Luonnollisen kielen tilastollinen käsittely

T-61.281 (3 ov) L

Kevät 2003

Luennot: **Timo Honkela**
Laskuharjoitukset: **Vesa Siivola**

Luentokalvot: Krista Lagus (päivityksiä: Timo Honkela)

9. Markov-mallit

9.1 Näkyvät Markov-mallit

Olkoon satunnaismuuttuja X jolla ajanhetkillä $1 \dots t$ arvot $X = (X_1, \dots, X_t)$. Satunnaismuuttujan arvo jokin tiloista $S = s_1, \dots, s_N$.

Satunnaismuuttujan arvot peräkkäisinä ajanhetkinä ovat toisistaan riippuvaisia. Yleisessä tapauksessa (ei-Markov) tn:t riippuvat koko (äärettömästä) historiasta, eli siitä miten tämänhetkiseen tilaan tultiin.

Markov-ominaisuudet:

1. **Äärellinen horisontti:**

$$P(X_{t+1} = s_k | X_1, \dots, X_t) = P(X_{t+1} = s_k | X_t) \tag{1}$$

2. **Aikariippumattomuus (stationaarisuus):**

$$P(X_2 = s_k | X_1) = P(X_3 = s_k | X_2) = P(X_{t+1} = s_k | X_t) \tag{2}$$

Markov-ketjun esitys taulukkona

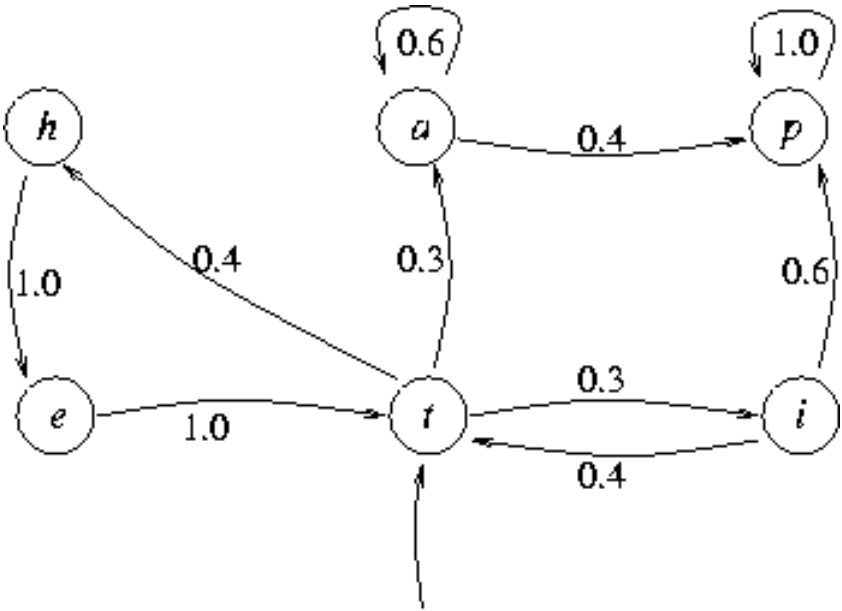
Tilojen joukko $S = \{a, e, i, h, t, p\}$

Tilasiirtymätn:t $a_{ij} = P(X_{t+1} = s_j | X_t = s_i)$ voidaan esittää 2-ulotteisena taulukkona:

$a_{i,j}$	a	e	i	h	t	p
a	0.6	0	0	0	0	0.4
e	...					...
i					0.6	0.4
h	...					...
t	0.3	0	0.3	0.4	0	0
p	...					...

Lisäksi määriteltävä tilojen aloitustodennäköisyydet: esim. ylimääräisen alkutilan s_0 avulla josta tilasiirtymätn muihin tiloihin taulukossa.

Esittäminen probabilistisena äärellisenä tila-automaattina



- Äärellinen tila-automaatti eli Finite-State-Automaton, FSA: tilat piirretään ympyröinä, tilasiirtymät kaarina.

- Mikäli automaattia käytetään lukemaan jotain syötejonoa (ja esim. tarkastamaan onko se sallittu), tai tulostamaan kielessä sallittuja symbolijonoja, voidaan kaariin piirtää ko. tilasiirtymässä luettava/tulostettava symboli.

