Tampereen yliopisto

Jussi Kalliomäki

VAHVISTUSOPPIMISEN HYÖDYNTÄMINEN SUOSITTELUJÄRJESTELMISSÄ

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Toukokuu 2022

TIIVISTELMÄ

Jussi Kalliomäki: Vahvistusoppimisen hyödyntäminen suosittelujärjestelmissä
Kandidaattitutkielma
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma
Toukokuu 2022

Internetissä toimijoilla on todettu hyödylliseksi olla suosittelujärjestelmä nykyaikana. Tällainen järjestelmä auttaa tuomaan käyttäjälle häntä kiinnostavia asioita informaatiotulvan keskellä. Tässä tutkielmassa esitellään vahvistusoppimisen metodeita, suosittelujärjestelmien metodeita, sekä kuinka vahvistusoppimisen metodeilla voidaan parannella erityyppisiä suosittelujärjestelmiä.

Esiteltyihin vahvistusoppimismetodeihin kuuluvat dynaaminen ohjelmointi, Monte-Carlo metodit, sekä Q-oppiminen alaluokkineen. Suosittelujärjestelmien periaatteita esitellään keskittyen pääosin kollaboratiivisiin, sekä sisältöperusteisiin suosittelujärjestelmiin. Tutkielman tavoitteena on selvittää vahvistusoppimisen toimimista suosittelujärjestelmien toimintaperusteena.

Työn johtopäätöksenä huomataan, että monissa suosittelujärjestelmissä vahvistusoppimisen käytöllä on huomattavia etuja muunlaisiin nykyaikaisiin ja perinteisempiin suosittelujärjestelmien toimintametodien sijaan. Näitä eroja ovat esimerkiksi nopeampi mieltymysten muutoksiin reagointi sekä kylmäkäynnistysongelman ratkaisu. Työssä myös esitellään syvän vahvistusoppimisen sekä muiden syväoppimisen metodien käyttöä suosittelujärjestelmien toteutuksessa.

Avainsanat: Vahvistusoppiminen, syvä vahvistusoppiminen, suosittelujärjestelmät, koneoppiminen, tekoäly

SISÄLLYSLUETTELO

1. JOHDANTO

Internetin käytön kasvaessa erilaiset kaupalliset palvelut ovat alkaneet kehittää palveluitaan ottamalla käyttöön suosittelujärjestelmiä. Netflix-suoratoistopalvelussa noin 80% katseluista tulee suosittelujärjestelmien kautta, ja näiden personoitujen suositteluiden rahalliseksi arvoksi arvioidaan yli miljardia dollaria (Gomez-Uribe ja Hunt 2016). Myös esimerkiksi musiikin suoratoistopalvelu Spotify suosittelee artisteja ja kappaleita joista voisit pitää, ja jo suomalaisetkin verkkokaupat suosittelevat tuotteita joista voisit pitää katselemiesi tuotteiden perusteella.

Vahvistusoppiminen on muun koneopin yleistymisen myötä yleistyvä koneopin alaosio. Sillä on monia käyttötarkoituksia esimerkiksi autonomisten kulkuneuvojen ja työvälineiden koulutuksessa sekä pelien oppimisessa ja tutkimisessa. Näiden lisäksi myös suosittelujärjestelmiin on alettu hyödyntää perinteisten metodien lisäksi koneoppimista.

Suosittelujärjestelmiä on ollut olemassa 90-luvulta, kun Goldberg ja muut (1992) kehittivät ensimmäisen järjestelmän, mutta koneopin kehittymisen myötä myös suosittelujärjestelmiä voidaan toteuttaa ja parantaa koneopin avulla. Vahvistusoppiminen on suosittelujärjestelmien toteuttamiseen erittäin mielenkiintoinen ratkaisu, sillä vahvistusoppimista ja suosittelujärjestelmiä yhdistää tavoite parantaa tuotettuja tuloksia pitkällä aikatähtäimellä.

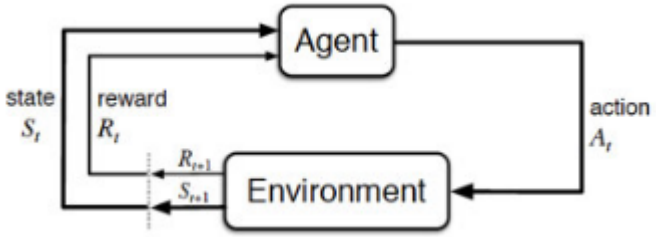
Tutkielman tarkoituksena on selvittää, miten vahvistusoppimisen eri metodit vertautuvat laadullisesti suosittelujärjestelmän toteuttamiseen. Tämän päätavoitteen saavuttamiseksi työssä on myös tarpeen esitellä teoriaa ja taustoja vahvistusoppimisen sekä suosittelujärjestelmien taustalla.

Luvuissa 2 ja 3 esitellään vahvistusoppimisen ja suosittelujärjestelmien teoriaa, sekä keskeisiä käsitteitä ja toimintaperiaatteita. Luvussa 4 esitellään metodeita suosittelujärjestelmien vertailuun, sekä kuinka vahvistusoppimisen käyttö on tehostanut suosittelujärjestelmiä. Viimeisenä lukuna 5 on yhteenveto, jossa käydään läpi saatuja tuloksia ja tulevaisuudennäkymiä aiheesta.

2. VAHVISTUSOPPIMINEN

Vahvistusoppiminen (engl. reinforcement learning, RL) on yksi kolmesta koneoppimisen osa-alueesta, joita ovat vahvistusoppimisen lisäksi ohjattu oppiminen ja ohjaamaton oppiminen. Koneoppiminen on tekoälyn osa-alue, joka koittaa selvittää kuinka rakentaa systeemi, joka osaa itse parantaa toimintaansa kokemuksensa kautta (Mitchell 1997, s. 1). Esimerkiksi koiran kouluttamista voidaan pitää vahvistusoppimisongelmana. Kun koiralle annetaan käsky istua, koira miettii hetken ja tekee jotain. Jos koiran toiminto oli oikein, eli koira istuu, saa koira palkkion. Jos koira toimii väärin se taas ei saa palkkiota. Näin koira oppii että tietyn käskyn kuullessaan kannattaa toimia tietyllä tavalla saadakseen palkkion.

Kaelbling ja muut (1996, s. 1) väittävät, että vahvistusoppiminen on ongelma jonka agentti kohtaa, kun se pakotetaan kehittämään toimintaansa yrityksen ja erehtymisen kautta dynaamisessa ympäristössä. Tämä on esitelty kuvassa 2.1. Näin vahvistusoppimisen ratkaisemat ongelmat ovat yleensä ongelmia, joissa selvitetään mikä olisi paras mahdollinen toiminto eri tilanteissa.



