Jarno Kiesiläinen

Koneoppimisen hyödyntäminen konenäössä

Tietotekniikan kandidaatintutkielma

20. joulukuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Jarno Kiesiläinen

Yhteystiedot: jarno.e.kiesilainen@student.jyu.fi

Työn nimi: Koneoppimisen hyödyntäminen konenäössä

Title in English: Machine Learning in Computer Vision

Työ: Kandidaatintutkielma

Sivumäärä: 24+0

Tiivistelmä: Konenäön hyödyntäminen yleistyy ja sitä mukaa myös konenäön ongelmat monimutkaistuvat. Yksi suosittu tapa ratkaista näitä ongelmia on hyödyntää koneoppimista. Tässä tutkielmassa tarkastellaan miten koneoppimista hyödynnetään konenäössä ja vertaillaan eri koneoppimisalgoritmeja konenäön näkökulmasta.

Avainsanat: konenäkö, koneoppiminen, neuroverkot, hermoverkot, tukivektorikone, boosting

Abstract: Computer Vision faces increasing challenges as its used more. Common way to solve these complex problems is to use Machine Learning. In this thesis workings of different Machine Learing algorithms are looked on and their advantages and disadvantages are compared.

Keywords: computer vision, machine vision, machine learning, neural network, support vector machine, boosting

Kuviot

Sisältö

1 Johdanto

Automaatio lisääntyy kaikessa teknologiassa ja tästä johtuen tilanteet, joissa tarvitaan visuaalista päättelyä lisääntyvät. Esimerkiksi itsestään ajavat autot tarvitsevat tehdä nopeaa päättelyä tienreunojen ja jalankulkijoiden liikkeen perusteella. Myös laitteiston ja kameroiden hintojen laskeminen luo mahdollisuuksia käyttää konenäköä yhä useammassa paikassa. Konenäköä käytetään mm. tehtaissa, robotiikassa ja digitaalisissa kameroissa.

Perinteisemmät konenäkömenetelmät ovat tehneet valmiiksi ohjelmoitua päättelyä kuvasta löytyvien reunojen ja kulmien perusteella. Kontrolloidussa ympäristössä, kuten liukuhihnalla, valaistus ei muutu ja esineet ovat hyvin samankaltaisia. Tälläisissä tilanteissa perinteisemmät menetelmät toimivat hyvin. Siirryttäessä esimerkiksi oikeaan liikenteeseen tilanteet saattavat olla konenäön näkökulmasta hyvin erilaisia. Valaistus voi muuttua sään ja vuorokauden ajan mukaan sekä tunnistettavat esineet, kuten autot, voivat olla osaksi peittyneitä esimerkiksi toisten autojen taakse (Buch, Velastin ja Orwell 2011). Monimutkaisten tilanteiden lisäksi kuvissa saattaa olla häiriötä tai niiden laatu voi olla muuten huono. Cenev ym. (2016) käyttivät koneoppimista tunnistaakseen mikrosirun paikan sumeasta kuvasta. Perinteinen reunantunnistus toimi hyvin tilanteissa missä mikrosiru pystyttiin kuvaamaan suoraan ylhäältä, mutta sivulta otetuissa kuvissa osa sirusta jää sumeaksi kameran lyhyen tarkennusvälin takia. Tällöin reunantunnistus ei pystynyt löytämään sumeita reunoja, mutta AdaBoost-algoritmilla opetettu luokittelija löysi mikrosirun sijainnin noin 99,9 prosentin tarkkuudella.

Useat konenäkömenetelmät hyödyntävät koneoppimista. Koneoppimisessa on tarkoitus luoda yleistäviä malleja tunnistettavista asioista. Esimerkiksi digikameroiden kasvojentunnistuksessa yksinkertaisia piirteitä on yhdistetty koneoppimisella, jotta tunnistus tapahtuu nopeasti ja hyvällä tarkkuudella. Tämän tutkielman tarkoitus on perehtyä eri konenäköön hyödynnettävien koneoppimismenetelmien toimintaan ja vertailla niitä. Luvuissa 2 ja 3 käsitellään konenäköä ja koneoppimista sekä niihin liittyviä käsitteitä ja ongelmia. Luvussa 4 perehdytään valittujen koneoppimismene-

telmien toimintaan ja tarkastellaan niiden käyttöä konenäössä. Viimeisenä luvussa 5 valittuja menetelmiä vertaillaan niiden vahvuuksien ja heikkouksien perusteella.

2 Konenäkö

Liu ja Cho (2011) määrittelevät konenäön olevan ”esineestä tai tilasta otetun kuvan tulkitsemista informaation saavuttamiseksi”. Konenäön pidempiaikainen tavoite onkin matkia ihmissilmää ja tietyissä tilanteissa koneet voivat olla ihmistä parempia (Fernandes ym. 2011). Tyypillisiä tehtäviä ovat asioiden luokittelu, liikkeentunnistus ja seuraaminen. Luokittelulla tarkoitetaan aineiston jakamista ryhmiin. Nykyään konenäköä käytetään esimerkiksi digikameroiden kasvojentunnistuksessa ja itsestään ajavissa autoissa.

Luultavasti tunnetuimpia perinteisiä konenäköalgoritmeja ovat ”Harris Corner Detector” ja ”Canny Edge Detector”. Useat algoritmit etsivät kuvistä avainpisteitä (eng. interest point). Esimerkiksi SIFT-algoritmin tarkoitus on löytää samasta kohteesta otetusta kahdesta eri kuvasta samoja avainpisteitä, jolloin voidaan tunnistaa kohteen paikka kuvissa (Lowe 1999).