- epädeterministinen automaatti: yhdestä tilasta johtaa ulos monta kaarta (joissa sama symboli, ts. ei ko. tilassa tiedetä mihin tulisi mennä)

- Painotettu tai probabilistinen automaatti: jokaisella kaarella tn, ja yksittäisestä tilasta ulosmenevien kaarien tn:ien summa $=1$.

- Näkyvä Markov-ketju vastaa äärellistä probabilistista automaattia jossa tilajono on tunnettu. Tilajonoa voidaan siis pitää automaatin tulosteena.

Tilajonon todennäköisyys:

$$
\begin{aligned}
P(X_1 \dots X_T) &= P(X_1)P(X_2|X_1)P(X_3|X_1X_2)\dots P(X_T|X_1\dots X_{T-1}) && (3)\\
&= P(X_1)P(X_2|X_1)P(X_3|X_2)\dots P(X_T|X_{T-1}) && (4)\\
&= \pi_{X_1}\prod_{t=1}^{T-1} a_{X_tX_{t+1}} && (5)
\end{aligned}
$$

jossa $a_{X_tX_{t+1}}$ on siirtymätodennäköisyys tilasta hetkellä t tilaan hetkellä $t+1$ (katsotaan tn:t taulukosta tai tilakoneen kaarista) ja π_{X_1} on tilan X_1 aloitustodennäköisyys.

N-grammimalli Markov-mallina:

- Esim. trigrammimallissa $P(w_3|w_1w_2)$ merkitään tilojen joukko S on kaikkien 2-grammien joukko $S = \{w_1w_1, w_1w_2, \ldots w_1w_N, w_2w_1, \ldots w_2w$
- Rajallisen pituiset historiat voidaan aina esittää äärellisenä joukkona tiloja, jolloin 1. Markov-oletus täyttyy.
- Markov-mallit erotellaan historian pituuden m mukaan, ja puhutaan m:nnen asteen Markov-mallista.
- $n$grammimalli on siis $n-1$:nnen asteen näkyvä Markov-malli.

9.2 Piilo-Markov-Malli (Hidden Markov Model, HMM)

- Tilat (set of states):
  $S = \{s_1, \ldots s_N\}$
- Havaintovaihtoehdot (output alphabet):
  $K = \{k_1 \ldots k_M\}$
- Tilojen alkutodennäköisyydet (initial state probabilities):
  $\Pi = P(s_i) = \pi_i$
- Tilasiirtymätodennäköisyydet (state transition probabilities):
  $A = \{a_{ij}\}, 1 \leq i, j \leq N$
- Havaintotodennäköisyydet kussakin tilassa (symbol emission probabilities): $B = \{b_{jk}\}$
- Tilajono (state sequence):
  $X = \{X_1 \ldots X_{T+1}\}$
- Havaintojono (output sequence):
  $O = \{o_1 \ldots o_t\}$

Ero näkyvään Markov-malliin: vaikka havaintojono tunnetaan, tilajono ei ole yksikäsitteisesti tunnettu.

Esimerkit:

Tilat: Hyväntuulinen / Ärsyyntynyt
Havainnot: 'Upeaa!', 'Lähdetäänkö leffaan?', 'Miksei kukaan ole tyhjentänyt roskista?'

Tilat: Joukko uurnia
Havainnot: Värejä (nostettu uurnista)

9.3 HMM-mallin käyttäminen

- Havaintojonon tuottaminen
- Annetun havaintojonon todennäköisyyden laskeminen
  - Forward-algoritmi (eteenpäinlaskenta)
  - Backward-algoritmi (taaksepäinlaskenta)
- Todennäköisimmän tilajonon etsiminen
  - Viterbi-algoritmi
- HMM:n parametrien estimointi

9.4 Havaintojonon (output sequence) tuottaminen

Jos HMM-malli on annettu, havaintojonojen tuottaminen siitä on triviaalia seuraavasti:

t:=1
Arvotaan alkutila X_1 tilojen aloitustn:ien perusteella: $P(s_i) = \pi_i$
while (1)
. . . . . Tilassa i arvotaan tilasiirtymä $s_i \rightarrow s_j$ tn:llä a_{ij}
. . . . . Arvotaan tilassa j tulostettava symboli o_t=k tn:llä b_{jk}
end

9.5 Havaintojonon todennäköisyyden laskeminen

Jos on annettuna tietty havaintojono $O = o_1 \dots o_T$ ja malli $\mu = (A, B, \Pi)$, miten saadaan tehokkaasti laskettua havaintojonon O todennäköisyys mallissa, $P(O|\mu)$?