Kuva 2.1. *Vahvistusoppimisen ongelma (Naeem ja muut 2020)*

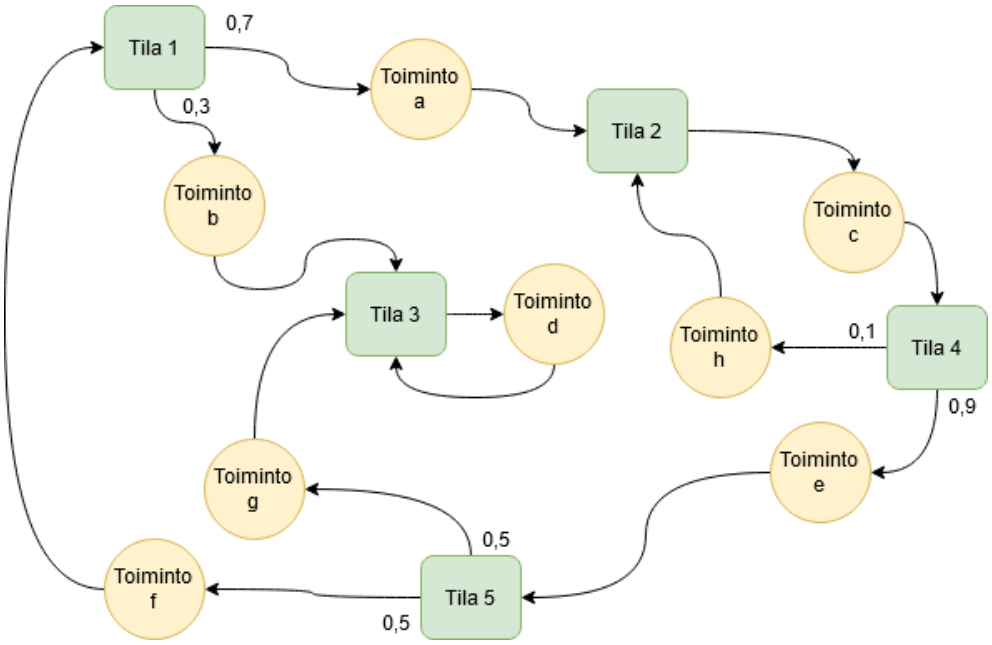
Vahvistusoppimisen ero muihin koneoppimisen osa-alueisiin on se, että vahvistusoppivalle agentille on vain määritelty millaiset toiminnot sille on mahdollisia, jonka jälkeen agentti joutuu itse selvittämään milloin mikäkin toiminto on hyvä yrityksen ja erehtymisen kautta. Ohjatussa oppimisessa taas agentille esitellään hyviä ja huonoja toimintoja tai määrittelyjä, ja agentin on tarkoitus oppia, mikä tekee näistä toiminnoista tai määrittelyistä hyviä ja huonoja. Ohjaamattomassa oppimisessa taas mallille annetaan dataa, josta malli yrittää tunnistaa eri ryhmiä tai tunnistettavia piirteitä.

2.1 Määritelmiä

Tässä alaluvussa määritellään ja esitellään vahvistusoppimisen kannalta oleellisia konsepteja ja terminologiaa.

2.1.1 Markovin päätösprosessi

Dynaaminen ympäristö jossa vahvistusoppimisagentti toimii kuvataan usein Markovin päätösprosessina (engl. Markov decision process, MDP). Otterlon ja Wieringin (2012, s. 10-12) mukaan tällaiselle prosessille ominaista on, että ympäristössä on tiloja, toimintoja, sekä siirtymiä tilojen välillä. Jokaisella tilalla voi myös olla palkkio, jonka agentti saa ollessaan tilassa. Kuvassa 2.2 on esimerkki yksinkertaisesta MDP:stä, jossa on joillekin toiminnoille myös todennäköisyydet.



Kuva 2.2. *Yksinkertaisen Markovin päätösprosessin esitys*

Esimerkiksi shakkirobotissa tiloina voisi toimia pelinappuloiden järjestys laudalla, jolloin tiloja on hyvin suuri määrä. Toimintoina luonnollisesti olisi jonkin pelinappulan siirto, jolloin tietenkin agentille pitää määrittää millaiset siirrot ovat laillisia. Palkkioiden määrittelyn taas voisi tehdä esimerkiksi määrittämällä palkkion voitetulle pelille, tai syödyille vastustajan nappuloille.

2.1.2 Arvopohjaisuus ja politiikkapohjaisuus

Vahvistusoppimisen algoritmeja voidaan jaotella joko arvo- tai politiikkapohjaisiksi, tai seuraavassa alaluvussa esitellysti mallipohjaisiin tai mallittomiin algoritmeihin. Arvopohjaisissa algoritmeissa on läsnä arvofunktio, joka määrittelee jonkinlaisen arvon tai palk-

kion MDP:n jokaiselle tilalle. Naeemin ja muiden (2020) mukaan arvopohjaisissa algoritmeissa agentti yrittää siis muodostaa tällaista arvofunktiota ympäristössään, ja edetä MDP:ssään vältellen epäedullisia tiloja, ja koittaen päästä edullisiin tiloihin. Arvopohjaisia algoritmeja on esimerkiksi luvussa 2.4 esiteltävä Q-oppiminen.

Politiikkapohjaisissa algoritmeissa yritetään Howardin (1960) mukaan luoda jonkinlainen politiikka, joka kuvaa kuinka agentin tulisi toimia jokaisessa tilassa. Hänen mukaansa optimaalisen politiikan on tarkoitus antaa agentille aina paras mahdollinen palkkio pitkällä tähtäimellä. Naeem ja muut (2020) väittävät, että politiikkoja voidaan jaotella joko deterministisiin politiikkoihin $\pi : S \to A$, tai stokastisiin politiikkoihin $\pi : S \times A \to [0, 1]$. He määrittivät että optimaalinen deterministinen politiikka luo eräänlaisen kartan, joissa joka tilasta seuraavaan annetaan aina optimaalinen toiminto. Heidän mukaansa optimaalinen stokastinen politiikka taas luo esityksen MDP:stä, jossa jokaisesta tilasta on kannattavuus tai todennäköisyys jokaiselle toiminnolle ja tilalle johon kyseinen toiminto vie. Esimerkiksi luvussa 2.4.1 esiteltävä SARSA on Q-oppimisesta johdettu politiikkapohjainen algoritmi.

2.1.3 mallipohjaisuus ja mallittomuus

Arvopohjaisuuden ja politiikkapohjaisuuden lisäksi toinen tapa jaotella algoritmeja on jakaa ne joko mallipohjaisiin tai mallittomiin algoritmeihin. Tässä kontekstissa mallin tuntemisella tarkoitetaan, että tiedetään millainen ympäristö on kyseessä. Wiering ja Van Otterlo (2012) väittävät, että mallipohjaisen algoritmin kehittäminen vaatii sitä, että ympäristönä on tavallinen MDP, eli että kaikki ympäristön tilat, toiminnot, siirtymät ja palkkiot tunnetaan, eli niitä on äärellinen määrä.

Mallittomissa algoritmeissa ei tarvita tietoa millainen ympäristö on kyseessä, vaan algoritmin toiminta perustuu täysin ympäristön tutkimiseen ja sen piirteiden muistamiseen. Sutton ja Barto (2018, s. 8) kutsuvat mallittomia algoritmeja "yritys ja erehdys -oppijoiksi".

Mallittomia algoritmeja ovat esimerkiksi luvussa 2.4 esiteltävät Q-oppiminen sekä SARSA, kun taas mallipohjaisia algoritmeja on esimerkiksi dynaamiseen ohjelmointiin kategorisoituvat algoritmit.

2.2 Dynaaminen ohjelmointi

Sutton ja Barto (2018, s. 89) määrittävät dynaamisella ohjelmoinnilla tarkoitettavan mallipohjaisia algoritmeja, joissa yritetään ratkaista optimaalisia politiikkoja vahvistusoppiongelman ratkaisemiseksi. Dynaaminen ohjelmointi on harvoin käytännössä toimiva ratkaisu, sillä käytännön ongelmissa harvoin tiedetään mallia täysin. Tämän vuoksi dynaamista ohjelmointia käytetään usein teoreettisena apuna, sillä sen avulla voidaan selittää olennaiset ongelmat ja mekanismit vahvistusopista.

Dynaamisen ohjelmoinnin perimmäinen idea on, että käydään mekaanisesti kaikki tilat ja toiminnot lävitse tunnetussa MDP:ssä, ja nämä kaikki tuntemalla voidaan iteroida optimaalinen arvofunktio tai politiikka. Dynaamiseen ohjelmointiin kuuluukin pääosin kaksi metodia, jotka esitellään seuraavissa alaluvuissa.