Konenäkömenetelmät koostuvat usein kolmesta osasta: kuvankaappaus, prosessointi ja tuloste. Kuvankaappaus tarkoittaa digitaalisen kuvan siirtämistä kamerasta konenäköjärjestelmään. Ennen varsinaista prosessointia voidaan suorittaa esiprosessointia, johon voi kuuluua esimerkiksi kohinan poisto ja värikanavien yhdistämistä mustavalkokuvaksi. Prosessointivaiheessa kuvasta etsitään informaatiota, jonka perusteella tehdään päätöksiä tulostetta varten (Fernandes, Moreira ja Mata 2011). Koneoppimista hyödynnetään varsinkin korkeamman tason prosessointiin, kuten muotojen ja kokonaisuuksien löytämiseen, sekä lopulliseen päätöksen tekoon.

3 Koneoppiminen

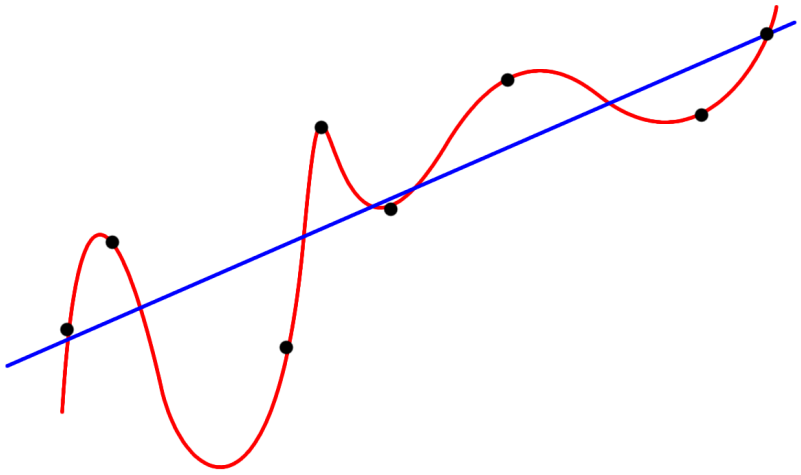
Koneoppiminen on työkalu, jonka tarkoitus on automatisoida päätöksentekoa datasta löytyvien mallien perusteella. Koneoppiminen on erityisen hyödyllistä jos dataa on paljon. Tällöin opitut mallit voivat olla tarkempia. Koneoppiminen jakaantuu kahteen kategoriaan: ohjattu oppiminen ja ohjaamaton oppiminen (Murphree 2016).

Ohjatussa oppimisessa järjestelmä ”opetetaan” opetusainestolla ennen kuin sitä käytetään oikeassa ympäristössä. Opetusaineisto on esimerkiksi valmiiksi luokiteltuja kuvia, joilla järjestelmän oppimista ohjataan (Sebastiani 2002). Opetusaineistosta jätetään usein käyttämättä pieni osa, jolla testataan (validoidaan) opittua luokittelijaa. Tällöin kuitenkin osa opetusaineistosta jää käyttämättä. Siksi usein käytetään ristivalidointia. **Ristivalidoinnissa** opetusaineisto jaetaan satunnaisesti esimerkiksi 10:een osaan ja opetteluvaiheessa yksi osista jätetään validointiin, muilla harjoitellaan. Opettelua toistetaan ja validointiin jätetään aina eri osa opetusaineistosta (Domingos 2012). **Ohjaamattomassa oppimisessa** opetusaineistoa ei ole luokiteltu valmiiksi vaan luokat etsitään datasta luonnollisten ryhmien ja klusterien perusteella (Jain ym. 2000).

Useat koneoppimisalgoritmit tuottavat luokittelijoita. **Luokittelija** on kuvaus, joka luokittelee tapauksia diskreettiin määrään luokkia. Luokittelija voi myös antaa todennäköisyyksiä tai pisteitä tapaukselle (Kohavi 1998). Yksinkertainen luokittelija voi olla esimerkiksi sellainen, joka kertoo onko kuvassa jotakin esinettä vai ei. Tälläistä luokittelijaa kutsutaan binääriluokittelijaksi (binary classifier). Binääriluokittelijan vastakohta on moniluokittelija (multiclassifier), joka siis pystyy luokittelemaan useampaan kuin kahteen luokkaan. Konenäön lisäksi koneoppimista voidaan käyttää myös muunlaisten ongelmien ratkaisuun, esimerkiksi puheentunnistukseen, sairauksien diagnosointiin tai peleihin (Mohri, Rostamizadeh ja Talwalkar 2012).

3.1 Koneoppimisen ongelmat

Ylisovitus (eng. overfitting) on yleinen ongelma monissa koneoppimismenetelmissä. Ylisovituksella tarkoitetaan sitä, kun opittu malli antaa hyviä tuloksia opetusaineistolla, mutta toimii oikealla datalla heikosti. Tämä kertoo siitä, ettei malli yleistä koko olemassa olevaa dataa vaan se on löytänyt jonkin relaation, joka esiintyy ainoastaan opetusaineistossa (Gonçalves ja Silva 2013). Kuviossa 1 punainen polynomi on sovitettu täydellisesti datapisteisiin, mutta sininen suora yleistää dataa luultavasti paremmin.



Kuvio 1. Ylisovitus

Useimmat koneoppimisalgoritmit käyttävät jonkinlaisia **ominaisuusvektoreita**. Konenäön näkökulmasta tämä tarkoittaa, että kuva täytyy muutta vektoriksi, joka sisältää jotakin ongelman kannalta tärkeää informaatiota kuvasta. Vektorien muodostamiseen ei välttämättä ole yhtä oikeaa tai parasta tapaa. Suosittuja tapoja ovat esimerkiksi SIFT-algoritmin avainpisteet ja pääkomponenttianalyysi (PCA), sekä HOG-vektorit.

4 Konenäköön soveltuvia koneoppimisalgoritmeja

Seuraavissa luvuissa esitellään koneoppimismenetelmiä joita voidaan hyödyntää konenäössä. Menetelmien toiminnan selittämiseen on valittu jokin yksinkertainen versio, joka kuitenkin sisältää menetelmän kannalta tärkeät ideat. Kaikki valitut menetelmät ovat ohjattua oppimista, geneettisiä algoritmeja lukuun ottamatta.

