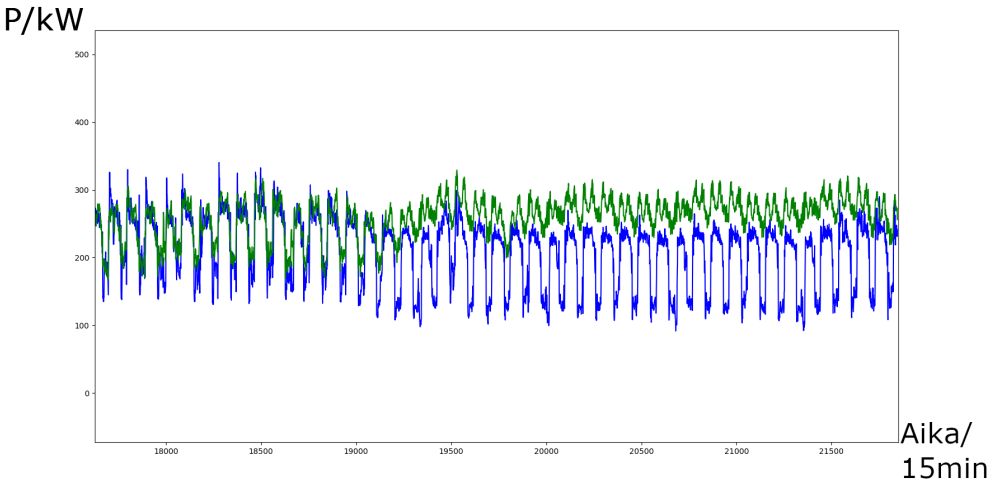
Yksittäiset kuvat kuvassa 15 näyttävät muutamia kaupan 2 parhaimmista ennustuksista. Kuvat ylävasemmalta alaoikealle ovat 60 minuutin kokonaiskulutuksen aineiston 36 tunnin ennuste, 15 minuutin jääkaappien tekemisen aineiston 2 tunnin ennuste, sekä jääkaappien 60 ja 15 minuutin aineisto 24 tunnin ennusteilla.

Monissa ennustuksissa näiden ennustusten ja oikeiden havaittujen arvojen huippukohdat osuvat eri ajankohtiin. Ennustukset myös usein arvioivat tehon liian suureksi kuin mitä se todellisuudessa on. Tämä vaikeuttaa ennustusten soveltamista esimerkiksi huippukuormien leikkaamiseen. Jotkut yritykset maksavat sähköstään huippukulutuksensa mukaan, joten jos yritykset voisivat pienentää suurinta hetkellistä kulutustaan, saataisiin sähkön kokonaishinnassa säästöjä.



Kuva 16: Ennuste 15 minuutin aineiston 2 tunnin ennustuksen parhaalle mallille kaupan 2 IV-alimittaroinnille. Kuvaajien x-akseli on aika-akseli ja y-akseli energiankulutus.

Kuten kuvasta 16 voidaan nähdä, mallit eivät oppineet ennustamaan tilannetta, jossa kulutus pysyy jonkin aikaa todella tasaisena. Tällaisissa tilanteissa ennuste pyörii oikean arvon ympärillä pitäen ennustuksen keskiarvon suunnilleen oikeana.



Kuva 17: Tukivektoriregression ennuste kaupan 2 aineistolle. Kuvaajien x-akseli on aika-akseli ja y-akseli energiankulutus.

Tukivektoriregressio toimi suhteellisen hyvin suurimmalla osasta kaupan 2 aineistoa. Kuitenkin aineiston testausosassa tukivektoriregression ennustuksen keskiarvo nousi oikeiden havaintojen keskiarvon yläpuolelle, jolloin virhe kasvoi. Kuvassa 17 havainnollistetaan tätä muutosta.

