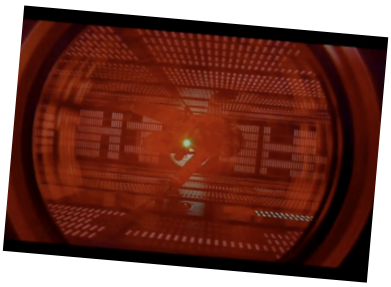

582216 Johdatus tekoälyyn

Syksy 2011
T. Roos

KURSSIKOODI: 582216
OPINTOPISTEET: 4.0
ERIKOISTUMISLINJA: Algoritmit ja koneoppiminen
TASO: Aineopinnot
KUVAUS:
Kurssilla käydään läpi tekoälyn ongelma-alueita ja niihin liittyviä ratkaisumenetelmiä koostuen luennoista, ohjelmointitehtävistä, harjoitustehtävistä, sekä kurssikokeesta.
KURSSIKOE: 21.10.2011 klo 09.00 A111 & B123

KURSSIN SISÄLTÖ eli "TULEEKO TÄMÄ KOKEESEEN???"

Kurssin päätavoitteena on saada käsitys tekoälyn perusongelmista, -sovelluksista ja -menetelmistä, sekä tekoälyn tärkeimmistä kehitysaskeleista sen historian kuluessa. Painotus on nykytutkimuksella ja käytännön sovelluksilla. Erityinen tavoite oli luoda yhteyksiä teorian ja käytännön välillä toteuttamalla luennolla opetettujen periaatteiden mukaisesti oikeasti hyödyllisiä työkaluja.

HUOM: Kokeeseen saa ottaa mukaan *käsinkirjoitetun A4-kokoisen (kaksipuolisen) lunttilapun,* joka pitää *palauttaa koepaperin mukana.*

Alla mainittavat asiat kuuluvat koealueeseen (poislukien ne asiat, joiden erikseen mainittavan olevan koealueen ulkopuolella):

Luento 1: Mitä on tekoäly?

Luennon alussa käytiin läpi kurssin logistiikkaa ja suorittamiseen liittyviä asioita. Lisäksi kuvailtiin tekoälyä tutkimusalueena. Jaottelu GOFAI ("Good Old-Fashioned AI") vs "moderni AI" esitettiin jo tässä vaiheessa ja siihen palattiin usein kurssin kuluessa.

Tekoäly esiintyy usein kulttuurissa (elokuvat, kirjat, pelit, ...), jossa on tapana painottaa kauhuskenaarioita.

Tekoälyn tavoitteet voidaan jaotella yhtäältä akselilla älykäs–ihmismäinen ja toisaalta akselilla ajattelee–toimii. "Vahva tekoäly" viittaa älykkäästi ajattelevaan ja itsestään tietoiseen tekoälyyn. "Heikolla tekoälyllä" tarkoitetaan älykkäästi toimivaa ohjelmaa tai agenttia. Ihmismäisesti toimiva kone läpäisee Turingin kokeen.

Kulttuuristen viitteiden ja filosofisen pohdinnan jälkeen siirryttiin ripeästi eteenpäin ja kysyttiin "Mitä tekoäly oikeasti on?" Sovellusten (itsestään ajava auto, Jeopardy-pelin voittava Watson-tietokone, Google, Amazonin suosittelualgoritmi) ja konferenssien ohjelmiston perusteella tekoälytutkimus ja sen sovellukset keskittyvät tyypillisesti pienten yksittäisten ongelmien, kuten konenäön, reitinoptimoinnin, konekäännöksen, suosittelun, tiedonhaun, jne. ratkaisuun. Suurten kysymysten ("Mitä on tietoisuus?" ja "Mitä eilen tapahtui BB-talossa?") pohdinnasta on enimmäkseen luovuttu hedelmättömänä.

Luento 2: Etsintä ongelmanratkaisuna

Luennon alussa palautettiin mieleen TiRa-kurssilta tutut leveys- ja syvyyssuuntainen haku, joiden erona huomattiin olevan pelkästään solmujen tallettamiseen käytettävä tietorakenne (jono vs pino).