4.1 Boosting

Boosting-menetelmissä ideana on ottaa useita heikkoja luokittelijoita h_t ja yhdistää ne vahvaksi luokittelijaksi h. Heikolla luokittelijalla tarkoitetaan sellaista luokittelijaa, joka on vähintään puolessa tapauksista oikeassa. Heikot luokkitelijat yhdistetään yhdeksi luokittelijaksi ottamalla niistä lineaarikombinaatio:

$$h(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \\ 0, & \text{muulloin} \end{cases}$$

missä h vahva luokittelija, h_t heikko luokittelija ja α_t luokittelijan painotus (Schapire 1990).

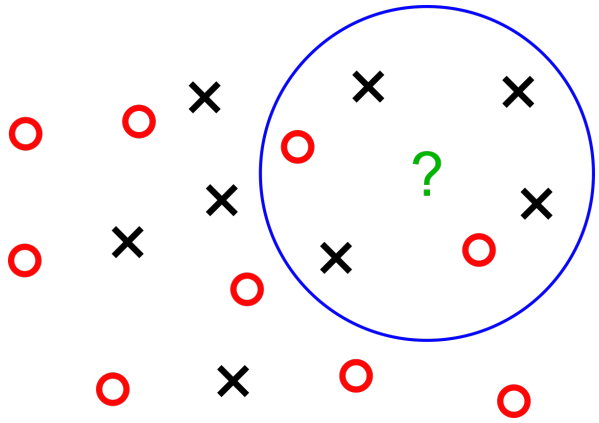
Boosting-algoritmien oppiminen tarkoittaa heikkojen luokittelijoiden opettamista ja niiden painotusten määrittämistä. Menetelmässä voidaan esimerkiksi käyttää lähimpien naapurien luokittelijoita tai useaa eri luokittelija-algoritmia (Kittler, Hatef, Duin ja Matas 1998).

Viola ja Jones (2001) esittelivät algoritmin, jolla tunnistetaan kuvasta kasvot. Algoritmissa yksinkertaisten ominaisuuksien avulla päätellään 24x24 pikselin ikkunasta onko siinä kasvot. Ominaisuuksiksi valittiin joukko (n. 180 000 kpl) Haar-aaltosiin perustuvia suorakulmio-ominaisuuksia. Sen jälkeen ominaisuuksia koulutettiin AdaBoost-algoritmilla ja 200 parhaiten suoriutuvaa ominaisuutta valittiin lopulliseen luokittelijaan. Ominaisuuksista muodostetaan Cascading-luokittelija, joka koulutetaan myös AdaBoost-algoritmilla. Cascading luokittelijan ideana on, että jo-

kaisesta ikkunasta ei tarkisteta kaikkia ominaisuuksia vaan selkeät kasvottomat ikkunat tunnistetaan muutamalla yksinkertaisella ominaisuudella. Jos ikkunassa kuitenkin vaikuttaa olevan kasvot, niin monimutkaisempia ominaisuuksia käytetään, kunnes tullaan päätökseen onko ikkunassa kasvot vai ei. Algoritmi soveltuu myös muiden asioiden tunnistukseen, kuten ihmishahmon tunnistukseen. Se on kuitenkin suunniteltu kasvojen tunnistukseen ja on siinä tehokas.

4.2 K:n lähimmän naapurin menetelmä

K:n lähimmän naapurin menetelmä on luultavasti yksinkertaisin luokittelualgoritmi. Menetelmässä pisteen luokaksi valitaan sitä lähinnä olevista pisteistä niiden yleisin luokka. Kuviossa 2 lähimpien naapurien määräksi on valittu 6. Tällöin tuntematon piste (*kysymysmerkki*) luokiteltaisiin x:ksi.



Kuvio 2. k:n lähimmän naapurin menetelmä

Menetelmän käyttö on nykyisin vähäistä konenäköongelmissa, koska se ei usein pärjää tarkkuudessa muille algoritmeille kuten tukivektorikoneet, boosting-algoritmit ja päätöspuut. K:n lähimmän naapurin menetelmällä on silti useita etuja muihin algoritmeihin verrattuna. Ensinnäkään se ei tarvitse erillistä opetteluvaihetta vaan luokittelua voidaan tehdä suoraan opetusaineiston perusteella. Tämä tarkoittaa ettei järjestelmää tarvitse opettaa uudestaan, jos dataa tulee lisää tai se muuttuu. Koska menetelmään ei kuulu opetusvaihetta ei se siksi pysty ylisovittumaan. Luokkien määrä ei myöskään vaikuta algoritmin toimintaan ja siksi se soveltyy yhtä hyvin

binääri-luokitteluun kuin moniluokitteluun (Boiman, Shechtman ja Irani 2008).

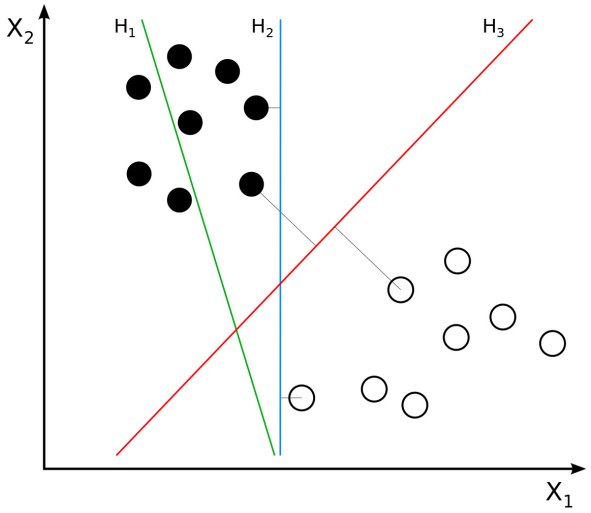
Keller ym. (1985) esittivät algoritmin toiminnan seuraavasti:

1. y tuntematon luokiteltava
   $W = \{x_1, x_2, \ldots, x_n\}$ n kappaletta opetteludatapisteitä
   $K : 1 \leq K \leq n$ lähimpien naapurien määrä
2. Laske jokaiselle $x_i \in W$ etäisyys pisteeseen y.
3. Valitse K lähintä pistettä.
4. y:n luokka on valittuissa pisteissä useimmin esiintyvä luokka. Jos syntyy tasatulos, näiden luokkien pisteiden etäisyyksien summa lasketaan ja pienimmän summan saanut luokka valitaan y:n luokaksi.