2.2.1 Arvoiterointi

Arvoiteroinnin tarkoituksena on MDP kokonaan lävitse käymällä selvittää optimaalinen arvofunktio, jota käyttämällä agentti voi tehdä halutun suorituksen aina optimaalisesti, eli joka tilassa ollessaan agentti tietää minkä toiminnon toteuttamalla sillä on suurin odotusarvo palkkiolle. Arvoiterointi perustuu Bellmanin (1957) kirjaan dynaamisesta ohjelmoinnista.

Algoritmi toimii käytännössä käymällä lävitse jokaisessa tilassa s jokaisen toiminnon a, ja laskemalla jokaisen näiden toimintojen aiheuttaman siirtymän jälkeisen tilan s' palkkioiden $R(s, a, s')$ odotusarvot Q-funktioon

$$Q(s, a) = \sum_{s'} T(s, a, s')(R(s, a, s') + \gamma V(s')) \tag{2.1}$$

käyttämällä apuna toiminnon todennäköisyysfunktiota $T(s, a, s')$. Yhtälössä myös käytetään parametrina diskonttokerrointa γ. Tämän jälkeen arvofunktioon lisätään arvoksi Q-funktion parhaan toiminnon arvo $max_a Q(s, a)$ tutkittavassa tilassa.

Arvoiteroinnin konsepti on selvästi huomattavissa luvussa 2.4 esiteltävässä Q-oppimisessa, ja tätä Bellmanin yhtälöksikin kutsuttua optimointimenetelmää sovelletaan paljon muissakin optimointiongelmissa, kuten säätö- ja talousteoriassa.

2.2.2 Politiikkaiterointi

Politiikkaiterointi on Howardin (1960) kehittämä metodi, jota voi soveltaa MDP:ssä josta tiedetään siirtymätodennäköisyysmatriisi P ja palkkiomatriisi R. Politiikkaiteroinnin koulutusprosessin tavoitteena on selvittää optimaalinen politiikka, tarkoittaen politiikkaa jossa keskimääräinen palkkio per siirtymä on mahdollisimman suuri.

Politiikkaiteraatio koostuu kahden osan toistamisesta, jotka ovat arvonpäättelyvaihe ja politiikanparannusvaihe. Arvonpäättelyvaiheessa selvitetään tilassa s saatavan palkkion odotusarvo

$$V(s) = R(s) + \gamma \sum_{s',r} P(s', r|s, a)V(s'), \tag{2.2}$$

jossa seuraavaan tilaan s' vievälle toiminnolle a, kun edellisessä tilassa s on saatu palkkio r, on todennäköisyys $P(s', r|s, a)$. Lisäksi seuraavalle tilalle on palkkio-odotusarvo $V(s')$, jota yhtälössä hyödynnetään. Yhtälössä esiintyvää diskonttausvakiota γ ei esitel-

ty alkuperäisessä paperissa, mutta nykyisissä artikkeleissa sitä käytetään usein. Tietoa tästä odotusarvosta käytetään seuraavaksi politiikanparannusvaiheessa.

Politiikanparannusvaiheessa toteutetaan yksinkertainen prosessi, jossa valitaan tilassa s mahdollisista toiminnoista a se, jonka seuraavan tilan odotusarvo $V(s')$ on suurin. Tätä voidaan kuvata politiikanparannusyhtälöllä

$$\pi(s) = argmax_a(V(s) + \sum_a P(a)V(s')). \tag{2.3}$$

Näitä kahta vaihetta toistetaan, kunnes politiikka konvergoituu, eli ei enää muutu iteraatioissa. Tällöin politiikka voidaan todeta optimaaliseksi.

2.3 Monte-Carlo metodit

Monte-Carlo metodit ovat tapoja ratkaista vahvistusoppimisongelmia, joissa ympäristöä ei tarvitse tuntea ollenkaan etukäteen. Sutton ja Barto (2018, s. 113) määrittelevät, että tällaiset metodit perustuvat ympäristön tutkimiseen episodeina, eli jaksoina jotka loppuvat joko saavutettaessa jokin tavoite, tai esimerkiksi tietyn määrän toimintoja jälkeen jolloin episodi määritetään epäonnistuneeksi. Yleensä Monte-Carlo metodit käyttävät hyväkseen satunnaista liikkumista ympäristössä, ja näin ainoastaan episodin päättyessä arvofunktiota ja/tai politiikkaa muutetaan.

Monte-Carlo metodeita on monenlaisia, ja niistä on kehitetty politiikkapohjaisia ja arvopohjaisia versioita, mutta niiden syvällisempi käsittely tässä työssä ei ole oleellista joten jätän niiden käsittelyn sikseen. Aiheen esilletuonti on tärkeää, jotta voi ymmärtää periaatteita uusien ympäristöä tuntemattomien metodien kehityksessä.

2.4 Q-oppiminen

Vahvistusoppimisen suurena läpimurtona (Sutton ja Barto 2018, s. 157) pidettävä Q-oppiminen on yksi tunnetuimmista mallittomista algoritmeista. Algoritmin kehittäjä Watkins (1989) kehitti algoritminsa tohtorinväitöskirjassaan. Tätä Watkinsin kehittämää Q-oppimista on hyödynnetty esimerkiksi nettisivujen kuormituksen tasaamisessa (Bu ja muut 2009), ja hieman käyttötarkoitukseen jalostettua muotoa siitä on käytetty myös suosittelujärjestelmän toteutukseen (Tang ja muut 2019).

Q-oppiminen on käytännötön algoritmi, jossa agentti kehittää jatkuvasti Q-funktiota tai Q-taulukkoa, joka kertoo saatavan palkkion jokaiselle toiminnolle jokaisessa tilassa. Aina toiminnon a_t tekemisen jälkeen tilassa s_t agentti päivittää Q-funktion arvon $Q_k(s_t, a_t)$ seuraavaksi arvoksi

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha(r_t + \gamma max_a Q_k(s_{t+1}, a) - Q_k(s_t, a_t)) \tag{2.4}$$

toiminnon jälkeisen tilan antaman palkkion r_t perusteella. Yhtälössä käytetään myös parametreina oppimisnopeutta α, joka määrittää kuinka nopeasti Q-funktion arvoja muutetaan, sekä diskonttausvakiota γ, joka määrittää kuinka suuri painoarvo seuraavan tilan s_{t+1} parhaalla mahdollisella palkkiolla on Q-funktion päivityksessä.

2.4.1 SARSA

Q-oppimisesta kehitettiin vuonna 1994 politiikkapohjainen metodi SARSA (Rummery ja Niranjan 1994), jonka nimi on lyhenne sanoista *state-action-reward-state-action*. Algoritmi nimettiin paperissa *muunnelluksi konnektionistiseksi Q-oppimiseksi*, mutta tekijät ottivat huomioon Suttonin antaman nimiehdotuksen SARSA, jonka lyhentämätön versio kuvaa algoritmia hyvin sen tarvitessa tietää tila, toiminto, palkkio, ja seuraavat tila ja toiminto. Paperin mukaan Q-oppimisessa on ongelma oppimisen alussa, jossa Q-funktion arvot voivat olla mitä sattuu, jolloin oppimisen alkupään toiminnot voivat olla hyvinkin epävarmoja, ja myöhemminkin Q-oppimisessa käytettävä max-funktio antaa yliarvioituja tuloksia.

Paperissa väitetään Q-oppiminen soveltumattomaksi tilanteisiin, joissa on ääretön tai tuntematon määrä mahdollisia tiloja. SARSAn paperissa laskettaisiin Q-funktion muutos käyttämällä ahneesta max-funktiosta riippuvan arvon sijaan Q-funktion päivitettynä arvona

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha(r_t + \gamma Q_k(s_{t+1}, a_{t+1}) - Q_k(s_t, a_t)), \tag{2.5}$$

missä seuraavan tilan maksimipalkkion sijaan funktio on riippuvainen todennäköisestä seuraavan tilan toiminnosta a_{t+1}. Tila a_{t+1} valitaan usein kirjallisuudessa toiminnoksi, jolla on seuraavan tilan toimintojen palkkioista suurin odotusarvo. Tätä ei olla kuitenkaan SARSAn paperissa määritelty olemaan näin, joten SARSAn sovelluksissa a_{t+1} voidaan valita myös muilla tavoin.

