

KOMPLEKSISET PÄÄTÖKSET

4.3

- Agentin hyötyarvo riippuukin nyt sarjasta toimintapäätöksiä
- Oheisessa 4×3 ruudukkomaailmassa agentti tekee siirtymäpäätöksen (Y, O, V, A) jokaisella ajanhetkellä
- Kun päädytään toiseen maalitiloista, niin toiminta lakkaa
- Maailma on täysin havainnoitava — agentti tietää sijaintinsa

			+1
			-1
Lähtö			

- Jos maailma on deterministinen, niin agentti pääsee lähtötilasta aina lopputilaan +1 siirtymän [Y, Y, O, O, O]
- Koska toiminnot kuitenkin ovat epäluotettavia, niin siirtymäsekvenssi ei aina johda haluttuun tulokseen
- Olkoon niin, että aiottu toiminto toteutuu todennäköisyydellä 0.8 ja todennäköisyydellä 0.1 liike suuntautuu kohtisuoriin suuntiin
- Jos agentti törmää maailmansa rajoihin, niin toiminnolla ei ole vaikutusta
- Tällöin sekvenssi [Y, Y, O, O, O] johtaakin maaliin vain todennäköisyydellä $0.8^5 = 0.32768$
- Lisäksi agentti voi päätyä maaliin sattumalta kiertämällä esteen toista kautta todennäköisyydellä $0.1^4 \times 0.8$, joten kokonais-todennäköisyys on 0.32776

- *Siirtymämalli* antaa todennäköisyydet toimintojen tuloksille maailman kaikissa mahdollisissa tiloissa
- Merk. $T(s, a, s')$ on tilan s' saavuttamistodennäköisyys kun toiminto a suoritetaan tilassa s
- Siirtymät ovat *Markovilaisia* sikäli, että vain nykytila s , ei aiempien tilojen historia, vaikuttaa siihen saavutetaanko s'
- Vielä on määrättävä hyötyfunktio
- Päätösongelma on sekventiaallinen, joten hyötyfunktio riippuu tilajonosta — ympäristöhistoriasta — eikä vain yhdestä tilasta
- Toistaiseksi agentti saa kussakin tilassa s *palkkion* (reward) $R(s)$, joka voi olla positiivinen tai negatiivinen

- Esimerkissämme palkkio on -0.04 kaikissa muissa tiloissa paitsi lopputiloissa
- Ympäristöhistorian hyöty puolestaan on palkkioiden summa
- Jos esim. agentti saavuttaa lopputilan $+1$ kymmenen siirtymän jälkeen, niin hyöty on 0.6
- Pienen negatiivisen palautteen tarkoitus on saada agentti maailmaan mahdollisimman nopeasti
- Täysin havainnoitavan maailman sekventiaallinen päätösongelma, kun
 - siirtymämalli on Markovilainen ja
 - palkkiot summataan,
 on *Markov-päätösongelma* (Markov decision problem, MDP)

- MDP on kolmikko
 - alkutila S_0 ,
 - siirtymämalli $T(s, a, s')$ ja
 - palkkiofunktio $R(s)$
- Ratkaisuksi MDP:hen ei kelpaa kiinteä siirtymäsekvenssi, koska se voi kuitenkin johtaa muuhun tilaan kuin maaliin
- Vastauksen onkin oltava **politiikka**, joka määrää toiminnan kussakin tilassa, johon agentti voi päätyä
- Politiikan π suosittama toiminto tilassa s on $\pi(s)$
- Täydellisen politiikan avulla agentti aina tietää mitä tehdä heittipä epädeterministinen toimintaympäristö sen mihin tilaan tahansa

- Joka kerta kun politiikkaa sovelletaan, maailman stokastisuus johtaa eri ympäristöhistoriaan
- Politiikan laadun mittari onkin sen mahdollisesti tuottamien ympäristöhistorioiden hyödyn odotusarvo
- Optimaalisen politiikan π^* odotusarvo on korkein