Algoritmin toiminnan kannalta on erittäin tärkeää miten kahden pisteen $x, y \in \mathbb{R}^n$ välinen etäisyys mitataan. Yksinkertaisin tapa selvittää etäisyys, on laskea pisteiden välinen euklidinen etäisyys $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$. Euklidinen etäisyys ei kuitenkaan useimmissa tilanteissa tuota erityisen hyviä tuloksia. Jokin hyvinkin yksinkertainen muunnos etäisyysmittaan, jossa otetaan huomioon opetusaineisto ja siinä olevia relaatioita, voi vaikutta suuresti algoritmin tarkkuuteen. Weinbergeri ym. (2005) esittelivät algoritmin, jossa opetusaineiston perusteella opetetaan etäisyysmitta siten, että K lähintä pistettä kuuluvat mahdollisimman hyvin samaan luokkaan ja että eri luokkien pisteet olisivat erotettu mahdollisimman kauas toisistaan. Koulutettua etäisyysmittaa käyttävä klassifikoija selvisi esimerkiksi kasvojentunnistustehtävästä euklidista etäisyyttä käyttävää huomattavasti paremmin.

4.3 Tukivektorikoneet (SVM)

Tukivektorikone (Support Vector Machine) yrittää jakaa luokitellut datapisteet mahdollisimman hyvin kahteen luokkaan. Tukivektorikone etsii hypertason siten, että tason kummallakin puolella on mahdollisimman paljon vain yhteen luokkaan kuuluvia datapisteitä (Pontil ja Verri 1998). Kuviossa 3 mallidataa on jaettu kolmella eri hypertasolla. Hypertaso H_1 ei jaa joukkoa luokkiin. H_2 jakaa, mutta ei niin, että etäisyys luokkiin olisi maksimi. Hypertaso H_3 jakaa luokat maksimietäisyydellä toi-

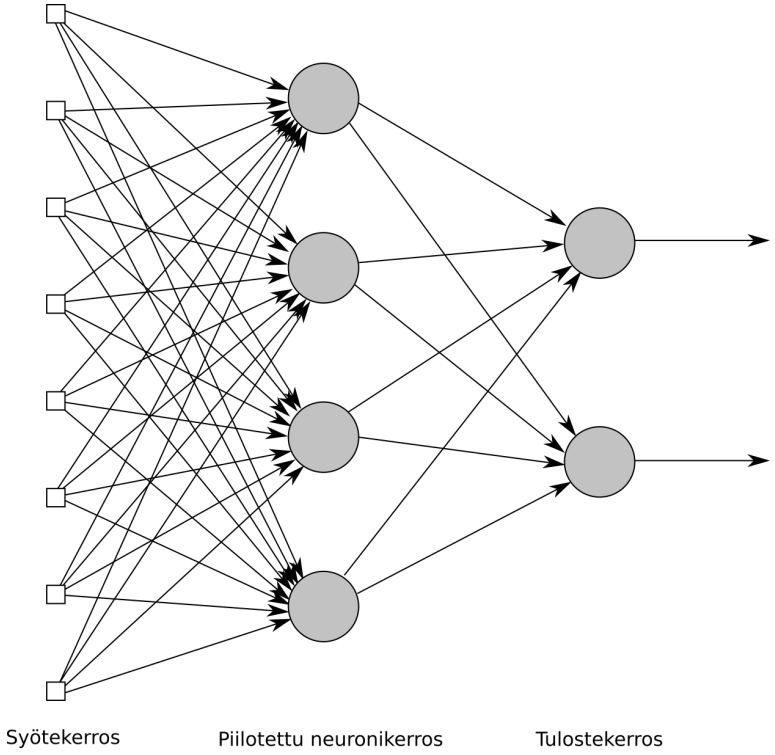


Kuvio 3. Tukivektori koneen tuottamia hypertasoja
Lähde: https://en.wikipedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg

siinsa nähden. Tukivektorikone soveltuu parhaitan tapauksiin, jossa datalle tehdään binääriluokitus. Tukivektorikoneita on mahdollista soveltaa useamman luokan tapauksiin esimerkiksi jakamalla ongelman useampaan binääriluokitteluongelmaan tai muokkaamalla algoritmia siten, että se pystyy jakamaan tietoa useampaa luokkaan (Hsu ja Lin 2002).

Tukivektorikoneita on käytetty erilaisten neurologisten sairauksien diagnosointiin. Esimerkiksi Alzheimerin ja normaalin vanhenemisen aiheuttaneet oireet pystyttiin erottamaan yli 90 %:n tarkkuudella potilaiden MRI- ja PET-kuvien perusteella. Samaa tekniikka on käytetty myös masennuksen tunnistamiseen (Orrùa ym. 2012). Suunnattujen gradienttien histogrammin (eng. Histogram of Oriented Gradients) käyttö on yleistä tukivektori koneideen kanssa. HOG-vektori sisältää tiedon kuvassa olevien gradienttien kulmista ja niiden voimakkuuksista. Gradientit voidaan ymmärtää reunoina ja tällä tavalla HOG-vektoroita voi käyttää siluettien löytämiseen. HOG-vektorien ja SVM:n yhdistelmää onkin käytetty muun muassa ihmisten (Bertozzi ym. 2007), käsieleiden (Feng ja Yuan 2013) ja autojen tunnistamiseen (Kang, Seok-Il ja Kyeong-Hoon 2016).