2.4.2 syvä Q-oppiminen

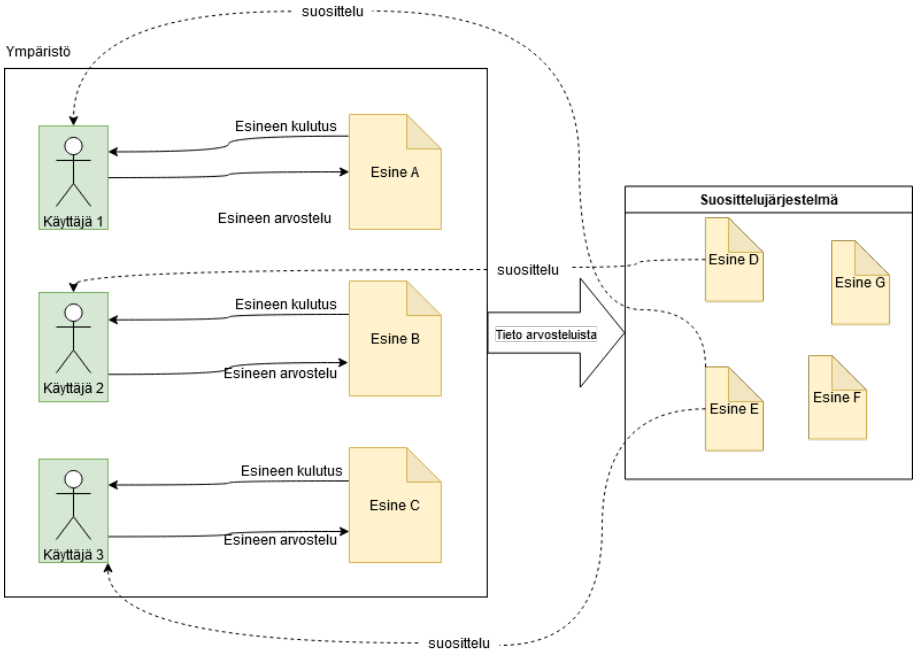
Q-oppimisen ideaa on sovellettu myös neuroverkkojen kanssa, ja luotu syvä Q-oppiminen (Mnih ja muut 2015). Syvässä Q-oppimisessa ei ole perinteisessä Q-oppimisessa käytettyä Q-funktiota tai -taulukkoa, vaan se on korvattu neuroverkolla. Tällä neuroverkolla on sisäänmenona esitys tilasta, ja ulostuloina ovat palkkioiden arvot jokaiselle mahdolliselle toiminnolle.

Mnih ja muut (2015) kehittivät syvän Q-oppimisen agentin, jota testattiin 49 videopelissä joista lähes kaikissa agentti voitti huomattavasti aikaisemman parhaan vahvistusoppimetodin. Syvää Q-oppimista on sovellettu esimerkiksi merkintunnistusalgoritmissa (Qiao ja muut 2018), sekä laivaliikenteen kolarinvälttämisjärjestelmässä (Shen ja muut 2019).

3. SUOSITTELUJÄRJESTELMÄT

Nykyään suosittelujärjestelmät (engl. recommender system) ovat oleellinen osa lähes kaikkia palveluita internetissä. Musiikin suoratoistopalvelut jatkavat automaattisesti toistoa kappaleilla joista järjestelmän mukaan saattaisit pitää, sekä verkkokaupat suosittelevat tuotteita aiemman selaushistorian perusteella. Suoratoistopalvelu Netflix on jopa järjestänyt kilpailun parhaan elokuvien suosittelujärjestelmän kehittämiseen miljoonan dollarin palkinnolla (Netflix 2006).

Suosittelujärjestelmä on siis järjestelmä, jonka tiedossa on käyttäjiä ja esineitä. Käyttäjistä ja heidän toiminnoistaan esineitä kohden järjestelmä jollakin tapaa päättelee, että tietty esine voisi olla tietylle käyttäjälle mieleinen, ja suosittelee tätä esinettä käyttäjälle. Tätä toimintaperiaatetta esitellään kuvassa 3.1.



Kuva 3.1. *Suosittelujärjestelmän periaate kuvamuodossa*

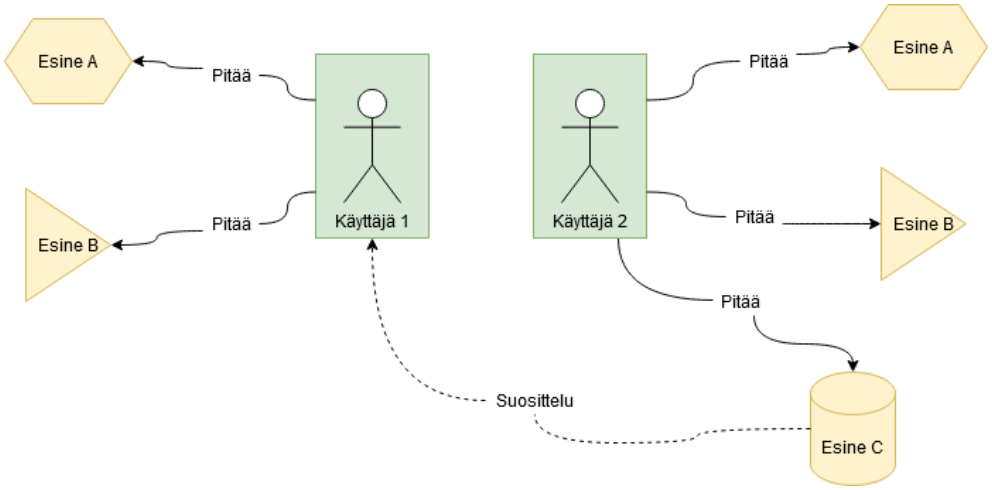
Suosittelujärjestelmien juuria on nähtävissä psykologian kognitiivisissa tieteissä, esimerkiksi Richin (1979) stereotypioita käsittelevässä artikkelissa, mutta omaksi tutkimusalakseen suosittelujärjestelmät kehittyivät 90-luvulla. Ensimmäinen maininta suosittelujärjestelmistä on, Karlgrenin (1990) tutkimusraportissa, jossa hän kehitti matematiikkaa juuri-

kin suosittelujärjestelmien kehittämiseen. Ensimmäisen kirjallisuudesta löytyvän suosittelujärjestelmän kehitti Goldberg ja muut (1992), jonka käyttötarkoituksena oli sähköpostin suodatus. Tämän jälkeen tutkimuksia alkoi tulemaan enemmänkin ja suosittelujärjestelmät kehittyivät omaksi tutkimusalakseen.

Suosittelujärjestelmiä on perinteisesti kahta eri tyyppiä, kollaboratiivisia ja sisältöperusteisia. Nykyaikana suosittelujärjestelmien kehittyessä on myös alettu käyttää muita monimutkaisempia tyylejä, sekä monen tyypin yhdistelmää, eli hybridisuosittelujärjestelmää. Näissä perinteisissä tyyleissä pääerona on, että etsitäänkö yhteneväisyyksiä käyttäjistä vai suositeltavista esineistä.