Ratkaisu 1

Suoraviivainen, tehoton ratkaisu: summataan havaintojonon tn kaikkien tilajonojen yli:

$$P(O|\mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_i} \prod_{t=1}^{T} a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t} \quad (6)$$

Havaintojonon tn:n laskeminen, ratkaisu 2

Edellistä huomattavasti tehokkaampi ratkaisu saadaan *dynaamisella ohjelmoinnilla*, kun huomataan että jonoilla on yhteisiä alijonoja, ja lasketaan tn:t kunkin alijonon osalta vain kerran, eli

Forward-algoritmi: (eteenpäinlaskenta)

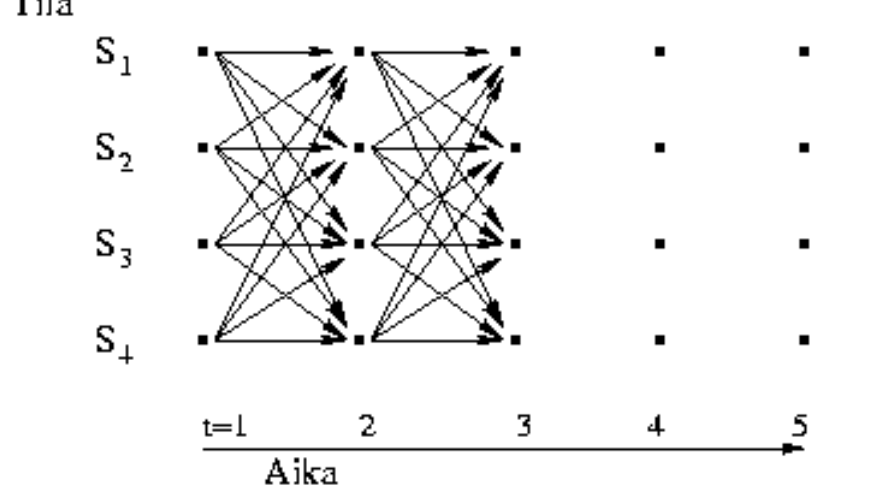
Kerätään tähänastisia tn:iä tilakohtaisiin apuparametreihin α_i:

- $\alpha_i(t) = P(o_1 o_2 \ldots o_t - 1, X_t = i|\mu)$

Kuljetaan hilaa ajassa eteenpäin:

- Alustetaan tilojen alkutodennäköisyyksillä: $\alpha_i(1) = \pi_i$
- Induktioaskel: summataan i:n tulokaarista saapuva tn-massa: $\alpha_i(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{ijo_t}$
- Summataan: $P(O|\mu) = \sum_{i=1}^{N} \alpha_i(T+1)$

Forward–algoritmi



Apuparametri $\alpha_j(t)$ kussakin hilapisteessä

Backward-algoritmi (taaksepäinlaskenta)

Toimii samoin mutta ajassa päinvastaiseen suuntaan.

Eteenpäin- ja taaksepäinlaskenta voidaan myös yhdistää missä kohdassa aikajanaa tahansa.

Dynaaminen ohjelmointi

Esimerkki dynaamisesta ohjelmoinnista www-sivulla
$http://www.cis.hut.fi/\sim tho/study/wld.html$

9.6 Todennäköisimmän tilajonon etsiminen

Tehtävä: etsi tilajono joka parhaiten selittää havaintojonon O.

Etsitään *todennäköisin tilajono*, joka on tuottanut havaintojonon.