4.4 Konvoluutioneuroverkot

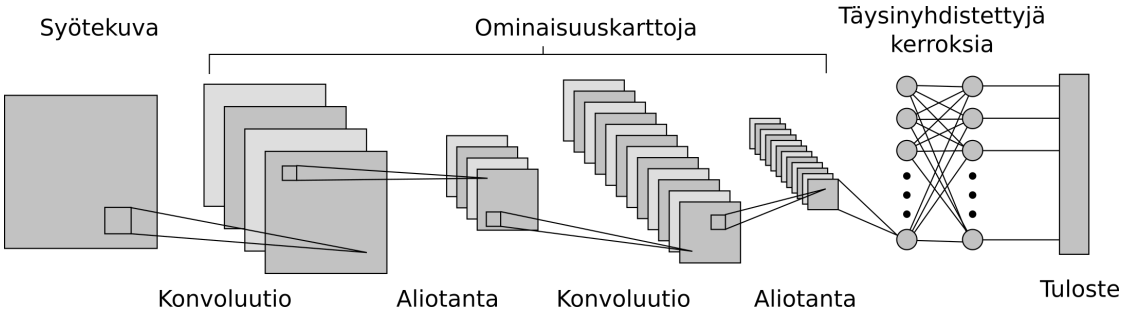


Kuvio 4. Täysin yhdistetty eteenpäin kytketty neuroverkko

Konvoluutioneuroverkot nousivat erityiseen suosioon kuvaluokittelussa, kun syvä konvoluutioneuroverkko voitti ”Imagenet Large Scale Visual Recognition Challenge” -kilpailun ja paransi edellisten vuosien tuloksia huomattavasti. Kilpailussa kuvia luokitellaan yhteen tuhannesta luokasta. Voittavassa neuroverkossa oli 60 miljoonaa muuttujaa ja 650,000 neuronia (Krizhevesky ym. 2012).

Inspiraatio neuroverkkojen toimintaan on saatu ihmisaivojen erilaisesta tavasta laskea tietokoneisiin verrattuna. Neuroverkot koostuvat solmuista, joita kutsutaan neuroneiksi. Neuronit muodostavat kerroksia, joissa edellisen kerroksen neuronien tulosteet on yhdistetty seuraavan kerroksen neuronien syötteeseen. Neuroverkko sisältää vähintään syötekerroksen ja neuronien tulostekerroksen. Monikerros neuroverkossa on myös piilotettuja kerroksia, jotka yhdistävät syöte- ja tulostekerrokset. Kuviossa 4 on täysin yhdistetty monikerrosneuroverkko, jossa on yksi piilotettu kerros. On myös olemassa takaisinkytkettyjä neuroverkkoja (eng. recurrent neural-

network), missä neuronit on yhdistetty silmukoiksi. Konvoluutioneuroverkot ovat kuitenkin eteenpäin kytkettyjä neuroverkkoja, jossa neuroninen yhteydet ovat vain yhteen suuntaan (Haykin 1998).



Kuvio 5. Yksi mahdollinen konvoluutio neuroverkon rakenne

Yksittäinen neuroni sisältää syötteitä ja jokaiselle syötteelle painotuksen, harhan (eng. bias), summausfunktion ja aktivointifunktion. Summausfunktion tehtävä on yhdistää syötteet ja aktivointifunktion tarkoitus on rajoittaa tulostetta ettei neuroni aktivoidu liian pienellä syötteellä (Haykin 1998).

Konvoluutioneuroverkko on monikerroksisen eteenpäin kytketyn neuroverkon erikoistapaus ja se on kehitety kaksiulotteisten muotojen tunnistamiseen. Neuroverkko koostuu konvoluutio-, aliotanta- (eng. subsampling layer) ja täysinyhdistetyistä kerroksista (kuvio 5). Konvoluutikerros koostuu ominaisuuskartoista. Yhdessä ominaisuuskartassa on jonkin ominaisuuden sijainti kuvassa. Ominaisuudet ovat esimerkiksi 4x4 pikselin kokoisia ikkunoita. Aliotantakerroksessa ominaisuuskartan resoluutiota pienennetään maksimikeräys (eng. max pooling) menetelmällä. Menetelmässä liikutetaan ikkunaa ominaisuuskartan yli ja jokaisessa kohdassa ikkunasta valitaan suurin arvo. Aliotanta vähentää laskentaa myöhemmissä kerroksissa ja herkkyyttä ominaisuuksien siirtymälle, kuitenkin säilyttäen löytyneiden ominaisuuksien sijainnin ja määrän. Verkon lopussa on usein täysinyhdistettyjä kerroksia, jotka toimivat normaalin neuroverkon tapaan (Krizhevsky, Sutskever ja Hinton (2012), Haykin (1998)).

Neuroverkkoja opetetaan vastavirta-algoritmilla (eng. backpropagation). Vastavirta-algoritmissa verkon tekemä virhe jollakin opetusaineiston alkiolla mitataan. Sen jäl-

keen neuroneita muutetaan liikkumalla verkossa takaperin ja muuttamalla niiden painotuksia pienellä määrällä niin, että luokitteluvirhe pienenee (Haykin 1998).

Neuroverkkojen ongelmana on, että ne ylisovittuvat helposti opetusaineistoon. Ylisovittamista vastaan on kehitetty useita tapoja kuten solmujen karsiminen, ristivalidointi ja useamman neuroverkon käyttö siten, että ne varmentavat toistensa tuloksia (Zhang 2000). Neuroverkkojen kouluttaminen voi olla myös laskennallisesti raskasta. Krizhevesky ym. (2012) joutuivat käyttämään opetusvaiheessa useampaa näytönohjainta, koska heidän verkkonsa oli niin monimutkainen.