3.1 Kollaboratiiviset suosittelujärjestelmät

Kollaboratiivisten suosittelujärjestelmän perimmäinen idea on, että menneisyydessä samaa mieltä olleet käyttäjät ovat jatkossakin samaa mieltä asioista. Esimerkiksi elokuvien suosittelujärjestelmässä jos kaksi henkilöä ovat antaneet saman arvosanan monelle elokuvalle, antavat he todennäköisesti muillekin elokuville lähes saman arvostelun. Tätä toimintaperiaatetta on myös esitetty kuvassa 3.2, jossa henkilöt 1 ja 2 ovat kertoneet pitävänsä esineistä A ja B. Näin ollen voidaan olettaa että henkilön 2 pitäessä esineestä C, myös henkilö 1 pitää siitä, ja esinettä suositellaan hänelle.



Kuva 3.2. *Kuvaesimerkki kollaboratiivisen suosittelujärjestelmän toiminnasta*

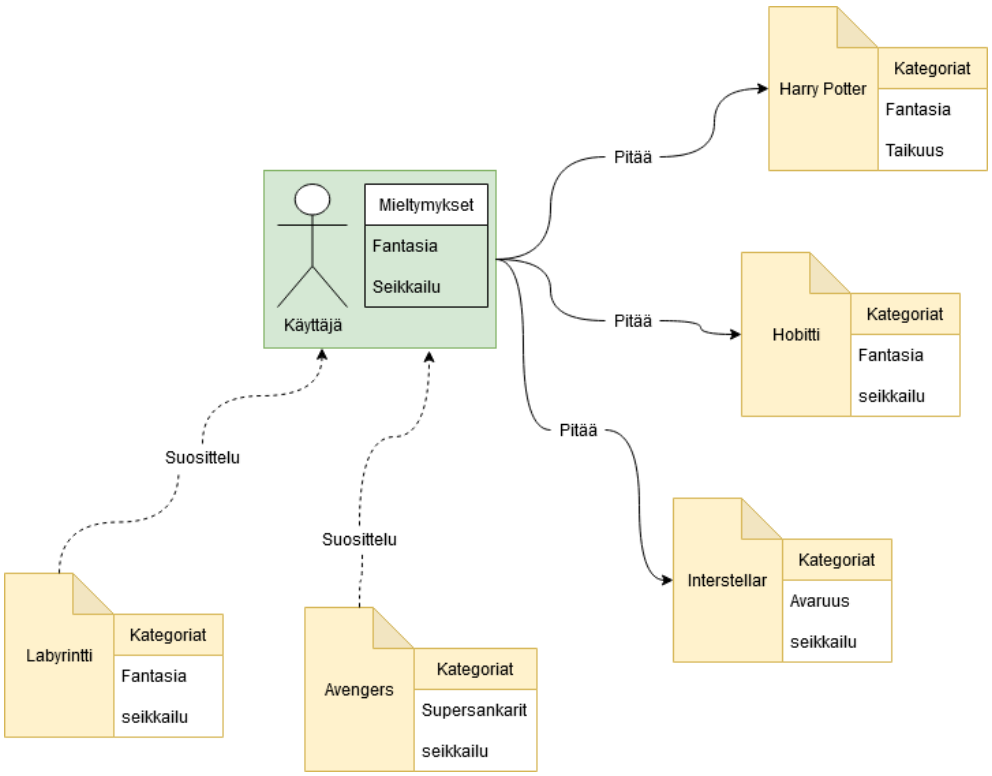
Schafer ja muut (2007, s. 293) jaottelevat käyttäjien mielipiteiden keräämisen esineistä suoraan ja epäsuoraan tapaan. Suora tapa tarkoittaa sitä, että henkilöltä kysytään suoraan mielipidettä esineestä. Epäsuora tapa taas on sellainen, jossa käyttäjän toimimisesta päätellään käyttäjän mielipide esineestä. Suorassa palautteenkeruussa esimerkiksi elokuvan katsomisen jälkeen voidaan pyytää numeroarvosana elokuvasta. Epäsuorassa palautteenkeruussa taas esimerkiksi elokuvan katsojan lopettaessa elokuvan A katsomisen 10 minuuttia elokuvan alusta, ja alkaa katsoa elokuvaa B, voidaan olettaa ettei katsoja

pitänyt elokuvasta A.

Kollaboratiiviset suosittelujärjestelmät kärsivät kolmesta ongelmasta, joita ovat kylmäkäynnistysongelmat, skaalausongelmat, sekä palautteiden harvinaisuus. Kylmäkäynnistysongelmat tarkoittavat ongelmaa, joka syntyy kun uusi henkilö tai esine lisätään järjestelmään (Schafer ja muut 2007, s. 311). Tälle uudelle asialle on annettava jonkinlaisia alkumääritelmiä, jotta suosittelujärjestelmä ottaa sen huomioon tuleviin suositteluihin. Toiset kaksi ongelmaa liittyvät siihen, että suosittelujärjestelmiä käytetään usein sivuilla joilla on joko paljon käyttäjiä tai esineitä. Jos sivulla on paljon käyttäjiä järjestelmän skaalaus kaikille käyttäjille on raskasta ja resurssi-intensiivistä. Jos taas sivulla on liikaa esineitä verrattuna käyttäjän arvostelemiin esineisiin, henkilö ei tiedä esineiden ja niiden kategorioiden olemassaolosta, eikä näin ollen tule arvostelemaan niitä (Lee ja muut 2004).

3.2 Sisältöperusteiset suosittelujärjestelmät

Sisältöperusteisissa suosittelujärjestelmissä perimmäisenä ideana on kerätä käyttäjälle avainsanoja tai ominaisuuksia, joiden perusteella voidaan suositella käyttäjälle häntä kiinnostavia asioita (Pazzani ja Billsus 2007). Kuvassa 3.3 on esimerkki elokuvien suosittelujärjestelmästä, jossa käyttäjälle on kerätty tykättyjen elokuvien perusteella tykätyiksi kategorioiksi fantasia ja seikkailu. Näistä johtuen käyttäjälle suositellaan elokuvia, joiden kategorioissa esiintyy näitä pidettyjä kategorioita.



Kuva 3.3. *Kuvaesimerkki sisältöperusteisen suosittelujärjestelmän toiminnasta*

Sisältöperusteisissa järjestelmissä käyttäjän attribuuttien määrittelemiseen voidaan käyttää luvussa 3.1 esiteltyjä suoria ja epäsuoria tapoja. Suorana tapana käyttäjälle voidaan esimerkiksi antaa lista erilaisia attribuutteja, joista hän voi valita itseään kiinnostavia asioita. Epäsuorana tapana taas esimerkiksi käyttäjän selatessa verkkokaupasta veitsiä ruoanlaittoon voidaan olettaa käyttäjän olevan kiinnostunut ruoanlaitosta.

Sisältöperusteiset suosittelujärjestelmät kärsivät kollaboratiivisten järjestelmien tavoin kylmäkäynnistysongelmista uusien käyttäjien kohdalla (Adomavicius ja Tuzhilin 2005, s. 737). Järjestelmä ei tiedä uudesta käyttäjästä mitään, ellei uudelta käyttäjältä pyydetä suoraan kiinnostavia attribuutteja, tai odoteta uuden käyttäjän manuaalisesti selaavan häntä kiinnostavia esineitä jotta attribuutteja saadaan kerättyä epäsuorasti.

Toinen sisältöperusteisten suosittelujärjestelmien kohtaama ongelma on se, että uuden käyttäjän on vaikeaa törmätä hänelle uusiin kategorioihin jotka voisivat kiinnostaa häntä (Adomavicius ja Tuzhilin 2005, s. 737). Esimerkiksi grilliruoasta pitävälle ei todennäköisesti suositella kasvisruokaravintolaa. Tätä ongelmaa vastaan voidaan käyttää esimerkiksi satunnaistamista, tai muuten vain kehittää suosittelut niin että ne ovat monipuolisia eivätkä sisällä suosituksia vain yhden attribuuttikategorian perusteella.