Hakualgoritmeja voi käyttää ongelmanratkaisuun, kunhan ongelman saa muotoiltua sopivaan muotoon. Lähetyssaarnaajien ja kannibaalien joenylitystä voi kuvata tilakaaviona, jossa sallitut tilat ja niiden väliset siirtymät määräävät hakuavaruuden. Ongelma voidaan ratkaista etsimällä (miellellään lyhyt) reitti alkutilan ja lopputilan välillä.

Toisena esimerkkinä esitettiin sudokualgoritmi, joka vastaa syvyyssuuntaista hakua, kun tiloja ovat kelvolliset osittaiset ratkaisut ja siirtymät vastaavat yhden numeron lisäämistä.

Luennon varsinainen pähkinä oli heuristinen haku, jossa jonon tai pinon asemesta solmut talletetaan ns. prioriteettijonoon, josta ne poimitaan paras-ensin–järjestyksessä: kustannusarvioltaan paras talletettu solmu poimitaan aina ensimmäisenä.

Jos kustannusarviona käytetään polkukustannuksen (nykyiseen solmuun päättyvän polun kustannus) ja ns. heuristiikan summaa

$$f(N) = g(N) + h(N),$$

saadaan erittäin käyttökelpoinen menetelmä eli A*-haku. Jos heuristiikka ei koskaan yliarvioi nykyisestä solmusta maaliin johtavan polun pituutta, löytää A* aina optimaalisen reitin. Yleensä A* löytää optimaalisen reitin vieläpä melko tehokkaasti eli käymättä läpi valtavaa määrää eri reittejä.

A*-haun sovelluksena esitettiin kaikkien suosikkilaskuharjoitus-tehtävä, Reittiopas.

Luento 3: Pelit

Tietokonepelit ovat yksi tärkeimmistä (ja kiinnostavimmista) tekoälyn sovellusalueista. Luennon aluksi ihmeteltiin StarCraft-peliä pelaavien tekoälyalgoritmien vuosittaista kilpailua[1].

Pelipuu, jossa kukin pelin tila vastaa solmua ja jokainen sallittu siirto vie eri alipuuhun, on keskeinen käsite yksinkertaisia kahden pelaajan pelejä pelattaessa. Puussa vuorottelevat Min- ja Max-pelaajien vuoroja

[1] StarCraft-kilpailuun osallistuu vuosittain monia yliopistojoukkueita ympäri maailmaa. Eikö olisi hauskaa, jos meiltäkin?

vastaavat tasot. Puun lehtinä ovat pelin lopputilat, joissa suuret luvut ovat Max-pelaajan tavoitteena ja toisinpäin.

Pienessä pelipuussa optimaalinen peli voidaan laskea minimax-algoritmilla, jossa kutakin vuoroa (Min tai Max) vastaa yksinkertainen rekursiivinen funktio. Kun jokaisen solmun arvo on laskettu, voidaan helposti valita optimaalinen pelisiirto kussakin solmussa.

Alpha-beta–karsinnassa minimax-algoritmin kumpaakin funktiota muokataan lisäämällä niihin tieto α- ja β-arvot, joiden perusteella voidaan osa alipuista jättää kokonaan laskematta ilman, että olisi vaaraa, että päädytään pelaamaan ei-optimaalisesti.

(Huomaa, että alpha-beta-karsintaa selventävä video löytyy kurssin sivulta 3. luennon kohdalta.)

Shakki on esimerkki pelistä, jonka optimaaliset siirrot voisi periaatteessa laskea minimax-algoritmilla, mutta se ei käytännössä ole mahdollista pelipuun suuren koon vuoksi. Siksi on tarpeen soveltaa heuristista arviointifunktiota, joka antaa arvon mille tahansa pelitilanteelle.

Luento 4: Logiikka tekoälyssä

Logiikka oli 1980-luvun alkupuolelle saakka tekoälyn keskeisin lähestymistapa. Ongelmiksi muodostuivat etenkin ns. kehysongelma ja epävarman tiedon käsittely sekä jossain määrin loogiset paradoksit, ratkeamattomuus, jne. Nykyään logiikkaa käytetään joillain tekoälyn osa-alueilla, kuten automaattisessa teoreemantodistuksessa ja ohjelmien oikeellisuuden todentamisessa, mutta suurimmalta osin sen ovat korvanneet modernin tekoälyn menetelmät, kuten neuroverkot ja todennäköisyysmallinnus.