4.5 Päätöspuut

Päätöspuu on rakenne, joka koostuu sisä- ja lehtisolmuista. Jokaiseen sisäsolmuun liittyy testi, josta voidaan siirtyä kahteen tai useampaan solmuun. Lehtisolmuun liittyy ennuste puulle annetusta syötteestä. Puulle annetaan siis syöte ja sisäsolmuissa testeillä päätetään mihin alipuuhun siirrytään. Tätä jatketaan kunnes päästään lehtisolmuun, joka määrää puun ennusteen. Koneoppimisen kannalta ongelmana on muodostaa hyvä päätöspuu opetusaineiston perusteella. Hyvällä puulla tarkoitetaan usein puuta, joka antaa oikeita ennusteita mahdollisimman pienellä solmujen määrällä. Tällä tavalla ennusteeseen päästä mahdollisimman vähillä testeillä.

Quinlan (1986) esitteli iteratiivisen algoritmin nimeltä ID3, joka yrittää rakentaa mahdollisimman yksinkertaisen puun. Algoritmin toiminta on seuraava: on olemassa attribuutteja A, jotka jakavat datajoukon C osiin $C_1, C_2, \ldots, C_n$. Mitataan jokaisen attribuutin A jaon informaation lisäystä joukkojen entropiaa mittaavalla funktiolla. Valitaan sisäsolmun jakavaksi attribuutiksi se attribuutti A, jonka informaation lisäys on suurin. Tehdään sama jokaiselle syntyneelle osajoukolle $C_1, C_2, \ldots, C_n$. Jos joukossa C on vain yhden luokan edustajia, niin tehdään kyseistä luokkaa edustava lehtisolmu.

Päätöspuut ovat herkkiä ylisovitukselle ja syvemmät puut ylisovittuvat matalia helpommin. Ylisovittumista on mahdollista estää yhdistelemällä puita esimerkiksi satunnaisiksi metsiksi (eng. random forest) tai käyttämällä Boosting-algoritmia. Kone-

näköongelmissa käytetään harvemmin yksittäisiä päätöspuita, koska yhdistelevillä menetelmillä saadaan varmempia tuloksia. Päätöspuita käytetään paljon lääketieteellisten kuvien analysointiin, mutta niitä voi käyttää tavalliseen kuvienluokitteluun kuten esimerkiksi kehonosien tunnistukseen (Norouzi, Collins, Johnson, Fleet ja Kohli 2015).

4.6 Geneettiset algoritmit

Geneettiset algoritmit ovat heuristiikka löytää riittävän hyvä ratkaisu, jos täydellisen ratkaisun löytäminen ei ole välttämätöntä. Geneettisiä algoritmeja voidaan soveltaa esimerkiksi laskennallisesti raskaisiin ongelmiin, mutta ne sovultavut myös luokitteluongelmiin (Reeves 1995). Algoritmin toiminta perustuu evoluutioon jossakin ratkaisujen perusjoukossa. Yhtä ratkaisua kutsutaan kromosomiksi ja koko ratkaisujen joukkoa populaatioksi. Kromosomi on vektorimuotoon koodattu esitys jostakin ratkaisusta. Algoritmissa siirryttään iteratiivisesti populaatiosta toiseen ja iteraatioita kutsutaan sukupolviksi (eng. generation).

Geneettisistä algoritmeista on useita versioita, mutta eräs yksinkertainen algoritmi on staattinen populaatio (eng. static population). Chtioui, Bertrand ja Barba (1998) esittelivät algoritmin toiminnan seuraavasti:

1. Valitse kromosomille esitys ja sopivuusfunktio (eng. fitness function).
2. Luo aloituspopulaatio.
3. Arvioidaan kaikki koromosomit sopivuusfunktiolla.
4. Valitaan kaksi kromosomia vanhemmiksi. Paremmilla kromosomeilla on suurempi todennäköisyys tulla valituksi.
5. Risteytetään vanhemmat ja mutatoidaan saatuja lapsia.
6. Lisätään lapset populaatioon ja poistetaan kaksi huonoimmin suoriutuvaa populaatiosta. Palataan kohtaan 4.

Risteyttäminen on tärkein tapa löytää uusia kromosomeja. Risteytys tehdään valitsemalla satunnainen leikkauskohta vanhemmista. Esimerkiksi jos leikkaus olisi 5. bitin jälkeen:

Vanhemmat: 01100**111** ja 11100**100**
Lapset saavat osan kummastakin vanhemmasta: 01100**100** ja 11100**111**.

Mutaatiolla tarkoitetaan kromosomiin tehtävää pientä satunnaista muutosta. Binäärikromosomissa mutaatio on yhden satunnaisen bitin muuttaminen 1:stä 0:ksi tai toisin päin. Mutaation tarkoitus on estää populaation samankaltaistumista ja estää algoritmin liian aikainen konvergoituminen johonkin ratkaisuun. Mutaatio tapahtuu asetetun mutaatiotodennäköisyyden mukaan, joka on yleensä matala.

Hu, Yu, Li ja Ma (2000) käyttivät kerroksittaista geneettistä algoritmiä tunnistaakseen ihmisen asennon videokuvasta. Ihmisen siluetti irotettiin kuvasta ja siitä tunnistettiin torso, pää ja raajat erikseen koulutetuilla geneettisillä algoritmeilla. Centeno, Lopes, Felisberto ja Ramos de Arruda (2005) taas käyttivät geneettistä algoritmiä löytääkseen esineen paikan, koon ja asennon kuvasta. Valmis luokittelija pystyi tunnistamaan esineen vaikka se olisi peittynyt tai osin kuvan ulkopuolella.

Useimmissa koneoppimismenetelmissä vaadittavien ominaisuusvektorien valinta on ongelma. Chtioui ym. (1998) yrittivät tunnistaa kasvinsiemenien lajin kuvasta lähimmän naapurin menetelmällä, mutta algoritmia varten tarvitaan ominaisuuksia ominaisuusvektorin muodostamiseen. Mahdollisia ominaisuuksia siementen tunnistamiseen oli niiden kokoon, muotoon, väreihin ja kuvioihin liittyen useita erilaisia. Käytännössä kaikkien ominaisuuksien käyttäminen ei ollut mahdollista, joten he käyttivät geneettistä algoritmia parhaiden ominaisuuksien kombinaation valitsemiseen.