→	→	→	+1
↑		↑	-1
↑	←	←	←

- Politiikka antaa eksplisiittisen kontrollin agentin käyttäytymiselle ja samalla yksinkertaisen refleksiivisen agentin kuvauksen

• $-0.0221 < R(s) < 0$:

→	→	→	+1
↑		←	-1
↑	←	←	↓

• $-0.4278 < R(s) < -0.0850$:

→	→	→	+1
↑		↑	-1
↑	→	↑	←

- Äärettömän horisontin tapauksessa agentin toiminta-ajalle ei ole asetettu ylärajaa
- Jos toiminta-aika on rajoitettu, niin eri aikoina samassa tilassa voidaan joutua tekemään eri toimintopäätöksiä — optimaalinen politiikka ei ole *stationäärinen*
- Sen sijaan äärettömän horisontin tapauksessa ei ole syytä muuttaa tilan toimintaa kerrasta toiseen, joten optimaalinen politiikka on stationäärinen
- Tilajonon s_0, s_1, s_2, \dots diskontattu palkkio on

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots,$$
 missä $0 \leq \gamma \leq 1$ on diskonttaustekijä

- Kun $\gamma = 1$, niin ympäristöhistorian palkkioksi saadaan erikoistapauksena additiivinen palkkio
- Lähellä nollaa oleva γ puolestaan tarkoittaa tulevien palkkioiden merkityksen pienenemistä
- Jos äärettömän horisontin maailmassa ei ole lainkaan saavutettavaa maalitilaa, niin ympäristöhistorioista tulee äärettömän pitkiä
- Additiivisilla palkkioilla myös hyötyarvot kasvavat yleisesti ottaen äärettömiksi
- Diskontatuilla palkkioilla ($\gamma < 1$) äärettömänkin jonon palkkio on äärellinen

- Olk. R_{\max} palkkioiden yläraja. Tällöin geometrisen sarjan summana

$$\sum_{t=0, \dots, \infty} \gamma^t R(s_t) \leq \sum_{t=0, \dots, \infty} \gamma^t R_{\max} = R_{\max} / (1 - \gamma)$$

- Kelvollinen politiikka (proper policy) takaa agentin pääsevän lopputilaan silloin kun ympäristössä on lopputiloja
- Tällöin äärettömistä tilajonoista ei ole huolta ja voidaan jopa käyttää additiivisia palkkioita
- Optimaalinen politiikka diskontatuilla palkkioilla on

$$\pi^* = \arg \max_{\pi} E[\sum_{t=0, \dots, \infty} \gamma^t R(s_t) \mid \pi],$$

missä odotusarvo lasketaan yli kaikkien mahdollisten tilajonojen, jotka voisivat tulla kyseeseen politiikalla

Arvojen iterointi

- Optimaalisen politiikan selvittämiseksi lasketaan tilojen hyötyarvot ja käytetään niitä optimaalisen toiminnon valitsemiseen
- Tilan hyödyksi lasketaan sitä mahdollisesti seuraavien tilajonojen odotusarvoinen hyöty
- Luonnollisesti jonot riippuvat käytetystä politiikasta π
- Olkoon s_t tila, jossa agentti on kun π_t :tä on noudatettu t askelta
- Huom. s_t on satunnaismuuttuja
- Nyt

$$U^\pi(s) = E[\sum_{t=0, \dots, \infty} \gamma^t R(s_t) \mid \pi, s_0 = s]$$

- Tilan todellinen hyötyarvo $U(s)$ on $U^{\pi^*}(s)$
- Palkkio $R(s)$ siis kuvaa tilassa s olemisen lyhyen tähtäyksen hyödyllisyyttä, kun taas $U(s)$ on s :n pitkän tähtäyksen hyödyllisyys siitä eteenpäin laskien
- Esimerkkimaailmassamme maalitilan lähellä olevilla tiloilla on korkein hyötyarvo, koska niistä matka on lyhin