7 Pohdinta

Tutkielmassa oli tarkoituksena kokeilla, kuinka hyvin eri koneoppimismallit ja tilastolliset mallit ennustavat aikasarjoja kun niille annetaan ainoina syötteinä saman aikasarjan edellisiä arvoja. Tutkielmassa sovitettiin viiden eri malliperheen malleja neljälle aikasarjalle, jotka johdettiin kahden kaupan sähkönkulutusaineistosta. Tutkituille aikasarjoille valittiin parhaat mallit, jossa hyödynnettiin neliöidyn keskivirheen nelilöjuurta virhefunktiona. Kolmella neljästä aikasarjasta oli selvästi parhaiten suoriutunut malli, kun taas yhdellä aikasarjalla kaksi mallia toimivat yhtä hyvin. Ensimmäisen kaupan parhaiten toimineet mallit perustuivat molemmat suoran ennustuksen tukivektoriregressioon. Toisen kaupan parhaiten menestyvistä malleista kaksi oli tavallisia eteenpäinsyöttäviä neuroverkkoja ja yksi ARIMA-malli. Parhaiten toimivien mallien normalisoitu RMSE oli suhteellisen pieni, 10-13%.

ARIMA-malli suoriutui yllättävän hyvin kaupassa 2. Ennen kuin ARIMA-malli sovitettiin aineistoon, tehtiin aineistolle kaudellinen differensointi, jossa aikasarjan arvosta ajanhetkellä t vähennettiin saman aikasarjan arvo ajanhetkellä $t - s$, missä s kertoo kuinka monta mittausta tapahtuu viikossa. Ennustusten tekemisen jälkeen aikasarjaan lisättiin takaisin aikaisemmin poistetut arvot, joten aikasarjan kausittaisuus pysyi hyvin vahvasti samana kuin aikaisemminkin. Tämä voisi viitata siihen, että kaupan 2 aineistossa kausi selittää suuremman osan kaupan sähkönkulutuksesta kuin kaupassa 1. Tämä taas viittaisi siihen, että kaupan 2 sähkönkulutus on säännöllisempää kuin kaupassa 1. ARIMA-malli on myös rakenteeltaan lineaarinen ja neuroverkot pystyvät mallintamaan sekä lineaarisia, että epälineaarisia rakenteita hyvin, joka mahdollisesti viittaa siihen, että kaupan 2 aineiston rakenne on rakenteeltaan lineaarista ja kaupan 1 aineisto ei.

Tässä tutkielmassa tavoitteena oli saavuttaa mahdollisimman hyvä ennustustarkkuus hyvin yksinkertaisilla malleilla, joiden opettamisessa käytettiin vain ennustettavan aikasarjan aikaisempia arvoja. Opetettavien mallien ei ollut siis tärkeää mallintaa aikasarjojen piileviä rakenteita, vaan ennustaa mahdollisimman tarkasti. Neuroverkot voivat oppia hyvinkin monimutkaisia malleja, joten neuroverkkoa opetettaessa tärkeää on löytää arkkitehtuuri, joka on tarpeeksi leveä ja syvä, että neuroverkko voi oppia ennustamaan annettua aikasarjaa. Toisaalta, jos arkkitehtuuri on liian syvä ja leveä, kestää opetuksessa kauan aikaa. ARIMA-mallissa ennustuksen luominen taas perustuu siihen, että aikasarjan piilevistä rakenteista tiedetään jotakin. Aikasarjan rakenteen perusteella pyritään selvittämään sopiva prosessi, jonka avulla aikasarjan käyttäytymistä voidaan mallintaa ja sitä kautta myös ennustaa. Neuroverkkojen opettamisessa keskitytään pelkästään ennustuksen pätevyyteen, kun ARIMA-mallissa taas pyritään päättelemään aikasarjan tuottaneen

prosessin ominaisuuksia. Tukivektoriregressio taas on jossain neuroverkkojen ja ARIMA-mallin välillä tässä suhteessa.