5 Algoritmien vertailua

Konenäköongelmaan tai tilanteeseen sopivan koneoppimismenetelmän valitseminen ei aina ole yksinkertaista. Kaikilla menetelmillä on omat vahvuutensa ja heikkoutensa. Tällä hetkellä neuroverkot, tukivektorikoneet ja boosting-algoritmit ovat suosiossa niiden hyvien tulosten takia. Nämä menetelmät kykenevät muodostamaan monimutkaisempia malleja ja se johtaa herkästi ylisovittumiseen, kun taas yksinkertaisemmilla menetelmillä etuna on selkeästi niiden toiminnan ymmärrättevyys. Valmis neuroverkko voi tuottaa hyviä tuloksia, mutta sen sisäinen toiminta on mustalaatikko, josta on vaikeaa päätellä miksi se toimii niin hyvin. Kaikissa menetelmissä on jotakin parametrejä, jotka jäävät ihmisen päätettäväksi. K:n lähimmän naapurin menetelmässä sellainen parametri on k eli tarkasteltavien naapurien määrä ja geneettisissä algoritmeissa päätettäväksi jää esimerkiksi populaation koko ja iteraatioiden määrä. Neuroverkoissa hyperparametrit voivat muodostaa ongelman, koska verkon kerroksien ja neuronien määrään sekä konvoluutioverkoissa eri kerrosten järjestyksen päättämiseen ei ole mitään algoritmia. Hyvät arvot riippuvat myös tilanteesta, johon menetelmää sovelletaan tarkoittaen, että hyviä arvoja voi löytää vain kokeilemalla (Maglogiannis (2007), Kotsiantis (2007)).

Opetusaineiston määrä vaikuttaa lopullisen luokittelijan tarkkuuteen. Toiset menetelmät kuten tukivektorikoneet ja neuroverkot tarvitsevat erityisen paljon opetusaineistoa ollakseen tarkkoja. Osa menetelmistä pystyy myös iteratiivisineen oppimiseen eli opetusaineistoa voi lisätä myöhemmin ilman, että koko opetusvaihe täytyy toistaa alusta asti uudelleen. Tälläisiä menetelmiä ovat esimerkiksi neuroverkot ja k:n lähimmän naapurin menetelmä, kun taas tukivektorikoneissa ja päätöspuissa uuden aineston hyödyntäminen tarkoittaa käytännössä opetusvaiheen uusimista (Maglogiannis (2007), Kotsiantis (2007)).

Boiman, Shechtman ja Irani (2008) kiistelevät lähimmän naapurin menetelmän puolesta. He perustelivat menetelmän huonompia tuloksia konenäköongelmissa kuvien vektorimuotoon muuttamisessa tapahtuvasta informaation häviämisestä. Koska menetelmä on niin yksinkertainen, se ei kykene kompensoimaan informaation

häviämistä kuten monimutkaisemmat menetelmät. He ehdottivat algoritmia, jolla kuvien välistä etäisyyttä mitataan suoraan ja muokattu menetelmä tuotti huomattavasti parempia tuloksia. Lähimmän naapurin menetelmän etuna on sen hyvä soveltuvuus moniluokitteluun ja laskennallinen keveys. Toisaalta menetelmä voi vaatia huomattavan määrän muistia laitteelta vaatiessaan koko opetusaineiston säilyttämistä. Menetelmä on myös herkkä virheille aineistossa.

Cracknell ja Reading (2014) vertailivat eri koneoppimismenetelmiä geologisen kartoituksen tekemiseen ilma- ja satelliittikuvien perusteella. Heidän vertailussaan päätöspuihin perustuvat satunnaiset metsät tuottivat parhaita tuloksia päihittäen neuroverkot ja tukivektorikoneet. Satunnaiset metsät olivat myös heidän mukaansa helppokäyttöisempiä ja siksi he suosittelivat satunnaisia metsiä kuvien perusteella tehtävään geoligiseen kartoitukseen.

Algoritmin tai menetelmän valintaan voi vaikuttaa useampi asia. Joissakin tapauksissa tarkin menetelmä riittää, mutta toisaalta taas käytettävä laite tai laitteet voivat asettaa rajoituksia menetelmälle esimerkiksi muistin tai prosessorin tehokkuuden puolesta. Yksittäisten menetelmien ongelmia voi kiertää käyttämällä useampaa luokittelijaa yhdistävää menetelmää kuten Boosting-algoritmia. Boosting-algoritmeissa on kuitenkin omat heikkoutensa, kuten opettelun keskittyminen häirötä sisältävään opetusaineistoon (Maglogiannis 2007).

6 Yhteenveto

Konenäkö on menetelmä, jolla automatisoidaan kuvasta tehtävää päättelyä ja sitä hyödynnetään esimerkiksi liikenteessä itsestään ajavissa autoissa sekä tehtaissa prosessien automatisoinnissa. Usein tunnistettavat asiat ovat niin monimutkaisia, että päättelyyn tarvitaan asiaa yleistäviä malleja ja koneoppimismenetelmät ovat hyvä tapa luoda näitä malleja. Tässä tutkielmassa suosituimpien koneoppimismenetelmien toimintaa selvitettiin ja niitä käyttäviä konenäkömenetelmiä esiteltiin. Lopuksi eri menetelmien heikkouksia ja vahvuuksia vertailtiin. Tällä hetkellä suosituimpia menetelmiä konenäkökäytössä ovat neuroverkot ja koneoppimismenetelmiä yhdistävät algoritmit kuten boosting. Kaikissa koneoppimismenetelmissä on omat ongelmansa, mutta ylisovitus on yhteinen ongelma lähes kaikissa niistä. Haasteita koneoppimisen käyttöön tuo myös tarve isolle määrälle opetusaineistoa ja kuvien muuntaminen algoritmeille tarpeelliseen vektorimuotoon. Jatkotutkimusta olisi mahdollista tehdä esimerkiksi koneoppimismenetelmien yhdistämisestä, tarkastella jonkin menetelmän sovelluksia tarkemmin tai vertailla menetelmiä empiirisesti.