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

- Nyt voidaan soveltaa hyödyn odotusarvon maksimointia

$$\pi^*(s) = \arg \max_a \sum_{s'} T(s, a, s') U(s')$$
- Koska tilan s hyöty nyt on diskontattujen palkkioiden summan odotusarvo tästä tilasta eteenpäin, niin se voidaan laskea:
 - Välitön palkkio tilassa s , $R(s)$, +
 - seuraavan tilan diskontatun hyödyn odotusarvo olettaen, että valitaan optimaalinen toiminto

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$
- Tämä on **Bellman-yhtälö**
- Toimintaympäristössä, jossa on n tilaa, on myös n Bellman-yhtälöä



- Bellman-yhtälöiden yht'aikaiseen ratkaisemiseen ei voi käyttää lineaaristen yhtälöryhmien tehokkaita ratkaisumenetelmiä, koska **max** ei ole lineaarinen operaatio
- Iteratiivisessa ratkaisussa aloitamme tilojen hyötyjen mv. arvoista ja päivitämme niitä kunnes saavutetaan tasapainotila

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U_i(s'),$$
 missä indeksi i viittaa iteraation i hyötyarvioon
- Päivitysten toistuva soveltaminen päättyy taatusti tasapainotilaan, jolloin saavutetut tilojen hyötyarviot ovat ratkaisu Bellman-yhtälöihin
- Löydetyt ratkaisut ovat yksikäsitteisiä ja vastaava politiikka on optimaalinen



Politiikan iterointi

- Alkaen lähtöpolitiikasta π_0 toista
- **Politiikan arviointi:** laske kaikille tiloille hyötyarvo $U_i = U^{\pi_i}$ politiikkaa π_i sovellettaessa
- **Politiikan parantaminen:** perustuen arvoihin U_i laske uusi hyödyn odotusarvon maksimoiva politiikka π_{i+1} (vrt. 259)
- Kun jälkimmäinen askel ei enää muuta hyötyarvoja, niin algoritmi päättyy
- Tällöin U_i on Bellman-päivityksen kiintopiste ja ratkaisu Bellman-yhtälöihin, joten vastaavan politiikan π_i on oltava optimaalinen
- Äärellisellä tila-avaruudella on vain äärellinen määrä politiikkoja, jokainen iteraatio parantaa politiikkaa, joten politiikan iterointi päättyy lopulta

- Koska kullakin kierroksella politiikka on kiinnitetty, niin politiikan arvioinnissa ei ole tarvetta maksimoida yli toimintojen
- Bellman-yhtälö yksinkertaistuu:

$$U_i(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s')$$
- Koska epälineaarista maksimoinnista on päästy eroon, on tämä lineaarinen yhtälö
- Lineaarinen yhtälöryhmä, jossa on n yhtälöä ja niissä n tuntematonta muuttujaa voidaan ratkaista ajassa $O(n^3)$ lineaarialgebran menetelmin
- Kuutiollisen ajan sijaan voidaan tyytyä approksimoimaan politiikan laatua ajamalla vain tietty määrä yksinkertaisia arvojen iterointi -askelia kelvollisten hyötyarvioiden saamiseksi

KONEOPPIMINEN

5

- Älykäs agentti voi joutua oppimaan mm. seuraavia seikkoja:
 - Kuvaus nykytilan ehdoilta suoraan toiminnolle
 - Maailman relevanttien ominaisuuksien päättelyminen havaintojonoista
 - Maailman muuttumisen seuraaminen ja toimintojen vaikutukset
 - Maailman mahdollisten tilojen hyötyarviot
 - Toimintojen arvottaminen
 - Maailman tilojen tunnistaminen hyödyn maksimoimiseksi