Predikaattilogiikan käsitteitä:

- muuttujat: A, B, C, ...
- vakiot: a, b, c, darth vader, reinikainen...
- predikaatit: isä(◦, ◦), lintu(◦), kissa(◦)
- lauseet: isä(X,Y) $\wedge$ VANHEMPI(Y,Z) $\Rightarrow$ ISOISÄ(X,Z)

Päättely voidaan osin automatisoida esim. Prolog-kielessä. Käytännössä on kuitenkin vaikea kirjoittaa ohjelmaa siten, että kaikki relevantti tieto on koodattu oikein: robotille unohdettiin mainita, että jos pommi on vetokärryn kyydissä, se tulee kärryn mukana huoneesta ulos. Lisäksi joidenkin lauseiden todistaminen voi olla käytännössä niin hankalaa, että vaikka ne pitävätkin paikkansa (esim. P vs NP -ongelma tai mikä tahansa muu toistaiseksi ratkaisematon matemaattinen

pulma), ei niitä voida koneellisesti todistaa: robotti voi jäädä pohtimaan onko Darth Vader Luke Skywalkerin isä vai ei, jne jne, kunnes pommi räjähtää (ks. kehysongelma luentokalvoilla).

Gödelin epätäydellisyyslause sanoo vieläpä, että joitakin tosia lauseita ei voidakaan todistaa! (Mikä on siis eri asia kuin se, että joidenkin lauseiden todistaminen kestää liian kauan.)

Prologin syntaksi:

PÄÄ := VARTALO

(eli johtopäätös ⇐ premissit).

VARTALO voi olla lause, joka sisältää useita predikaatteja yhdistettynä konjunktioilla (“ja”, merkitään ‘,’) ja disjunktioilla (“tai”, merkitään ‘;’). Predikaatin negaatio merkitään ‘\+’.

CYC on esimerkki hankkeesta, jonka toivotaan johtavan ihmistasoiseen älykkyyteen kasaamalla yhteen valtava määrä logiikan avulla esitettyä tietämystä. Suuret odotukset ovat toistaiseksi jääneet joka kerta täyttämättä.

Tässä vaiheessa kurssia jätettiin taakse GOFAI ja siirryttiin modernin tekoälyn puolelle.

5. *Luento: Päättely epävarmuuden vallitessa I. Probabilistinen päättely*

Oikeassa maailmassa pitää pystyä ottamaan huomioon monenlaisia epävarmuustekijöitä. Esimerkkinä auto, joka ei starttaa: ongelma voidaan yrittää paikallistaa laskemalla eri vikojen todennäköisyys. Todennäköisyys muuttuu, kun saadaan uutta tietoa (esim. “radio soi”).

Bayes-verkko koostuu joukosta solmuja (muuttujat), niiden välisistä kaarista (suorat riippuvuudet), ja parametreista, jotka määräävät kunkin solmun ehdollisen todennäköisyysjakauman annettuna ko. solmun vanhempien arvot. Esimerkki parametriarvosta ja sen määräämästä todennäköisyydestä on

$$P(\text{“KÄYNNISTYY”} \mid \text{“AKUSSA VIRTAA”} \wedge \text{“BENSAA”}) = 0.99.$$

Kun Bayes-verkko ja sen parametrit on määrätty, voidaan laskea minkä tahansa monikon (monikko on lista, joka määrää kunkin muuttujan arvon) todennäköisyys, esim.:

$$P(A,R,S,B,\neg K,L) = P(A)\,P(R|A)\,P(S|A)\,P(B)\,P(\neg K|S,B)\,P(L|\neg K).$$

(Huomaa, että kunkin muuttujan kohdalla ehdollistajana, eli pystyviivan oikealla puolella on ko. muuttujan vanhemmat.) Bayes-verkosta on myös helppo generoida monikkoja.

Mikä tahansa ehdollinen todennäköisyys voidaan laskea hyödyntämällä Bayes-verkkoa. Jos halutaan laskea todennäköisyys $P(\neg A|\neg L)$, voidaan generoida iso kasa monikkoja, laskea kuinka monessa niistä pätee $\neg A \wedge \neg L$ sekä kuinka monessa pätee $\neg L$, ja jakaa edellinen lukumäärä jälkimmäisellä. Näin saatu arvo ei kuitenkaan ole tarkalleen oikea vaan satunnaisapproksimaatio, jonka tarkkuudesta ei aina ole takeita.