Kirjallisuutta

Bertozzi, Broggi, Del Rose, Felisa, Rakotomamonjy ja Suard *Pedestrian Detector Using Histograms of Oriented Gradients and a Support Vector Machine Classifier* Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE

Boiman, Shechtman ja Irani *In Defense of Nearest-Neighbor Based Image Classification* Computer Vision ja Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, 23-28 June 2008

Buch, Velastin ja Orwell *A Review of Computer Vision Techniques for the Analysis of Urban Traffic* IEEE Transactions On Intelligent Transportation Systems, Vol. 12, No. 3, September 2011

Cenev, Venäläinen, Sariola ja Zhou *Object Tracking in Robotic Micromanipulation by Supervised Ensemble Learning Classifier*. Manipulation, Automation ja Robotics at Small Scales (MARSS)

Centeno, Lopes, Felisberto ja Ramos de Arruda *Object Detection for Computer Vision Using a Robust Genetic Algorithm* Applications of Evolutionary Computing: EvoWorkkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC Lausanne, Switzerland, March 30 - April 1, 2005 Proceedings, s. 284–293

Chtioui, Bertrand ja Barba *Feature Selection by a Genetic Algorithm. Application to Seed Discrimination by Artificial Vision* J Sci Food Agric 1998, 76, s. 77-86

Cracknell ja Reading *Geological mapping using remote sensing data: A comparison of five machine learning algorithms, their response to variations in the spatial distribution of training data and the use of explicit spatial information* Computers ja Geosciences 63, 2014, s. 22 – 33

Domingos *A Few Useful Things to Know About Machine Learning* Communications of the ACM, Volume 55 Issue 10, October 2012, s. 78-87

Feng ja Yuan *Static hand gesture recognition based on HOG characters and support vector machines* 2013 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA), Toronto, ON, 2013, s. 936-938.

Fernandes, Moreira ja Mata *Machine vision applications ja development aspects* Control

ja Automation (ICCA), 2011 9th IEEE International Conference on 19-21 Dec. 2011

Gonçalves ja Silva *Balancing Learning ja Overfitting in Genetic Programming with Interleaved Sampling of Training Data* EuroGP 2013: Genetic Programming s. 73-84

Haykin *Neural Networks: A Comprehensive Foundation*

Hsu ja Lin *A Comparison of Methods for Multiclass Support Vector Machines* IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 13, NO. 2, MARCH 2002

Hu, Yu, Li ja Ma *Extraction of parametric human model for posture recognition using genetic algorithm* roceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580), Grenoble, 2000, s. 518-523.

Jain, Duin ja Mao *Statistical Pattern Recognition: A Review* IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 22, NO. 1, JANUARY 2000

Kang, Seok-Il ja Kyeong-Hoon *Analysis of the HOG Parameter Effect on the Performance of Vision-Based Vehicle Detection by Support Vector Machine Classifier* Advances in Parallel and Distributed Computing and Ubiquitous Services: UCAWSN & PDCAT 2015, s. 221-228

Keller, Gray And Givens *A fuzzy K-nearest neighbor algorithm* IEEE Transactions on Systems, Man ja Cybernetics (Volume: SMC-15, Issue: 4, July-Aug. 1985)

Kittler, Hatef, Duin ja Mata *On Combining Classifiers* IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 20, No. 3, March 1998

Kohavi *Glossary of Terms* Machine Learning, February 1998, Volume 30, Issue 2, s. 271–274

Kotsiantis *Supervised Machine Learning: A Review of Classification Techniques* Proceedings of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval ja Pervasive Technologies, s. 3-24

Krizhevsky, Sutskever ja Hinton *Imagenet classification with deep convolutional neural networks* Advances in neural information processing systems. 2012.

Liu ja Cho *Machine vision in Process Systems Engineering* Control, Automation ja Systems (ICCAS), 2011 11th International Conference on 26-29 Oct. 2011

Lowe *Object Recognition from Local Scale-Invariant Features* Computer Vision, 1999.

The Proceedings of the Seventh IEEE International Conference on 20-27 Sept. 1999

Maglogiannis *Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies* IOS Press, 2007

Mohri, Rostamizadeh ja Talwalkar *Foundations of Machine Learning*

Murphree *Machine learning anomaly detection in large systems* 2016 IEEE AUTOTESTCON, 12-15 Sept. 2016

Norouzi, Collins, Johnson, Fleet ja Kohli *[*Efficient Non-greedy Optimization of Decision Trees] Advances in Neural Information Processing Systems 28, Curran Associates, Inc., s. 1729–1737

Orrùa, Pettersson-Yeoa, Marquand, Sartori, Mechelli *Using Support Vector Machine to identify imaging biomarkers of neurological ja psychiatric disease: A critical review* Neuroscience & Biobehavioral Reviews, Volume 36, Issue 4, April 2012, s. 1140–1152

Pontil ja Verri *Support vector machines for 3D object recognition* IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 20, No. 6, June 1998

Quinlan *Induction of decision trees* Machine Learning, March 1986, Volume 1, Issue 1, s. 81–106

Reeves *A genetic algorithm for flowshop sequencing* Computers & Operations Research, Volume 22, Issue 1, January 1995, s. 5-13

Schapire *The Strength of Weak Learnability* Machine Learning June 1990, Volume 5, Issue 2, s. 197–227

Sebastiani *Machine learning in automated text categorization* ACM Computing Surveys, Volume 34 Issue 1, March 2002, s. 1-47

Viola ja Jones *Rapid object detection using a boosted cascade of simple features* Computer Vision ja Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, Dec. 8-14 2001

Weinberger, Blitzer ja Saul *Distance metric learning for large margin nearest neighbor classification* Advances in neural information processing systems s. 1473-1480

Zhang *Neural Networks for Classification: A Survey* Ieee Transactions On Systems, Man ja Cybernetics—Part C: Applications And Reviews, Vol. 30, NO. 4, NOVEMBER 2000