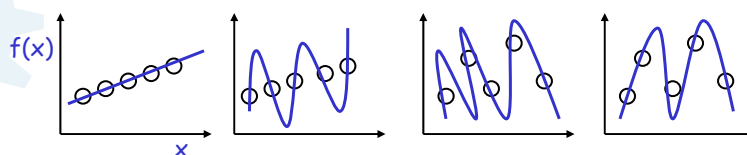


- Oppijan saama palaute määrää oppimisen tyypin
- **Ohjatussa** oppimisessa (supervised l.) tavoitteena on oppia kuvaus syöteiltä tuloksille annettujen esimerkkien perusteella
- **Ohjaamattomassa** oppimisessa (unsupervised l.) annettuja syötteitä pyritään ryhmittelemään samankaltaisiin ryhmiin ilman erillisiä tulosarvoja
- **Palautteoppimisessa** (reinforcement l.) oppija saa (ohjattua oppimista epämääräisempää) palautetta, jonka perusteella sen tulee oppia syötteistä
- Tietämyksenesisitys määrää vahvasti oppimisalgoritmin toiminnan



Induktiivinen oppiminen

- Ohjatussa oppimisessä tavoitteena on oppia tuntematon funktio f annettuna *esimerkkejä* $(x, f(x))$, missä x on syöte(vektori)
- *Induktiivisessa inferenssissä* tuloksena on *hypoteesi* h , joka on oppijan approksimaatio f :stä
- Tavoiteltava hypoteesi *yleistää* (generalize) esimerkkien perusteella uusien tapausten ennustamiseksi
- Hypoteesi valitaan hypoteesiluokasta H
- Esimerkiksi kun sekä arvot x ja $f(x)$ ovat reaalilukuja, niin H voi olla mm. korkeintaan astetta k olevat polynomit:
 $3x^2 + 2, x^{17} - 4x^3$



- *Konsistentti hypoteesi* on annetun aineiston kanssa yhtäpitävä
- Mikä valita mahdollisesti monista konsistenteista hypoteeseistä?
- **Occamin partaveitsen** (Occam's / Ockham's razor) mukaan yksinkertaisin hypoteeseistä on paras
- Yksinkertaisuuden määrittäminen?

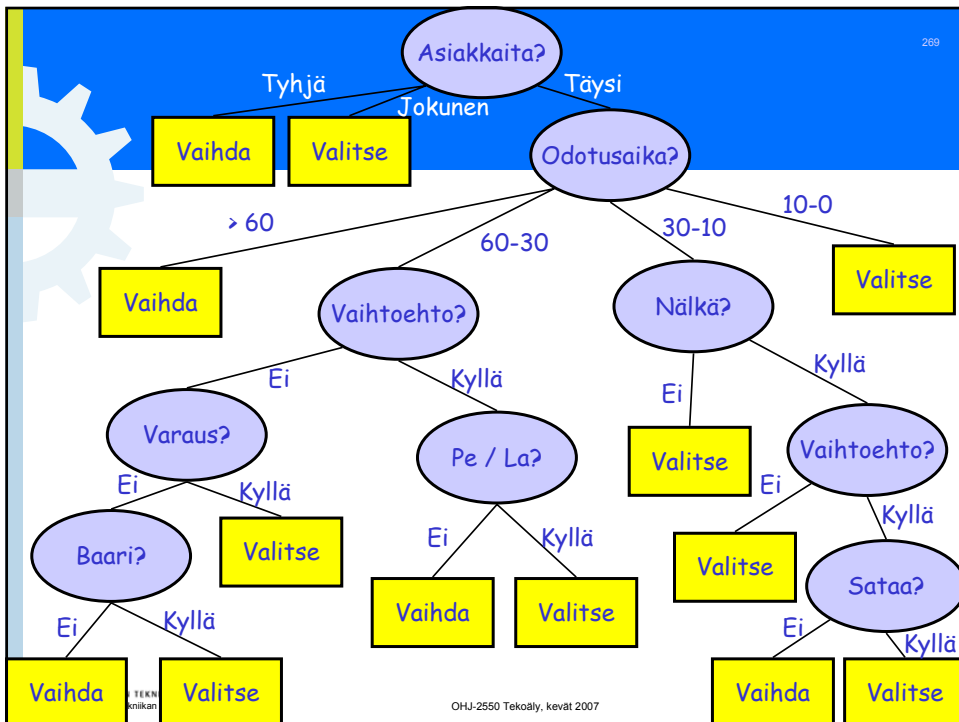
- Hypoteesiluokka ei aina sisällä konsistenttia hypoteesiä
- Toisaalta (paremman yleistyskyvyn nimissä) voi olla järkevää vaihtaa sovituksen tarkkuutta hypoteesin yksinkertaisuuteen
- Ts., tyytyä huonommin annettuun aineistoon sopivaan hypoteesiin, joka kuitenkin on yksinkertainen
- Hypoteesiluokka on rajoitettava, jotta hyvin aineistoon sopivan hypoteesin löytäminen olisi laskennallisesti tehokasta
- Koneoppiminen keskittyykin melko yksinkertaisten tietämyksen esitysmuotojen oppimiseen