Huom: Muitakin menetelmiä on ja jotkut niistä tuottavat tarkan arvon. Luennon alkupuolella esitettiin yksi tällainen menetelmä ("matemaattinen lähestymistapa"), mutta sen yksityiskohdat *eivät* kuulu koealueeseen.

6. *Luento: Kuvankäsittely, grafiikka -- ei kuulu koealueeseen!*

   Tällä luennolla katseltiin YouTube-videoita erilaisista kuvankäsittelyyn ja tietokonegrafiikkaan liittyvistä aiheista. Luennon tarkoitus oli antaa yleiskuva aihealueesta ja nykyaikaisen kuvankäsittelyn mahdollisuuksista. Etenkin hahmontunnistuksesta lisää asiaa 8. luennossa. (Huom: Se että SURF-piirteitä käsiteltiin 6. luennossa ei tarkoita, että ne eivät kuuluisi koealueeseen, koska niitä käsiteltiin myös 8. luennolla!)

7. *Luento: Päättely epävarmuuden vallitessa II. Case study: Naive Bayes-roskapostinsuodatus*

   Luennon alussa käytiin läpi tarvittava määrä todennäköisyyslaskentaa. Se sisältyy koealueeseen vain siltä osin kuin se on tarpeellista alla olevan sisällön kannalta. (Ei siis kysytä luentokalvon sivun 6 kaavoja erikseen, mutta niiden osaamisesta on hyötyä esim. alla olevien kahden pointin (a ja b) hallitsemisessa.)

   Bayesin kaavalla voi laskea todennäköisyyden, että yhden sanan mittainen viesti on roskapostia. Pastori Bayesilla oli kaksi vinkkiä pitempien viestien käsittelyyn:

   a) On kätevämpi laskea osamäärä

      $$P(SPAM|EVIDENSSI)/P(\neg SPAM|EVIDENSSI)$$

      kuin pelkkä osoittaja.

   b) Naivi Bayes-mallissa pätee

      $$P(SANA_2=IS|SANA_1=VIAGRA,SPAM)=P(SANA_2=IS|SPAM),$$

      eli sanat ovat riippumattomia toisistaan, jos oletetaan, että viesti on spammia (ja sama pätee hammille).

Nämä vinkit helpottivat laskentoa merkittävästi ja jäljelle jäi melko yksinkertainen osamäärien tulo (ks. luentokalvojen sivu 16). Tähän perustuva roskapostintunnistusalgoritmi on helppo toteuttaa (luentokalvojen sivu 20).

Implementoidessa on syytä murehtia yli- ja alivuodoista, jotka voidaan välttää ylläpitämällä osamäärän (ks. a-kohta yllä) logaritmia, mikä vaatii minimaalisia muutoksia koodiin.

Todennäköisyydet määräävien parametrien vetäminen hatusta on hankalaa, joten ne kannattaa arvioida oikeasta datasta. Nollatodennäköisyydet eivät ole hyvä asia (koska ne saattavat johtaa tilanteeseen, jossa P(EVIDENSSI|SPAM) ja P(EVIDENSSI|HAM) ovat molemmat nollia ja siitä ei hyvä seuraa. Siksi nollat on syytä korvata esim. hyvin pienellä vakiolla, kuten 0.0001.)

Luento 8: Kuvankäsittely

Digitaaliset signaalit kuten kuva ja ääni vaativat erilaisia menetelmiä kuin symbolinen data. (Miksi?)

Kuva voidaan tulkita joukkona pikseliarvoja ja vastaavasti ääni voidaan tulkita värähtelyä kuvaavana lukujonona.

Kuvien ja äänen tunnistamisessa on olennaista pyrkiä löytämään piirteet, jotka eivät ole herkkiä muuttumaan, kun samasta kohteesta saadaan uusi havainto.

Kuvantunnistuksessa etenkin ns. invariantit piirteet (invariant features) ovat suosittuja.