3.3 Hybridisuosittelujärjestelmät

Hybridisuosittelujärjestelmät ovat sellaisia järjestelmiä, jotka yhdistävät joitakin tunnettuja suosittelujärjestelmätyyppejä. Usein kirjallisuudessa hybrideiksi kutsutaan juuri kollaboratiivisen ja sisältöperusteisen suosittelujärjestelmän yhdistelmiä, mutta termiä ei olla määritelty tarkoittamaan järjestelmiä, jotka yhdistävät juuri näitä kahta.

Koska usean eri tyypin suosittelujärjestelmät eivät usein ole ristiriidassa keskenään, vaan täydentävät toisiaan, on hyödyllistä yhdistää eri tyyppisiä suosittelujärjestelmiä. Hybridin vertautumista yksinkertaiseen suosittelujärjestelmään on tutkittu, ja ne ovat usein tarkempia keskipoikkeaman (engl. mean absolute error, MAE) perusteella suorituskykyä uhraamatta. Tällaisiin tuloksiin on tullut esimerkiksi Basu ja muut (1998) sekä Gupta ja Gadge (2015).

Suosittelujärjestelmätyyppien yhdistäminen hybridiksi voidaan tehdä esimerkiksi laittamalla useampi suosittelujärjestelmä rinnan ja yhdistämällä niiden suosittelutulokset, tai lisäämällä karakteristisia ominaisuuksia toisesta tyypistä toiseen esimerkiksi ottamalla kollaboratiiviseen suosittelujärjestelmään mukaan esineitä kuvaavia attribuutteja ja käyttäjän suosimia attribuutteja.

4. VAHVISTUSOPPIMINEN SUOSITTELUJÄRJESTELMIEN PARANTAMISESSA

Adomavicius ja Tuzhilin (2005) väittävät, että tämän sukupolven suosittelujärjestelmissä on paljon parannettavaa. Käyttäjien profilointi on usein liian yksinkertaista, arvosteluiden käsittelyssä voisi käyttää laajemmin matematiikan approksimaatioteoriaa, sekä $käyttäjä \times esine$ ulottuvuuden sijaan voitaisiin ottaa huomioon kontekstuaalisia piirteitä kuten kellon- tai vuodenaikaa, säätä tai henkilön perhetilannetta.

Perinteistä vahvistusoppia käyttävissä suosittelujärjestelmissä kuten luvussa 2.4 esiteltyä Q-oppimista käyttävä Tangin ja muiden (2019) kehittämää järjestelmää, voidaan käyttää yksinkertaisempiin suosittelujärjestelmiin menestyksekkäästi. Kuitenkin kun halutaan ottaa huomioon monipuolisempaa dataa, on käytännöllisempää ottaa käyttöön nykyaikaisia koneoppimisratkaisuja. Koneopille on helppo antaa dataa syötteenä, ja antaa agentin päätellä mikä data on oleellista ja tehdä päätelmät tehtävistä suosituksista.

4.1 Arvostelukriteerit

Suosittelujärjestelmien arvioimisissa käytetään monenlaisia kriteerejä. Suosituin niistä on jo luvussa 3.3 mainittu keskipoikkeama

$$MAE = \frac{\sum_{i=1}^{n} |e_i|}{n}, \tag{4.1}$$

jossa lasketaan yksinkertaisesti keskiarvo virheistä e_i, jotka ovat erotuksia käyttäjän arvosanan ja järjestelmän arvioiman arvosanan välillä. Toinen samankaltainen mittari on keskineliövirheen neliöjuuri (engl. root mean squared error, RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}}. \tag{4.2}$$

Virheenlaskutavat MAE ja RMSE ovat hyödyllisiä arvosteluperiaatteita yksittäisten esineiden suosittelujen osuvuuteen. Jos kuitenkin halutaan mitoittaa koko suosittelujärjestelmän osuvuutta hyvin, voidaan tarvita erilaisia mittareita.

Koko suosittelujärjestelmän arvioimiseen voidaan käyttää esimerkiksi yksinkertaista laskuria, jossa käyttäjä kertoo suositellusta esineestä oliko se relevantti vai ei. Tämän jälkeen

voidaan laskea suhde relevanteille suositteluille kaikista suositteluista. Mainonnassa tällaiselle arvioinnille on kehitetty oma epäsuoraa palautetta käyttävä mittari, *klikkiprosentti (engl. click-through rate, CTR)* (Google 2018). CTR mittaa siis suhteen, kuinka iso osa mainoksen nähneistä ihmisistä klikkaa mainosta.

Suosittelujärjestelmien vertailuissa tieteellisissä artikkeleissa on kuitenkin hieman hankaluuksia, sillä eri lähteissä käytetään eri vertailutapoja. Zhang ja muut (2019) sekä Afsar ja muut (2021) kritisoivat sitä että tieteellisten artikkeleiden kirjoittajat saavat käytännössä itse päättää käytetyn data-aineiston, sekä vertailukohdat. Tämä virallisten tai vakinaistuneiden arvosteluperusteiden puute mahdollistaa sen, että uusissa tutkimuksissa voidaan aina etsiä data-aineistot ja mittausmetodit, joilla tutkimuksessa kehitetty metodi näyttää paperilla paremmalta.

4.2 Perinteinen vahvistusoppiminen suosittelujärjestelmissä

Perinteistä vahvistusoppimista käyttävien suosittelujärjestelmien on huomattu tuottavan monissa tutkimuksissa perinteisiä kollaboratiivisia tai sisältöperusteisia suosittelujärjestelmiä parempia tuloksia (Shi ja muut 2021; Choi ja muut 2018; Tang ja muut 2019). Näiden kehittäminen toi tutkimuskentälle tietoisuutta ja ymmärrystä vahvistusoppimisen vahvuuksista sekä heikkouksista.

Afsar ja muut (2021) huomasivat löytämiensä tutkimusten perusteella, että perinteistä vahvistusoppimista hyödyntävissä suosittelujärjestelmissä yleisin toteutusalgoritmi on luvussa 2.4 esitelty Q-oppiminen. Tämä johtuu todennäköisesti Q-oppimisen yksinkertaisuudesta ja sitä kautta sen soveltuvuudesta moneen erilaiseen asiaan hyvin.

Samassa Afsarin ja muiden (2021) kirjallisuuskatsauksessa huomattiin löydettyjen tutkimusten perusteella suosittelujärjestelmien tutkimuskentän muuttuneen huomattavasti syvän vahvistusoppimisen kehityksen myötä. Heidän mukaansa vahvistusoppimista käyttävien suosittelujärjestelmien kenttää on syytä jakaa aikaan ennen ja jälkeen syvän vahvistusopin kehityksen.

4.3 Syvä vahvistusoppiminen suosittelujärjestelmissä

Munemasa ja muut (2018) huomasivat syvällä vahvistusoppimisella olevan etuja ainakin moniulotteisen tila-avaruuden hyväksikäyttämisessä. Perinteinen vahvistusoppiminen ei ole kovin käytännöllistä tällaisissa monimutkaisissa suosittelujärjestelmissä, sillä kehittyneissä järjestelmissä halutaan usein juurikin ottaa moniulotteisemmin tiloja huomioon, ja tämä aiheuttaisi hyvin monimutkaisen prosessin suosittelujärjestelmän kehitykseen perinteisin metodein. Syvällä vahvistusoppimisella on huomattu olevan muitakin hyviä ominaisuuksia suosittelujärjestelmien käyttöön, kuten että sillä usein löydetään lopulta optimaalinen ratkaisu (Shi ja muut 2021; Shin ja Bulut 2021).

Syvän vahvistusoppimisen mukanaan tuomien neuroverkkojen on huomattu ottavan suositteluissa huomioon kontekstuaalisia piirteitä joita perinteisemmät järjestelmät eivät ole onnistuneet tekemään yhtä kattavasti. Tämä johtuu neuroverkkojen tavasta löytää datasta yhteneväisyyksiä, jollaisia ihmistutkija ei välttämättä datasta edes huomaisi.