Päätöspuiden oppiminen

- *Päätöspuun* (decision tree) syöte on attribuuttiarvoin ilmaistu olion tai tilanteen kuvaus
- Tuloksena on päätös — ennuste syötettä vastaavalle tulosarvolle
- Jos tulosarvot ovat diskreettejä, niin puu *luokittelee* syötteet
- Jatkuva-arvoisen funktion oppimista nimitetään *regressioksi*
- Syötteen käsittelemiseksi se ohjataan puun juuresta lähtien sisäsolmujen polkua aina lehteen, johon liittyvä päätös on syötteen tulosarvo
- Sisäsolmuihin liittyy attribuuttiarvon testi, jonka mukaan käsiteltävä tapaus ohjataan eteenpäin puussa






270

- (Kohtuullisen kokoinen) päätöspuu on helposti ymmärrettävä tietämyksen esitysmuoto
- Käytännössä tärkeä, heuristisesti opittavissa
- Edellisen esimerkin päätöspuu vastaa päätöstä siitä kannattaako ravintolaan jäädä odottamaan pöytää
- Sen ilmaisema predikaatti on

$$\forall s: \text{Valitse}(s) \Leftrightarrow (P_1(s) \vee \dots \vee P_n(s)),$$
 missä kukin ehdoista $P_i(s)$ on puun polkua juuresta **Valitse**-lehteen vastaavien testien konjunktio
- Eksponentiaalisen kokoisella päätöspuulla voidaan ilmaista mikä tahansa Boolean funktio

OHJ-2550 Tekoäly, kevät 2007

19.4.2007

 TAMPEREEN TEKNILLINEN YLIOPISTO
Ohjelmistotekniikan laitos

- Yleensä funktion esittämiseen riittää eksponentiaalista kokoa pienempi puu
- Eräät funktiot ovat kuitenkin vaikeita ilmaista päätöspuun, esim. **xor** ja **maj** vaativat eksponentiaalisen kokoisen puun
- Päätöspuut, kuten kaikki muutkin tietämyksen esitysmuodot, soveltuvat hyvin joidenkin funktioiden kuvaamiseen ja huonommin toisten
- n :n muuttujan Boolean funktion totuustaulussa on 2^n riviä, joten eri funktioita on 2^{2^n} kappaletta
- Esim. $n = 6 \Rightarrow 2^{2^6} > 18 \times 10^{18}$, joten konsistentin hypoteesin löytäminen on haastavaa

Päätöspuiden osittava oppiminen

- Algoritmin syöte on *opetusjoukko* (training set), joka koostuu joukosta esimerkkejä (X, y) , missä X on attribuuttiarvojen vektori ja y on arvoihin liitetty luokka-arvo
- Konsistentti puu jossa on oma juuri-lehti-polku kutakin opetusesimerkkiä kohden voitaisiin rakentaa helposti
- Tällöin ei kuitenkaan saavutettaisi yleistyskykyä
- Occamin partaveistä noudattaen meidän tulisi etsiä pienin opetusjoukon kanssa konsistentti puu, mutta pienimmän konsistentin puun löytäminen on laskennallisesti tehotonta (NP-kovaa)