Viterbi-algoritmin perusteet

(Tunnetaan myös nimillä DP alignment, Dynamic Time Warping, one-pass decoding)

- etsii havainnoille todennäköisimmän kokonaisen tilajonon tehokkaasti
- tallettaa jokaiseen hilapisteeseen siihenastisen todennäköisimmän polun:

$$\delta_j(t) = \max_{X_1 \ldots Xt-1} P(X_i \ldots X_{t-1}, o_1 \ldots o_{t-1}, X_t = j|\mu) \qquad (7)$$

Viterbi-algoritmi

- Initialisoi tilojen alkutodennäköisyyksillä: $\delta_j(1) = \pi_j$
- Induktioaskel: valitse edellinen tila, josta tulee suurin jonon tn:
  talleta tn: $\delta_j(t+1) = \max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t}$
  talleta pointteri ko. tilaan: $\phi_j(t+1) = \arg\max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t}$
- Lopuksi: Lue hilaa lopusta taaksepäin seuraten ϕ_j:n talletamia kaaria ja kerää todennäköisin tilajono.
- Tasatilanteet voidaan ratkaista arpomalla.
- Keskeinen ero forward-algoritmiin: *todennäköisyyksien summan* sijaan talletetaan *maksimi-tn*.
- Joskus yhden parhaan jonon sijaan halutaan n parasta jonoa (n-best-list). Tällöin talletetaan jokaisessa hilapisteessä $m < n$ parasta tn:ää ja tilaa.

9.7 HMM-mallin parametrien estimointi

Jos annettuna on tietty havaintojono O (opetusdata), etsitään HMM-mallin parametreille $\mu = A, B, \pi$ arvot, jotka uskottavimmin selittävät havainnot. Sovelletaan MLE-estimointia:

$$\arg\max_{\mu} P(O_{\text{opetusdata}}|\mu) \tag{8}$$

Tälle ei ole tiedossa analyyttistä ratkaisua.

Baum-Welch eli forward-backward-algoritmi

- Iteratiivinen optimointimenetelmä (sovellus EM-algoritmista, jota tarkastellaan myöhemmin). Periaate: vuorotellaan seuraavia kahta askelta:

  1. Pidä mallin parametrit vakiona, laske $P(O|\mu)$ ja kerää tiedot siitä miten O:n tn-massa jakautui mallin hilaan (eli miten monta kertaa kuljettiin mitäkin reittiä).
  2. Uudelleenestimoi mallin parametrit $\mu \rightarrow \hat{\mu}$ siten, että O olisi mahdollisimman todennäköinen.

- Varmasti joka kierroksella $P(O|\hat{\mu}) \geq P(O|\mu)$
- Malli voidaan initialisoida satunnaisesti tai jollain nopealla raa'alla menetelmällä.
- Iteroidaan, kunnes datan tn ei enää muutu merkittävästi.

- Löytää lokaalin maksimin, ei välttämättä globaalia (kustannusfunktio on suuren joukon parametrejä mitä ilmeisimmin epälineaarinen funktio).

Parametrien estimaatit sanallisesti:

$$\hat{\pi}_i = \text{ odotettu frekvenssi tilassa } i \text{ hetkellä } t = 1$$

$$\hat{a}_{ij} = \frac{\text{odotettu transitioiden lkm tilasta } i \text{ tilaan } j}{\text{odotettu transitioiden lkm tilasta } i}$$

$$\hat{b}_{jk} = \frac{\text{odotettu transitioiden lkm tilaan } j \text{ kun havainto oli } k}{\text{odotettu transitioiden lkm tilaan } j}$$

Huomaa, että toinen formulointi on myös mahdollinen: havainto tuotetaan tilasiirtymän yhteydessä, ei tietyssä tilassa. Tällöin viimeksimainittu kaava on muotoa:

$$\hat{b}_{ijk} = \frac{\text{odotettu transitioiden lkm tilasta } i \text{ tilaan } j \text{ kun havainto oli } k}{\text{odotettu transitioiden lkm tilasta } i \text{ tilaan } j}$$

9.8 HMM-mallien käytännön implementointi ja joitakin ongelmia

Kerrotaan paljon hyvin pieniä tn:iä keskenään → ylivuoto-ongelmia.

- Viterbissä vain kertolaskua ja max-operaatiota → voidaan toteuttaa kokonaan logaritmoituna, jolloin kertolaskut summia.
- Baum-Welchissä voidaan käyttää skaalauskertomia, joita kasvatetaan ajan t funktiona.