Zhao ja muut (2021) kehittivät tutkimuksessaan syvää vahvistusoppimista hyödyntävän suosittelujärjestelmän, joka käyttää tageja sisältävää dataa. Kun tätä suosittelujärjestelmää sovellettiin MovieLens elokuvadataan, saatiin MAE arvoksi 0,3720, joka oli noin 40 % pienempi kuin perinteisemmän klusterointia hyödyntävän kollaboratiivisen metodin MAE 0,6142. He myös spekuloivat tutkimustulostensa perusteella, että syvällä vahvistusoppimisella on taipumusta ratkaista kylmäkäynnistysongelmaa.

4.4 Muu syväoppiminen suosittelujärjestelmissä

Syvää vahvistusoppimista pidetään hyvänä metodina sellaisten suosittelujärjestelmien kehitykseen, joissa halutaan mahdollistaa käyttäjän mielenkiinnon nopeat muutokset (Zhang ja muut 2019). Tällaisia käyttötarkoituksia voisi olla esimerkiksi uutissivustot, joissa käyttäjän kiinnostukset voivat vaihdella nopeasti esimerkiksi yllättävien tapahtumien vuoksi.

Muita syväoppimista hyödyntämiä algoritmeja on esimerkiksi monikerrosperseptroneja (engl. multi-layer perceptron) käyttävät menetelmät, joilla voidaan matkia luvussa 3 mainittuja perinteisiä suosittelujärjestelmätyyppejä, sekä konvoluutioneuroverkkoja hyödyntävät järjestelmät, jotka ovat hyviä esimerkiksi graafimuotoisen datan ymmärtämisessä ja siitä suosittelujen luomisessa (Zhang ja muut 2019). Zhang ja muut (2019) esittelevät tutkimuksessaan hyvin kattavasti myös muita syväoppimisen algoritmeja, ja tutkimuksia joissa niitä on käytetty suosittelujärjestelmiin. Kuitenkaan heidän tutkimuksessaan, tai muidenkaan löydettyjen tutkimusten tuloksissa, ei ole vielä osattu eritellä onko jokin algoritmi tai metodi toisia parempi.

4.5 Nykyaikaisten suosittelujärjestelmien ongelmat

Vaikka nykyaikaisissa vahvistusoppivissa suosittelujärjestelmissä on piirteitä, joita perinteiset metodit eivät voi matkia, on niissä silti ongelmia. Suurimpia näistä suosittelujärjestelmien ongelmista ovat tarvittavan laskentatehon suuruus, koulutukseen tarvittavan datan keruu ja siistiminen, parametrien hienosäätö sekä neuroverkkojen mustan laatikon luonteesta aiheutuva tulosten analysoinnin vaikeus (Zhang ja muut 2019). Viimeinen näistä mainituista on hyvin tärkeä ongelma ratkaista, sillä jos suosittelujärjestelmän suositteluperusteita ei ymmärretä, on järjestelmää hyvin vaikeaa kehittää. Lisäksi käyttäjälle voi olla tärkeätä nähdä yhteyksiä hänelle mieluisten esineiden välillä.

Vahvistusoppivien suosittelujärjestelmien vaatiman koulutusdatan keruu ja siistiminen on myös suuri haaste järjestelmien kehittäjille. Dataa tarvitaan niin paljon, että manuaalisena

työnä sen keruuta ja siistimistä voidaan pitää mahdottomana. Tähän ongelmaan ratkaisuna tarvitaan siis jonkinlaista järjestelmää tai yleistä täydellistä tietokantaa. Elokuvien suosittelujärjestelmien kehittämiseen on olemassa hyvin suuri yleisesti käytetty data-aineisto MovieLens, mutta muihin tarkoituksiin tällaista ei ole.

Nykyaikana suoratoistopalveluissa tai verkkokaupoissa suositeltavien esineiden määrät voivat kasvaa jopa miljooniin, joten niiden datan käsittelyyn tarvitaan valtavasti laskentatehoa. Vaikka nykyaikana erityisesti grafiikkaprosessorien kasvavaa laskentatehoa voidaan käyttää tekoälysovelluksissa, näkyy suurempi laskentatehon tarve myös tarvittavissa sähköenergiamäärissä. Nykyään kaikissa suurissa tekoälyjärjestelmissä on siis otettava huomioon myös ympäristötekijät.

5. YHTEENVETO

Vahvistusoppimisella on viime aikoina huomattu olevan huomattavan hyödyllisiä piirteitä suosittelujärjestelmien kehitykseen ja tästä johtuen vahvistusoppia hyödyntävien suosittelujärjestelmien kehitys on kasvava tutkimusala. Vahvistusoppi on periaatteeltaan suosittelujärjestelmiin hyvin sopiva metodi, sillä kummassakin pääosassa on tuntemattomasta tilanteesta liikkeelle lähteminen ja järjestelmän jatkuva parantaminen saadun palautteen perusteella.

Kuten luvussa 4.1 todettiin, nykyaikaisten vahvistusoppimista hyödyntävien suosittelujärjestelmien vertailu keskenään on haastavaa yhteisten määriteltyjen arviointiperusteiden puutteen takia. Luvussa esitetyistä tuloksista on kuitenkin huomattavissa vahvistusoppia hyödyntävien järjestelmien etu perinteisiin järjestelmiin verrattaessa, sekä hyviä piirteitä verrattaessa muihin koneoppimista hyödyntäviin suosittelujärjestelmiin. Vielä parempia tuloksia on saatu ottaessa käyttöön perinteisen vahvistusoppimisen sijaan neuroverkot, eli siirtymällä syvään vahvistusoppimiseen.

Työssä kuitenkin huomataan syväoppia käyttävien suosittelujärjestelmien tutkimuskentän olevan vielä niin uusi, että laadukasta eri syväoppimisen metodeita vertailevaa tutkimusta ei ole vielä toteutettu. Syväoppimisessa on kuitenkin huomattu olevan hyviä puolia suosittelujärjestelmien kehittämiseen esimerkiksi kontekstuaalisten suositteluiden paremmassa tekemisessä. Syväoppimisella saadaan siis otettua kontekstuaalisia piirteitä paremmin huomioon, sillä tila-avaruuden ja markovin päätösprosessin kentän ei tarvitse olla yhtä yksinkertainen kuin perinteisemmissä metodeissa.

Työn tavoitteena oli tutustua vahvistusoppimisen käyttämiseen suosittelujärjestelmien toteutuksessa, ja tutkia kuinka eri vahvistusoppimisen metodit ovat vaikuttaneet tutkimuskenttään. Lisäksi tarkoituksena oli tutustua suosittelujärjestelmien sekä vahvistusoppimisen toimintaperiaatteisiin. Näihin tavoitteisiin päästiin kirjallisuuskatsausmenetelmällä, jossa tuotiin esille eri tavoin oleellisia tutkimuksia, ja esiteltiin näiden tuloksia.

Jatkotutkimusaiheita tutkielman aiheisiin liittyen on useampikin. Eräs hyvin hyödyllinen jatkotutkimusaihe voisi myös olla kehittää metodi jolla voidaan luoda paremmin ymmärrettäviä suositteluita. Kuten luvussa 4.5 mainittiin, syväoppimisalgoritmeilla on eräänlainen "musta laatikko-luonne, jonka ratkominen ja selventäminen olisi todennäköisesti suuri edistysaskel suosittelujärjestelmien tutkimuskentällä. Myös suosittelujärjestelmien vertai-

lutapojen analyysissä, ja hyvien arvosteluperiaatteiden kehittämisessä olisi tutkittavaa ja kehitettävää.