- Menestyksekkäät päätöspuiden oppimisalgoritmit perustuvat heuristiseen pienehkön puun löytämiseen
- Lähtökohtana on valita tärkeimmät attribuutit ensin
- Koska tavoitteena on tapausten luokittelu, niin attribuutti on tärkeä, kun sen vaikutus luokittelussa on suuri
- Päätöspuun muodostaminen itsessään voidaan tehdä rekursiivisella algoritmilla
 - Ensin valitaan puun juureen se attribuutti, joka osoittautuu parhaaksi,
 - jaetaan opetusaineisto tämän attribuutin arvojen perusteella ja
 - jatketaan puun muodostamista alipuista samalla periaatteella

```

Tree growConsTree( Attrs A, Exs S )
{
  if ( all examples in S have class C )
    return an one-leaf tree labeled by C;
  else {
    select an attribute a from A;
    partition S into S1, ..., Sk by the value of a;
    for ( i = 1; i <= k; i++ )
      Ti = growConsTree( A - {a}, Si );
    return a tree T that has a in its root and
      Ti as its i-th subtree; }
}

```

- Jos algoritmi joutuu valitsemaan puun tyhjälle esimerkkien joukolla (opetusaineiston jaon yhteydessä), niin puuksi valitaan lehti, jonka ennustama luokka on koko opetusaineiston yleisin
- Jos taas attribuutit loppuvat kesken s.e. jäljellä olevassa aineistossa on vielä useamman luokan edustajia, niin aineisto on keskenään ristiriitainen i. *kohinainen* (noisy)
- Kohina voi olla seurausta riittämättömistä tilannetta kuvaavista attribuuteista tai se voi olla sovellusalueen aitoa epädeterminismiä
- Yleisimmän luokan ennustaminen on yksinkertainen tapa toimia tässä tilanteessa

Attribuutin valinta

- Puuhun valittavan attribuutin tulisi parantaa esimerkkien jakoa luokka-attribuutin arvon ennustamiseksi
- Parhaimmillaan attribuutti jakaisi esimerkit joukkoihin, joissa on vain yhden luokan edustajia
- Heikoimmillaan attribuutti ei muuta eri luokkien edustajien suhteellisia osuuksia juuri lainkaan
- Attribuuttien hyödyllisyyden mittaamiseen voidaan käyttää mm. sen antaman *informaation arvoa* eli *Shannon entropiaa*
- Informaatioteoriassa informaattiosisältöä mitataan bitein
- Yksi bitti riittää kyllä/ei-kysymykseen (kolikonheittoon) vastaamiseksi

- Yleisesti, kun mahdollisilla vastauksilla v_i on todennäköisyydet $P(v_i)$, niin

$$H(P(v_1), \dots, P(v_n)) = \sum_{i=1, \dots, n} -P(v_i) \log_2 P(v_i)$$

- Esim. $H(\frac{1}{2}, \frac{1}{2}) = 2(-\frac{1}{2} \log_2(\frac{1}{2})) = 1$ bitti
- Attribuuttien arvottamiseen sovellettuna haluamme laskea luokka-attribuutin C arvojakaumaan $P(C)$ kohdistuvan muutoksen, kun opetusaineisto S jaetaan attribuutin A mukaan osajoukkoihin

$$H_S(P(C)) - H_S(P(C) | A),$$

missä

$$H_S(P(C) | A) = \sum_{S_i} H_{S_i}(P(C)),$$

kun A jakaa S :n osiin S_i

Oppimisalgoritmin testaaminen

- Jaa esimerkkiaineisto *opetusaineistoksi* ja *testiaineistoksi*
- Sovella oppimisalgoritmia opetusaineistoon, tuota hypoteesi h
- Testaa kuinka suuren osan testiaineiston esimerkeistä h luokittelee oikein
- Em. askelia toistetaan eri opetusaineiston koolla kullakin kerralla vetäen opetusesimerkit satunnaisesti
- Tämän toiminnon kuvaaja on *oppimiskäyrä* (learning curve)
- Vaihtoehtoinen testitapa on *ristiinvaldointi* (cross-validation)