Parametrien suuri määrä

- tarvitaan paljon dataa, tai
- oletetaan että osa parametreistä jaetaan verkon eri osien kesken ('parameter tying': sidottuja tiloja tai sidottuja tilasiirtymiä), tai
- oletetaan että verkossa on 'rakenteisia nollia', ts. rakenteellisesti mahdottomia tilasiirtymiä

Estimointialgoritmi löytää lokaalin maksimin, ei globaalia

- Ratkaisuna parametrien fiksu initialisointi
- Erityisen tärkeää initialisoida havaintotn:t $B = b_{ijk}$

9.9 Soveltamistavoista

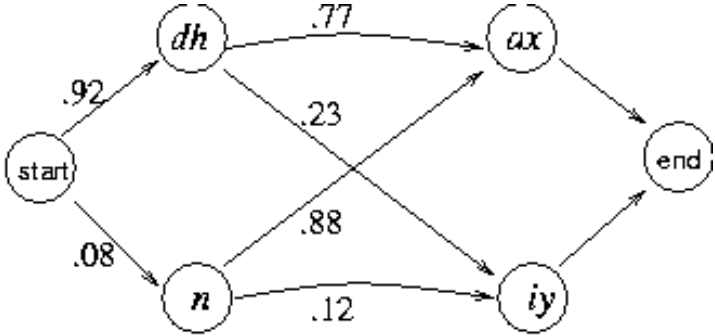
Soveltaminen puheentunnistukseen

Englannin sanojen ääntämisen mallinnus: Jurafsky & Martin (2000).

Havainnot fooneja. Sanaa vastaa joukko tiloja joita voidaan käyttää ko. sanan ääntämisessä.

Lasketaan kullekin sanalle oma HMM joka mallintaa sanan eri lausumistavat ja näiden tn:t.

Havaintojono: foonijono, esim. [dh iy]
Sana: 'the', jonka yksinkertaistettu tilamalli (kuva)



Hyvin yksinkertaistettu ääntämismalli sanalle 'the'

Jos tehtävänä on tunnistaa mikä yksittäinen sana lausuttiin, yhdistetään sanamallit *rinnakkain* yhdeksi isoksi HMM:ksi

Yhdistetty segmentointi + tunnistus

Viterbi-algoritmilla voidaan suorittaa samalla kertaa sekä segmentointi sanoiksi että sanojen tunnistus.

Havaintojono: foonijono, esim. [aa n iy dh ax]
Tuloste: sanajono 'I need the'

- Jokaiselle sanalle oma HMM
- Poimitaan sanaparien esiintymistn:t bi-grammimallista, ja käytetään näitä sanojenvälisiä tn:iä yhdistettäessä sanakohtaiset HMM:t *peräkkäin*
- Huom: näin tehdessä oletetaan, että edellisen sanan ja seuraavan sanan lausuntatavat eivät riipu toisistaan. Oikeasti sanojen välillä riippuvuuksia lausumistavoissa, esim. savolainen viäntää, ja helsinkiläinen sihauttaa s-kirjainta sanasta riippumatta.
- Riippumattomuusoletusten tarkoitus helpottaa estimointia (vähemmän parametreja). Voidaan myös soveltaa pelkästään mallin initialisointiin.

Viterbin ongelma:

- Etsii todennäköisimmän tilajonon, mikä ei ole sama asia kuin todennäköis sanajono.
- Kukin tilajono vastaa jotain sanojen lausuntavaihtoehtoa. Löytää siis todennäköisimmän *lausuntavaihtoehdon*.
- Kuitenkin sanajonon tn on summa sen sanajonon eri lausuntavaihtoehtojen tn:istä.
- Ei voida soveltaa kaikkien kielimallien kanssa (lähinnä bigrammin kanssa, jo trigrammien kanssa ongelmia)
- Eräs vaihtoehto: multi-pass-decoding: Viterbi nopeaan dekoodaukseen joka tuottaa n-best-listan, sitten sovelletaan parempaa kielimallia näistä valitsemiseen.

Aidompi HMM:ien sovellus puheentunnistukseen:

- Havainnot puheen spektristä laskettuja piirrevektoreita.
- Tilat vastaavat fooneja
- Aika ei kulje fooni kerrallaan vaan tietynkestoisina segmentteinä.
- Erikestoiset foonit hoidetaan esim. siten että foonista on kaari myös siihen itseensä.

Muita HMM:ien sovelluksia

- sanaluokkien taggaus (aiheena seuraavalla luennolla)
- geenisekvenssien analyysi / taggaus