LÄHTEET

Adomavicius, G ja A Tuzhilin (2005). "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions". eng. *IEEE transactions on knowledge and data engineering* 17.6, s. 734–749. ISSN: 1041-4347.

Afsar, M. Mehdi, Trafford Crump ja Behrouz Homayoun Far (2021). "Reinforcement learning based recommender systems: A survey". *ArXiv* abs/2101.06286.

Basu, Chumki, Haym Hirsh ja William Cohen (1998). "Recommendation as classification: Using social and content-based information in recommendation". eng. Teoksessa: *Proceedings of the National Conference on Artificial Intelligence*, s. 714–720.

Bellman, Richard (1957). "Dynamic Programming". Princeton, Yhdysvallat: Princeton University Press.

Bu, Xiangping, Jia Rao ja Cheng-Zhong Xu (2009). "A Reinforcement Learning Approach to Online Web Systems Auto-configuration". eng. Teoksessa: *2009 29th IEEE International Conference on Distributed Computing Systems*. IEEE, s. 2–11. ISBN: 9780769536606.

Choi, Sungwoon, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha ja Sungroh Yoon (2018). "Reinforcement Learning based Recommender System using Biclustering Technique". *CoRR* abs/1801.05532. arXiv: 1801.05532. URL: http://arxiv.org/abs/1801.05532.

Goldberg, David, David Nichols, Brian Oki ja Douglas Terry (1992). "Using collaborative filtering to weave an information tapestry". eng. *Communications of the ACM* 35.12, s. 61–70. ISSN: 0001-0782.

Gomez-Uribe, Carlos ja Neil Hunt (2016). "The Netflix Recommender System: Algorithms, Business Value, and Innovation". eng. *ACM transactions on management information systems* 6.4, s. 1–19. ISSN: 2158-656X.

Google (2018). "Arkistoitu: Google ads CTR definition". (viitattu: 23.3.2022). URL: http://web.archive.org/web/20180817221401/https://support.google.com/google-ads/answer/2615875?hl=en.

Gupta, Jyoti ja Jayant Gadge (2015). "Performance analysis of recommendation system based on collaborative filtering and demographics". eng. Teoksessa: *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*. IEEE, s. 1–6. ISBN: 9781479955220.

Howard, Ronald A. (1960). "Dynamic Programming and Markov Processes". Oxford, Englanti: The M.I.T Press, s. 32–39.

Kaelbling, Leslie Pack, Michael L Littman ja Andrew W Moore (1996). "Reinforcement learning: A survey". eng. *The Journal of artificial intelligence research* 4, s. 237–285. ISSN: 1076-9757.

Karlgren, Jussi (1990). "An algebra for recommendations". *Syslab Working Paper* 179.

Lee, Sanghack, Jihoon Yang ja Sung-Yong Park (2004). "Discovery of hidden similarity on collaborative filtering to overcome sparsity problem". eng. Teoksessa: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 3245. Berlin: Springer, s. 396–402. ISBN: 9783540233572.

Mitchell, Tom M. (1997). "Machine learning". eng. McGraw-Hill series in computer science. New York: McGraw-Hill, s. 1. ISBN: 0-07-042807-7.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg ja Demis Hassabis (2015). "Human-level control through deep reinforcement learning". eng. *Nature (London)* 518.7540, s. 529–533. ISSN: 0028-0836.

Munemasa, Isshu, Yuta Tomomatsu, Kunioki Hayashi ja Tomohiro Takagi (2018). "Deep reinforcement learning for recommender systems". eng. Teoksessa: *2018 International Conference on Information and Communications Technology (ICOIACT)*. Vol. 2018-. IEEE, s. 226–233. ISBN: 1538609541.

Naeem, Muddasar, Syed Tahir Hussain Rizvi ja Antonio Coronato (2020). "A Gentle Introduction to Reinforcement Learning and its Application in Different Fields". eng. *IEEE access* 8, s. 209320–209344. ISSN: 2169-3536.

Netflix (2006). "Arkistoitu: The Netflix Prize Rules". (viitattu: 14.3.2022). URL: https://web.archive.org/web/20061029084552/https://www.netflixprize.com/assets/rules.pdf.

Otterlo, Martijn van ja Marco Wiering (2012). "Reinforcement Learning and Markov Decision Processes". eng. Teoksessa: *Reinforcement Learning*. Vol. 12. Adaptation, Learning, and Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 3–42. ISBN: 364227644X.

Pazzani, Michael J ja Daniel Billsus (2007). "Content-Based Recommendation Systems". eng. Teoksessa: *The Adaptive Web*. Vol. 4321. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 325–341. ISBN: 9783540720782.

Qiao, Junfei, Gongming Wang, Wenjing Li ja Min Chen (2018). "An adaptive deep Q-learning strategy for handwritten digit recognition". eng. *Neural networks* 107, s. 61–71. ISSN: 0893-6080.

Rich, Elaine (1979). "User modeling via stereotypes". eng. *Cognitive science* 3.4, s. 329–354. ISSN: 0364-0213.

Rummery, G. ja Mahesan Niranjan (1994). "On-Line Q-Learning Using Connectionist Systems". *Technical Report CUED/F-INFENG/TR 166*.

Schafer, J. Ben, Dan Frankowski, Jon Herlocker ja Shilad Sen (2007). "Collaborative Filtering Recommender Systems". eng. Teoksessa: *The Adaptive Web*. Vol. 4321. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 291–324. ISBN: 9783540720782.

Shen, Haiqing, Hirotada Hashimoto, Akihiko Matsuda, Yuuki Taniguchi, Daisuke Terada ja Chen Guo (2019). "Automatic collision avoidance of multiple ships based on deep Q-learning". eng. *Applied ocean research* 86, s. 268–288. ISSN: 0141-1187.

Shi, Bichen, Elias Z Tragos, Makbule Gulcin Ozsoy, Ruihai Dong, Neil Hurley, Barry Smyth ja Aonghus Lawlor (2021). "DARES: An Asynchronous Distributed Recommender System Using Deep Reinforcement Learning". eng. *IEEE access* 9, s. 83340–83354. ISSN: 2169-3536.

Shin, Jinnie ja Okan Bulut (2021). "Building an intelligent recommendation system for personalized test scheduling in computerized assessments: A reinforcement learning approach". eng. *Behavior research methods*. ISSN: 1554-351X.

Sutton, Richard S. ja Andrew G. Barto (2018). "Reinforcement learning : an introduction". eng. Second edition. Adaptive computation and machine learning. Cambridge, MA: The MIT Press, s. 8, 89, 113, 157. ISBN: 9780262039246.

Tang, Xueying, Yunxiao Chen, Xiaoou Li, Jingchen Liu ja Zhiliang Ying (2019). "A reinforcement learning approach to personalized learning recommendation systems". eng. *British journal of mathematical & statistical psychology* 72.1, s. 108–135. ISSN: 0007-1102.

Watkins, Christopher (1989). "Learning From Delayed Rewards". Tohtorinväitöskirja. Cambridge, Englanti: Cambridge University.

Wiering, Marco ja Martijn Van Otterlo (2012). "Reinforcement Learning State-of-the-Art". eng. 1st ed. 2012. Adaptation, Learning, and Optimization, 12. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 17. ISBN: 1-280-79537-9.

Zhang, Shuai, Lina Yao, Aixin Sun ja Yi Tay (2019). "Deep Learning Based Recommender System: A Survey and New Perspectives". eng. *ACM computing surveys* 52.1, s. 1–38. ISSN: 0360-0300.

Zhao, Zhiruo, Xiliang Chen, Zhixiong Xu ja Lei Cao (2021). "Tag-Aware Recommender System Based on Deep Reinforcement Learning". eng. *Mathematical problems in engineering* 2021, s. 1–12. ISSN: 1024-123X.