SURF-piirteenirrotus ja hahmontunnistus koostuu kolmesta vaiheesta:

1. Avainpisteiden valinta: Etsitään kuvasta joukko pisteitä maksimoimalla ns. Hessen matriisin determinantti, joka kuvaa paikallisen intensiteetin vaihtelua. Tuloksena löydetään "blobeja". (Tästä on tarkoitus sisäistää yleinen idea. Teknisiä yksityiskohtia ei vaadita.)
2. Piirteiden kuvaus: Tarkastellaan avainpisteiden ympäristöä ja määritetään 64 lukuarvoa sisältävä piirrevektori. Piirrevektorilla on lisäksi skaala (koko) ja orientaatio (suunta).
3. Hahmontunnistus: Etsitään piirteet kahdesta kuvasta ja pyritään löytämään piirteitä, jotka toistuvat molemmissa. Piirreparit tunnistetaan piirrevektorien (euklidista) etäisyyttä vertaamalla. Tunnistusta voidaan vielä parantaa geometrisilla rajoitteilla, jos hahmon rakenteesta (esim. kasvot) on lisätietoa.

Luentokalvojen lopussa oleva Elo ranking liittyy shakkiturnaukseen *eikä* se sisälly koealueeseen.

Luento 9:Vierailuluento Patrik Hoyer – Koneoppiminen

Koneoppimisen avulla on saavutettu viime aikoina läpimurtoja monella alalla (itsestään ajavat autot, roskapostin suodatus, tiedonhaku, tekstintunnistus, suosittelu, konekäännös, jne).

Koneoppimisen tavoitteena on automatisoida oppiminen, jolloin jonkin ongelman ratkaisua ei tarvitse syöttää valmiina tietokoneeseen vaan sen voi tuottaa automaattisesti opetusaineistosta. Tämä säästää käsityötä (vaikkapa roskapostisuodattimen parametrien säätämisessä) ja etenkin jos saatavilla on suuri määrä opetusaineistoa, yleensä johtaa parempaa tulokseen kuin käsin koodattu ratkaisu.

Koneoppimisessa keskeisiä käsitteitä ovat tehtävä (ongelma ja tavoitteen kuvaus, sallitut ratkaisut), hyvyysmitta (ratkaisun hyvyys), esimerkit/data (aineisto, josta opitaan). Usein datan esittäminen sopivassa muodossa on haasteellista, vrt. kuvantunnistuksessa käytettävät piirteet.

Koneoppimisen lajeja ovat:

1. Ohjattu (supervised) oppiminen: Esimerkit ovat muotoa (x,y), missä x on syöte ja y haluttu tuloste (esim luokittelu tai ennuste jatkuvasta arvosta).
2. Ohjaamaton (unsupervised) oppiminen: Esimerkit ovat muotoa x, missä tavoitteena on luoda datasta esitys, joka on helpommin ymmärrettävissä tai muutoin hyödynnettävissä kuin raakadata. Tyypillinen esimerkki: klusterointi (eli ryvästys).
3. Vahvistusoppiminen (tai palauteoppiminen; reinforcement learning): Palaute on epäsuorempaa kuin ohjatussa oppimisessa.

Luennolla käsiteltiin enimmäkseen ohjattua oppimista, mutta on syytä painaa mieleen, että se ei ole ainoa koneoppimisen laji. Muita lajeja ei kuitenkaan tarvitse erityisesti hallita kokeessa.

Lähimmän naapurin luokittelija etsii opetusdatasta sen syötevektorin x^{train}, joka on lähinnä uutta syötevektoria x^{test} ja palauttaa edellistä vastaavan luokan y^{train}. k-lähimmän naapurin naapurin luokittelija hyödyntää useampaa lähintä naapuria.

Muita menetelmiä, joita voi soveltaa samaan ongelmaa ovat naive Bayes -luokittelija ja neuroverkot, joita käsiteltiin kurssilla enemmän muilla luennoilla

Luento 10: Vierailuluento: Jouko Strömmer – Robotiikka

Luennolla esiteltiin robotiikkaa Lego Mindstorms -robottien avulla käytännön tasolla. Tavoitteena oli saavuttaa taito toteuttaa yksinkertaisia toiminnallisuuksia, kuten laskuharjoituksissa tehtävinä olleet viivan seuraaminen valosensorin avulla ja pysäköinti ultraäänisensorin avulla.