8 Tuleva tutkimus

Tutkielmassa käytettyä aineistoa kerätään jatkuvasti lisää. Tulevaisuudessa voitaisiinkin tarkastella aineiston mallintamista ja ennustamista, kun aineistoa on kertynyt enemmän, jolloin voitaisiin paremmin mallintaa mm. aineiston vuositason vaihteluita. Suurempi aineisto saattaisi lisäksi auttaa parantamaan jo tässä tutkielmassakin muodostettujen mallien tarkkuutta. Analyysiin voitaisiin kokeilla myös muita koneoppimisen malleja kuten vaikka yhteismalleja (ensemble models), joissa useamman mallin ennusteet yhdistetään lopullisen ennusteen tekemiseksi. Tälläisilla yhteismalleilla tiedetään olevan erityisen hyviä ominaisuuksia.

Tässäkin tutkielmassa muodostettujen mallien ennustuksia voitaisiin myös hyödyntää muualla, kuten vaikka sähkömarkkinoilla sähkönkulutuksen ennustamiseksi. Sähkömarkkinoilla on myös mahdollista pyrkiä ennustamaan omaa sähkönkulutustaan ja siirtämään sitä ajankohtiin, jolloin sähköhinnat ovat matalemmalla. Tähän tarkoitukseen voitaisiin pyrkiä hyödyntämään esimerkiksi syvää vahvistusoppimista. Vahvistusoppimisen avulla voitaisiin luoda järjestelmä, joka käyttää ennustuksia hyväkseen sähkön kokonaishinnan alentamiseksi.

Lähdeluettelo

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro ym. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

Box, G. ja G. Jenkins (1976). *Time series analysis: forecasting and control*. Holden-Day series in time series analysis and digital processing. Oakland: Holden-Day.

Chollet, F. (2015). *Keras*. `https://keras.io`.

Fingrid (2018). *Sähkömarkkinoiden tulevaisuus*. Luettu 9.8.2018. URL: `www.sahkomarkkinoidentulevaisuus.fi`.

Gers, F. (2001). "Long short-term memory in recurrent neural networks". Tohtorinväitöskirja. Department of Computer Science, Swiss Federal Institute of Technology, Lausanne.

Goodfellow, I., Y. Bengio ja A. Courville (2016). *Deep Learning*. MIT Press. URL: `http://www.deeplearningbook.org`.

Greene, W. H. (2003). *Econometric Analysis*. Boston: Pearson Education.

Hastie, T., R. Tibshirani ja J. Friedman (2001). *The Elements of Statistical Learning*. Vol. 1. 10. New York: Springer series in statistics.

Hochreiter, S. ja J. Schmidhuber (1997). "Long short-term memory". *Neural Computation* 9 (8), s. 1735–1780.

Hsu, C.-W., C.-C. Chang, C.-J. Lin ym. (2003). *A practical guide to support vector classification*. Tekninen raportti.

Kingma, D. P. ja J. Ba (2014). "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.

Marsland, S. (2014). *Machine Learning: an algorithmic perspective*. Boca Raton: Chapman ja Hall/CRC.

McKinney, W. (2010). "Data Structures for Statistical Computing in Python". Teoksessa: *Proceedings of the 9th Python in Science Conference*. Toim. S. van der Walt ja J. Millman, s. 51–56.

Python Software Foundation (ei julkaisupäivää). *Python Language Reference, version 3.6*. `http://www.python.org`.

Rojas, R. (2013). *Neural networks: a systematic introduction*. New York: Springer Science & Business Media.

Seabold, S. ja J. Perktold (2010). "Statsmodels: Econometric and statistical modeling with python". Teoksessa: *9th Python in Science Conference*. Toim. S. van der Walt ja J. Millman, s. 92–96.

Smola, A. J. ja B. Schölkopf (2004). "A tutorial on support vector regression". *Statistics and Computing* 14 (3), s. 199–222.

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.