Tässä vielä esimerkiksi toimivaksi havaittu viivanseurausmenetelmä:

```
while(true) {
    while(light.getLightValue > 40)
        Motor.A.forward();   // Käännytään mustalle
    while(light.getLightValue < 40)
        Motor.A.forward();   // Jatketaan kääntymistä kunnes taas valkoisella

    Motor.A.stop();

    while(light.getLightValue > 40)
        Motor.B.forward();   // Käännytään takaisin mustalle
    while(light.getLightValue < 40)
        Motor.B.forward();   // Jatketaan kääntymistä kunnes taas valkoisella

    Motor.B.stop();
}
```

Yleisempänä oppimistavoitteena oli havaita, että robotiikassa joutuu törmäämään melkoiseen määrään tosielämän ongelmia, kuten sensorilukemien ja liikkumisen omituisuudet. Nämä tulivat selväksi sekä luennon demoissa että laskuharjoituksissa.

Luennolla esitettyjä teoreetisempia asioita (robotiikkaohjelmoinnin yleisemmät periaatteet, arbitraattorit) *ei* tule kokeeseen.

Luento 11: Neuroverkot

Neuroverkot ovat olleet tekoälyn hitoriassa logiikan ohella yksi keskeisimmistä aihealueista. Ne voidaan nähdä Turingin koneisiin ja tavalliseen (proseduraaliseen) ohjelmointiin verrattuna kokonaan erillisenä laskennan mallina. Toisaalta ne voidaan käsittää pelkästään yhtenä mallien joukkona, johon sovelletaan samoja periaatteita kuin muihinkin malleihin (esim. probabilistiset ja koneoppimisen menetelmät). Jälkimmäinen näkökanta painottuu etenkin silloin, kun neuroverkkoja simuloidaan tavallisella tietokoneella eikä erillisellä laitteistolla.

Luonnollisista neuroverkoista (kuten aivoista) kopioitu perusidea keinotekoisissa neuroverkoissa on usean yksinkertaisen prosessorin kytkeminen yhteen laajaksi verkostoksi, jossa prosessorit viestivät

keskenään. Muilta ominaisuuksiltaan luonnolliset ja keinotekoiset neuroverkot voivat olla hyvinkin erilaisia.

Luennolla tutustuttiin kolmeen neuroverkkotyyppiin:

1. Eteenpäin syöttävät (feedforward) neuroverkot, joissa laskenta etenee syötteistä aina kohti tulosteita, eikä verkossa esiinny "silmukoita".

2. Takaisinkytkeytyvät (recurrent) neuroverkot, joissa verkon yhteydet muodostavat silmuikoita tai syklejä, joissa neuronin A tuloste voi olla neuronin B syöte, neuronin B tuloste voi olla neuronin C syöte, ja neuronin C tuloste voi olla neuronin A syöte. Tällaisissa verkoissa ei välttämättä ole lainkaan lopullista tulostetta, kuten eteenpäinsyöttävissä verkoissa, vaan laskennan "tulos" luetaan verkon dynaamisesta käyttäytymisestä.

3. Itseorganisoiva (self-organizing) kartta eli Kohosen SOM. Solmut kilpailevat siitä, kuka saa aktivoitua syötteestä ja muokkaavat omaa tilaansa muistuttamaan enemmän syötettä. Myös aktivoituneen voittajasolmun naapureiden tilaa muokataan saman suuntaisesti. Tuloksena on kartta, jossa samankaltaiset syötteet aktivoivat lähekkäin olevia neuroneita.

Neuroverkkojen sovellukset vaihtelevat tyypeittäin. Eteenpäin syöttävän verkon sovelluksia ovat tyypilliset ohjatun koneoppimisen ongelmat. Takaisinkytkeytyviä verkkoja voidaan soveltaan kohinanpoistoon (esim käsin kirjoitetuista numeroista) tai vikasietoisina muisteina (ns. autoassociative memory). Kohosen karttoja puolestaan voidaan käyttää mm. monenlaisen tiedon visualisointiin.

Kunkin verkkotyypin käytöstä esitettiin periaatteet: kuinka verkkoa opetetaan, mitä ovat syötteet ja mikä on toivottu tulos. Itse opetusalgoritmien yksityiskohtia *ei* tarvitse opetella kokeeseen.

Päivitetty 14.10.2011 T. Roos